

DEGREE PROJECT IN ELECTRICAL ENGINEERING, SECOND CYCLE, 30 CREDITS STOCKHOLM, SWEDEN 2020

An FPGA Implementation of Arbiter PUF with 4x4 Switch Blocks

CAN AKNESIL

KTH ROYAL INSTITUTE OF TECHNOLOGY SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

An FPGA Implementation of Arbiter PUF with 4×4 Switch Blocks

Can Aknesil

School of Electrical Engineering and Computer Science Royal Institute of Technology (KTH), Stockholm, Sweden aknesil@kth.se

Master of Science Thesis in Embedded Systems Supervisor: Elena Dubrova

June 15, 2020

Abstract

Theft of services, private information, and intellectual property have become significant dangers to the general public and industry. Cryptographic algorithms are used for protection against these dangers. All cryptographic algorithms rely on secret keys that should be generated by an unpredictable process and securely stored. The keys are usually stored in a memory, e.g. Flash or fuses. Therefore, the strength of cryptographic protection relies upon the ability of an attacker to extract the keys from the hardware. Modern hardware implementation methods are very advanced, weakening cryptographic algorithms against physical attacks. Finally, memories that provide extra security are expensive to be used in Integrated Circuits (ICs).

As a solution to the memory key storage problem, Physically Unclonable Functions (PUFs) have been proposed. A PUF is an electronic circuit that evaluates responses of hardware to given input stimuli. Due to manufacturing process variations, every IC has different characteristics at the analog level. These variations lead to measurable differences, hence different responses of PUFs implemented on different IC chips.

In this thesis, we are implementing recently proposed 4×4 Arbiter Physically Unclonable Function (APUF) on Field-Programmable Gate Arrays (FPGAs), performing statistical analysis including uniformity, reliability, and uniqueness, comparing hardware overhead of our FPGA design to other APUF variants, providing a mathematical model using homogeneous coordinates, and proposing methods that enable usage of our PUF in real-world applications. We selected this particular type of PUF because it is claimed to be more area efficient than its alternatives while providing strong security and reconfigurability.

According to our analysis, the presented 4×4 APUF design is suitable for many security applications, including identification, authentication, encryption, and key generation. Furthermore, its FPGA area is considerably smaller than the area of 2×2 APUF variants accepting challenges of the same size. However, since uniqueness of our design is lower than desirable, to be used in security applications, our PUF requires repeated invocations and generation of larger keys by combining many responses, thus additional computation during runtime.

Sammanfattning

Stöld av tjänster, privat information och immateriell egendom har blivit betydande faror för allmänheten och industrin. Kryptografiska algoritmer används för att skydda mot dessa faror. Alla kryptografiska algoritmer förlitar sig på hemliga nycklar som ska genereras genom en oförutsägbar process och säkert lagras. Tangenterna lagras vanligtvis i ett minne, t.ex. Flash eller säkringar. Därför beror styrkan på kryptografisk säkerhet på en angripares förmåga att extrahera nycklarna från hårdvaran. Moderna fysiska metoder på hårdvara är mycket avancerade och försvagar kryptografiska algoritmer mot fysiska attacker. Slutligen är minnen som ger extra säkerhet dyra att använda i Integrated Circuits (ICs).

Som en lösning på minnesnyckellagringsproblemet har fysiskt oklonbara funktioner (PUF) föreslagits. En PUF är en elektronisk krets som utvärderar svar från hårdvara på givna input stimuli. På grund av variationer i tillverkningsprocessen har varje IC olika egenskaper på den analoga nivån. Dessa variationer leder till mätbara skillnader, därmed olika svar från PUF: er implementerade på olika IC-chips.

I den här avhandlingen implementerar vi nyligen föreslagna 4×4 Arbiter Physically Unclonable Function (APUF) på fältprogrammerbara gate-arrayer (FPGAs), och utför statistisk analys inklusive enhetlighet, tillförlitlighet och unikhet, jämförande hårdvarukostnader för vår FPGAdesign med andra APUF varianter, ger en matematisk modell med homogena koordinater och föreslår metoder som möjliggör användning av vår PUF i verkliga applikationer. Vi valde den här typen av PUF eftersom den påstås vara mer areaeffektiv än dess alternativ samtidigt som den ger stark säkerhet och omkonfigurerbarhet.

Enligt vår analys är den presenterade 4×4 APUF-designen lämplig för många säkerhetsapplikationer, inklusive identifiering, autentisering, kryptering och nyckelgenerering. Dessutom är dess FPGA resursanvändning avsevärt mindre än området för 2×2 APUF-varianter som accepterar utmaningar av samma storlek. Eftersom unikhet med vår design är lägre än önskvärt, för att användas i säkerhetsapplikationer, kräver vår PUF upprepade åkallelser och generering av större nycklar genom att kombinera många svar, därmed ytterligare beräkning under körning.

Contents

1	Introduction	9				
	1.1 Applications of PUFs	10				
	1.2 4×4 Arbiter PUF	10				
	1.3 PUF security	11				
	1.4 ChipWhisperer Tool-Chain	11				
	1.5 Related Work	11				
2	Implementation	14				
	2.1 4×4 APUF	14				
	2.1.1 Switch Blocks	14				
	2.1.2 Arbiter	17				
	2.2 APUF Driver	19				
	2.3 ChipWhisperer Wrapper	20				
	2.4 Testbed	20				
	2.5 Further Details on Implementation	21				
3	4×4 APUF Model	21				
4	Data Collection & Analysis	24				
	4.1 Format of Responses	24				
	4.2 Influence of Arbiter Paths	24				
	4.2.1 Response Correction	25				
	4.2.2 Analysis of Illegal Responses	25				
	4.3 Uniformity Analysis	25				
	4.4 Reliability Analysis	26				
	4.5 Uniqueness Analysis	26				
5	Results	27				
	5.1 Uniformity Analysis Results	27				
	5.1.1 Distribution of Mutual Order Bits	28				
	5.2 Reliability Analysis Results	29				
	5.3 Uniqueness Analysis Results	30				
	5.4 Illegal Response Analysis	30				
	5.5 Area Comparison	32				
6	Discussion	33				
	6.1 Our 4×4 APUF in the Real World	33				
7	Future Work	35				
8	Conclusion	36				
\mathbf{A}	ppendices	37				
A	Hardware Schematics for 4×4 APUF 37					

B 4×4 **APUF Net Delays**

References

42 48

Glossary

- **6-LUT** 6 Input Loop-Up Table. 32
- **APUF** Arbiter Physically Unclonable Function. 2, 9–14, 16, 18–21, 23–27, 31–33, 35, 36
- ASIC Application-Specific Integrated Circuit. 9
- ${\bf CW}$ ChipWhisperer. 11, 20, 24
- **FPGA** Field-Programmable Gate Array. 2, 9, 11, 14, 15, 20, 21, 24–26, 29, 30, 36
- ${\bf FSM}\,$ Finite State Machine. 19
- IC Integrated Circuit. 2, 15, 16
- **ML** Machine Learning. 11, 13, 14, 36
- PUF Physically Unclonable Function. 2, 9–14, 34, 35
- **RFID** Radio-Frequency Identification. 10
- RNG Random Number Generator. 9, 10
- SSH Secure Shell. 9
- ${\bf TRNG}\,$ True Random Number Generator. 35

List of Figures

1	APUF high-level hardware design.	14
2	Switch block hardware design.	15
3	Two switch blocks with default placement. 4 6-LUTs belonging to	
	the first switch block at right, 4 6-LUTs belonging to the second	
	switch block at left.	16
4	Two switch blocks with manual placement. 4 6-LUTs belonging	
	to the first switch block on top, 4 6-LUTs belonging to the second	
	switch block at the bottom.	17
5	Arbiter hardware design	18
6	APUF driver hardware design.	19
7	APUF driver Moore FSM state diagram. Output bits = (pulse,	
	busy, challenge_enable, response_enable)	19
8	CW305 Artix FPGA Target board (right) and CW1173 ChipWhispe	rer-
	Lite capture board (left).	21
9	Response histogram for 24 stage 4×4 APUF (ignoring illegal	
	responses).	27
10	Hamming weight distribution for responses divided by 3 for 24	
	stage 4×4 APUF (ignoring illegal responses)	28
11	Hamming weight distribution of legal responses (mutual order	
	bits) for 24 stage 4×4 APUF.	29
12	Illegal responses histogram for 24 stage 4×4 APUF	31
13	Hamming weight distribution of illegal responses (mutual order	
	bits) for 24 stage 4×4 APUF.	31
14	Bitwise transitions from legal to illegal responses for 4 stage 4×4	
	APUF	32
15	Contour diagram for P (uniqueness probabilities)	35
16	Vivado schematics for 4×4 APUF	37
17	Vivado schematics for 4×4 APUF (Front closeup)	37
18	Vivado schematics for 4×4 APUF (Back closeup)	37
19	Vivado schematic for 4×4 APUF permutation component	38
20	Vivado schematics for 4×4 APUF switch block	39
21	Vivado schematics for 4×4 APUF switch block multiplexer	40
22	Vivado schematics for 4×4 APUF arbiter	41

List of Tables

1	Average of every mutual order bit.	28
2	The distribution of the number of different responses for 2 FPGA	
	chips	29
3	The percentage of reliable challenges	29
4	The probabilities of a randomly selected challenge to produce its	
	n^{th} likely response, $n = 1, 2, \dots, \dots, \dots, \dots, \dots$	30
5	The percentage of unreliable challenges	30
6	Area comparison of 4×4 APUF to other various $2x2$ APUF variants.	33

1 Introduction

Many security applications rely on secret keys that should be generated by an unpredictable process and stored securely. Secret keys are generally stored in non-volatile memories, disks, or they are hard-wired in the hardware in a way that only allows access to authorized people. One example is private keys used during encrypted communication, such as with the Secure Shell (SSH) protocol. These keys are stored in the disk and their access is controlled by the file permissions. Another example is smart cards (credit cards, SIM cards, etc.) that store a unique key used to identify the owner of the card.

However, attackers can extract the secret key via physical attacks such as micro-probing, laser cutting, glitch attacks, and power analysis. It is also possible to clone the Random Number Generator (RNG) used to generate the secret keys by analyzing previously generated ones and guess the future keys. Even though it is possible to achieve strong randomness via computational complexity, hardware implementations of RNGs can be predicted with reverse engineering [16]. Consequently, there is a need to improve security at the hardware level.

One solution of secure generation and storage of secret keys is PUFs. PUFs exploit random manufacturing process variations of electronic devices such as Application-Specific Integrated Circuits (ASICs) and FPGAs to generate device-specific keys that cannot be cloned [16]. They are built into a chip during manufacturing. This eliminates the need for manual per-device configuration. The keys are generated only when required and do not remain stored on-chip. This provides a higher resistance to physical attacks. Furthermore, they cannot be cloned because it is nearly impossible to fabricate an electronic device with the same manufacturing imperfections.

For many applications, PUFs should be reconfigurable so that they can generate different responses to different inputs, where each input represents a different configuration.

Previous Work A model of a new type of PUF, an APUF with 4×4 switch blocks, is proposed in [8]. It is based on the conventional APUF [16] that uses 2x2 switch blocks. The new design is claimed to provide better resistance against modeling attacks while keeping hardware overhead and computation time low.¹

Our Contribution The contributions of this thesis can be summarized as follows:

- An FPGA implementation of APUF with 4×4 switch blocks.
- Statistical analysis on our implementation including uniformity, reliability, and uniqueness. We also extend our analysis to newly discovered behavior due to unequal delays leading to the arbiter.

 $^{^1{\}rm The}$ comparison with other PUF designs are performed for Xilinx series 7 FPGAs [10] which are also used for our implementation.

- Verification of previously anticipated [10] hardware overhead.
- A model for 4 × 4 APUF based on homogeneous coordinates, which can be generalized to switch blocks of arbitrary size.
- Methods to enable the usage of the presented 4 × 4 APUF design in realworld applications.

Research question Is APUF with 4×4 switch blocks realizable on FPGAs with a sufficient amount of exploitation of manufacturing differences to be used in real-world applications?

1.1 Applications of PUFs

PUFs can be thought of as a unique fingerprint for electronic devices. There are numerous applications of PUFs [3]:

Identification A PUF is embedded into a device, such as a Radio-Frequency Identification (RFID) tag, intended to store an ID. This can reduce the cost since an internal non-volatile memory will not be needed.

Authentication A user owns a device containing a PUF that is used for authentication via a server. The server sends an input to the user who uses it to generate a response from the PUF and sends the response back to the server. Then, the server compares the received response to the one in its database to check its validity. In this scenario, a different input should be used for every authentication to prevent the attackers, listening to the communication between the user and the server, to impersonate the user. This necessitates the definition of "lifetime" of the PUF, which should expire before disclosure of a sufficient amount of input-response pairs to break the security.

Random Number Generation A PUF can be used as an unpredictable RNG that behaves differently on every device. Generated numbers can be used as secret keys if desired.

1.2 4×4 Arbiter PUF

A general APUF constitutes of symmetrically placed paths. It is important that the propagation times of a signal through all of these paths are as close as possible. An APUF is used by simultaneously sending a signal, called a "stimuli", to each of these paths and comparing the propagation delays with an arbiter that generates a response representing the order of the delays. In real life, due to the random manufacturing differences, the delays slightly differ resulting in unique responses for every chip.

 4×4 APUF proposed in [8] contains a structure called " 4×4 switch block" as the building block. A 4×4 switch block is capable of mapping 4 inputs to

4 outputs in every 24 (4!) possible ways, and is controlled by a 5-bit selector called the "challenge". The most important property of a switch block is that all the 16 (4^2) paths connecting one input to one output (among which 4 of them are selected according to the challenge) has theoretically the same delays, resulting in 16 distinct delay values per chip when implemented in real life.

A 4×4 APUF is constructed by concatenating multiple switch blocks, creating 4 long paths whose delays are reconfigured via a 5-bit challenge per switch block. A high-level diagram of 4×4 APUF is shown in Fig. 1.

1.3 PUF security

Reconfigurability of a PUF is very important in terms of security. A software clone of a PUF can be created given a sufficient amount of challenge-response pairs. Due to this, PUFs are divided into two broad categories: "weak" PUFs that accept only one or few different challenges and "strong" PUFs that accept a large amount of challenges, which makes them more secure.² 4×4 APUF implemented in this thesis is a strong PUF.

PUFs can be attacked by collecting and analyzing certain amount of challengeresponse pairs. There are many ways to attack a PUF, such as "reverse engineering attack", during which the architecture of the PUF is modeled to create a software clone that can later be used as the PUF itself to get responses to the challenges that have not been collected, and "collision attack", during which identical responses of different PUFs are analyzed to guess other responses [17].

Delay-based PUFs are vulnerable to Machine Learning (ML)-based modeling attacks [20], categorized under reverse engineering attacks. 4×4 APUF implemented in this thesis is a delay-based PUF. During a modeling attack, paths in an APUF are modeled, the model is then trained with the help of ML algorithms using collected challenge-response pairs.

1.4 ChipWhisperer Tool-Chain

ChipWhisperer (CW) is a free tool-chain for side-channel power analysis and glitching attacks [5]. In this thesis, we used a target FPGA board on which our PUF is implemented, a capture board that is used to communicate with the FPGA board, and a software framework that is used to communicate with both target and capture boards, all provided by CW. We used this setup to collect challenge-response pairs from our PUF.

1.5 Related Work

The biggest weakness of PUF is that modeling attacks can break it given a sufficient amount of time and challenge-response pairs. The literature is full of

²The difficulty of breaking a PUF is inversely proportional to the percentage of challengeresponse pairs that are revealed. So, security a PUF provides can be traded with its lifetime. Strong PUFs can be used with higher number of challenge-response pairs, which gives them a longer lifetime. In our case, a 28 stage 4×4 APUF accepts 24^{28} challenges, which makes infeasible to collect a sufficient amount of challenge-response pairs to easily brake its security.

novel PUF designs, mostly improvements to 2x2 APUF, aiming resistance to the modeling attacks. This is generally achieved either by increasing the complexity of the design to make the model difficult to build, and/or by increasing the number of parameters in the model to increase the required time and challenge-response pairs to brake the design. Many examples from the literature are presented in the following paragraphs.

Feed-Forward Arbiter PUF [16] FF-APUF improves 2x2 APUF by determining some of the challenges using the output of intermediary arbiters (feedforward arbiters) put in between some of the switch blocks, rather than taking all the challenges from the user. Modeling attacks performed on regular 2x2 APUF do not work on FF-APUF and more sophisticated model is needed to imitate the new behavior.

Non-Linear Arbiter APUF [16] Improves 2x2 APUF by modifying the mapping from the challenge to switch block delays to make it non-linear. This modification makes the PUF resistant to physical probing attacks.

XOR-PUF [16] The response is produced by XORing responses from multiple APUFs. This multiplies the number of delays by the number of APUFs, thus, making modeling attacks more difficult.

Lightweight Secure PUF [18] Wraps multiple 2x2 APUFs by an "interconnect network", a logic circuit through which the challenge passes before being distributed to individual PUFs, "input networks" per PUF, through which challenges destined to individual PUFs pass after being processed by the interconnect network, and an "output network" outputs of all the PUFs pass to generate the final response. This complicates the modeling attacks.

Reconfigurable Optical PUF [15] Proposes "a structure that consists of a polymer containing randomly distributed light scattering particles". This structure produces a "steady" speckle pattern when exposed to a laser beam. The structure can be reconfigured by exposing it to a laser beam that is outside of operating conditions.

Phase Change Memory (PCM) based Reconfigurable PUF [15] PCM works by subjecting it to a specific heating pattern, which induces the resistivity of the material to change. High resistivity represents '1' and low resistivity '0'. Also, the intermediate states can be achieved with a writing operation that cannot be controlled; however, these states can be easily read. As a result, a long-lived random state can be created, which can be reconfigured at will.

The difference between "Reconfigurable Optical PUF" and "PCM based Reconfigurable PUF" compared to other PUFs presented in this section is that challenge-response behavior is uncontrollable. After reconfiguration, the previous challenge cannot be regenerated by another reconfiguration. **Logically Reconfigurable PUF** [11] This is an implementation of a use case of PUFs: securely storing and reading a secret key. It combines a PUF with a non-volatile memory that stores state information. The state information is used to hash the response of the PUF and the hashing process can be reconfigured by changing the stored state.

Intrinsically reconfigurable D-RAM based PUF (D-PUF) [22] Normally, D-RAM based PUFs are "weak", thus, open to modeling attacks. This design proposes a "strong" D-RAM PUF. Manipulating the pausing time-interval during refresh operation of the memory changes the challenge-response behavior of the PUF. In other words, reconfiguration is achieved by changing the pausing interval.

 $\mathbf{R^{3}PUF}$ [12] $\mathbf{R^{3}PUF}$ is based on memristive devices. It uses the resistance variations in memristive devices not only among CMOS devices but also among different reprogramming cycles within the same device. The response is generated by comparing two or more memristive-devices. The advantage of this design is that the responses are highly reliable (error-free).

Interpose PUF [19] Internally uses 2 XOR-PUFs, the upper layer and the lower layer. The challenge taken from the user is used as the challenge to the upper layer without modification, while the challenge to the lower layer is created by interposing the response of the upper layer into the challenge. This design makes modeling attacks more difficult while staying lightweight and strong (in the PUF sense).

MPUF [21] Multiplexer-based PUF (MPUF) uses responses of several APUFs as inputs and selectors of a multiplexer. The final output is the output of the multiplexer. This design aims to improve the vulnerability of the challenge against modeling and statistical attacks, and also its reliability.

Resistive RAM-based String Arbiter PUF [14] Proposes a string APUF based on a modified Resistive RAM. One key property of this design is that the APUF is realized within the memory array turning it to an APUF. Another key property is that it can be configured for different numbers of stages, which can be used to hide the number of bits in the challenge. This provides an additional layer of protection.

Majority Vote XOR-PUF [24] Solves the problem of XOR-PUF that is it cannot be realized with more than 12 internally used PUFs due to noise. Proposes majority voting for every PUF before XORing. This enables larger XOR-PUFs to be built, providing more resistant to ML attacks. **FF-XOR-PUF** [2] FF-XOR-PUF is a combination of Feed-Forward Arbiter PUF and XOR-PUF, where each PUF in the XOR-PUF is an FF-PUF. Various versions with different kinds of FF-PUFs are proposed. This design aims to overcome the issues in XOR-PUF, such as, vulnerability to ML attacks and response instability.

FPGA implementation of a challenge pre-processing structure APUF [13] Proposes a pre-processing structure through which the challenge passes before going into the PUF. Each challenge bit passes through a modified version of RS-flip-flop, where the output of every flip-flop is acting at the same time as inputs of the adjacent flip-flops. This creates an additional behavior defined by the manufacturing differences, apart from the actual PUF itself. The aim is to make the design resistant to ML attacks.

2 Implementation

2.1 4×4 **APUF**

Our FPGA implementation for 4×4 APUF constitutes of several switch blocks, and an arbiter. The schematics of the hardware design is shown in Fig. 1.



Figure 1: APUF high-level hardware design.

Every switch block has 4 inputs and 4 outputs. The inputs can be mapped the outputs in 24 (4!) different configurations specified by the 5-bit challenge representing numbers in the interval [0, 23].

The arbiter compares every 6 pairs of the outputs belonging to the last switch block $\binom{4}{2} = 6$, and generates a 6-bit response representing the order of the pulse for each pair.

In the rest of this thesis, we call an APUF with n switch blocks as an n-stage APUF.

2.1.1 Switch Blocks

The hardware design of a switch block is shown in Fig. 2.



Figure 2: Switch block hardware design.

A switch block includes 4.4×1 multiplexers, each producing one of the 4 outputs. The 4-bit input of all the multiplexers are connected to the 4-bit input of the switch block. A challenge translation module is implemented to translate the 5-bit challenge that represents numbers in the interval [0, 23], into 8 bits: 2-bit selectors for every multiplexer.

Equality of Paths In theory, in our implementation, the length of all 16 (4^2) paths in a switch block should be identical. In the ideal case, all these paths must be implemented symmetrically on the IC chip and the only factor causing the delays to differ must be the manufacturing differences. However, when it comes to implementing a design on an FPGA, there are many other factors that can cause the delays to differ; such as, the compiler decisions on the placement of the cells, the routing between these cells, the internal implementation of these cells and the look-up tables, etc. These additional factors can dominate the

manufacturing differences and cause it to end up with predictable similarities between APUFs implemented on different IC chips.

In this thesis, we implemented and evaluated two different kinds of hardware placement for switch blocks and the arbiter. First one is the default placement performed by the design tool. In the second one, we manually placed all the switch blocks and the arbiter. The manual placement is performed to make the paths within and between switch blocks, and paths between the last switch block and the arbiter as symmetrical as possible.

Two switch blocks and routing between them are shown in Fig. 3 (default placement) and Fig. 4 (manual-placement).³



Figure 3: Two switch blocks with default placement. 4 6-LUTs belonging to the first switch block at right, 4 6-LUTs belonging to the second switch block at left.

 $^3\mathrm{Only}$ 4 out of 8 6-LUTs of a switch block, the ones belonging to the multiplexers, are manually placed.



Figure 4: Two switch blocks with manual placement. 4 6-LUTs belonging to the first switch block on top, 4 6-LUTs belonging to the second switch block at the bottom.

The results, however, are better with the default placement. Accordingly, we only present results for the default placement.

Individual delays in the switch blocks are presented in the appendix.

2.1.2 Arbiter

The hardware design of an arbiter is shown in Fig. 5.



Figure 5: Arbiter hardware design.

The arbiter module includes 6 D-flip-flops. The D-input and the clock input of every one of them are connected to one of the input pairs it compares. If the path connected to the D-input is faster, the output is 1, otherwise, it is 0. In the rest of the thesis, the output of each of these every flip-flops is called a "mutual order bit".

The arbiter design proposed in the previous work [10] included the translation of the mutual order bits into a 5-bit number that represents one of the permutations of the 4 paths. We excluded this translation module and directly used mutual order bits.

The translation module proposed in [10] caused most of the illegal responses to be mapped to 0. This disturbed the response distribution. In other words, how the translation module was implemented influenced the outcome of statistical analysis performed in this thesis. To make the APUF responses independent from the implementation of the translation module, and to be able to detect illegal responses, the translation module is excluded. Illegal responses are explained in the rest of the thesis in detail. **Arbiter placement** There is a fundamental difficulty at the placement and routing of arbiter flip-flops due to the fact that design tools handle clock paths differently: Clock paths and the components through which they pass within the cells are different from regular paths; furthermore, the clock signal is shared by all the flip-flops in a cell etc. During this thesis, we tried to place arbiter flip-flops manually, as we did with the switch blocks; however, since the default placement results were better, we present only them in this thesis.

2.2 APUF Driver

The process of getting a response from the APUF is as such: One should set the challenge to configure the switch blocks, then send a rising edge that is forked to all of the inputs of the first switch block, and finally read the mutual order bits from the arbiter. An APUF driver module is implemented as a Moore Finite State Machine (FSM) to perform this process. The hardware design of the driver is shown in Fig. 6. The state diagram of the FSM is shown in Fig. 7.



Figure 6: APUF driver hardware design.



Figure 7: APUF driver Moore FSM state diagram. Output bits = (pulse, busy, challenge_enable, response_enable).

2.3 ChipWhisperer Wrapper

At the topmost level, the challenges are applied and responses are received by a computer program. The communication between the computer and the FPGA containing the APUF is performed via the CW tool-chain.

This wrapper contains 3 parts provided by CW: (1) a hardware design that wraps the APUF, (2) a set of circuit boards including the FPGA chip on which APUF is implemented, and (3) a software framework used to in the developed software that communicates with the APUF.

The tool-chain encapsulates the design unit implemented on the FPGA, abstracts the inputs to the unit as the "plaintext" and the "key", and abstract the output from the unit as the "ciphertext". In our case plaintext is the combined challenge to the switch blocks. It is 5n bits in size where n is the number of stages. The ciphertext is the response taken from the arbiter (mutual order bits), 6 bits in size. Since the key is the FPGA chip itself, that input is not in use.

Since CW offers 128 bits for the plaintext and the ciphertext, some of the spare bits are used to get a predefined signature as part of the response for debugging purposes.

In our thesis, CW is only used to apply challenges to the APUF and receive responses in an easy way. Even though performing other features of the toolchain, such as power trace capturing and correlation power analysis, is not in the scope of this thesis, they are among the possible future work. So, using this tool-chain is also a preparation for future research.

Once challenge-response pairs are received using CW, all the data analysis software is written independently from the tool-chain, without needing the CW framework.

2.4 Testbed

- FPGA target board: 2 distinct CW305 Artix FPGA Target boards [7] with Xilinx Artix-7 XC7A100T FPGA [1]
- Capture board: CW1173 ChipWhisperer-Lite [6]
- Hardware design tool: Xilinx Vivado v2019.2.1 (64-bit) [23]
- Computer operating system: Ubuntu 18.04.4 LTS
- Software: ChipWhisperer 5.0 [4]

A photo of the target FPGA board and the capture board together is shown in Fig. 8. Other ends of the two USB cables connected to each board are connected to the computer.



Figure 8: CW305 Artix FPGA Target board (right) and CW1173 ChipWhisperer-Lite capture board (left).

2.5 Further Details on Implementation

This particular implementation of APUF is not intended to be resistant to hardware attacks, such as power and side-channel analysis, but sufficient to evaluate the statistical properties of the proposed APUF implemented on FPGAs.

We developed our APUF hardware design in VHDL, with "number of stages" as a generic input.

Since an APUF design does not make sense in the digital level but in the analog level, the design compiler corrupts the design during optimizations. To prevent this, we disabled compiler optimizations for switch blocks and the arbiter. This was done via "DONT_TOUCH" compiler attribute of Vivado.

During the development of the APUF hardware design, alongside our main development board, it is also loaded on Nexys 4 DDR FPGA board and controlled via switches and LEDs.

3 4×4 APUF Model

We propose a new mathematical model for 4×4 APUF using homogeneous coordinates.

The behavior of a switch block can be expressed as two consecutive operations with 4 inputs and 4 outputs: "permutation" that changes the order of the inputs and "translation" that adds certain delays to the input.

We will express the input and the output, $\mathbf{i} = [i_1, i_2, i_3, i_4]^T$ and $\mathbf{o} = [o_1, o_2, o_3, o_4]^T$, in homogeneous coordinates as $\mathbf{i} = [i_1, i_2, i_3, i_4, 1]^T$ and $\mathbf{o} = [o_1, o_2, o_3, o_4]^T$.

 $[o_1, o_2, o_3, o_4, 1]^T$.

This will enable the translation operation to be expressed with a matrix multiplication.

Permutation This operation can be performed by multiplying the input with the matrix P, created using 4 distinct standard basis vectors put in the desired permutation. An example of a permutation operation that changes the order of the first and the second inputs is as follows:

$$\underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\boldsymbol{P}} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ 1 \end{bmatrix} = \begin{bmatrix} i_2 \\ i_1 \\ i_3 \\ i_4 \\ 1 \end{bmatrix}$$

The bold part of \boldsymbol{P} shows the standard basis vectors.

Translation This can be achieved by multiplying the input with the matrix T as follows:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \mathbf{\Delta d_1} \\ 0 & 1 & 0 & 0 & \mathbf{\Delta d_2} \\ 0 & 0 & 1 & 0 & \mathbf{\Delta d_3} \\ 0 & 0 & 0 & 1 & \mathbf{\Delta d_4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ 1 \end{bmatrix}}_{\mathbf{T}} = \begin{bmatrix} i_1 + \Delta d_1 \\ i_2 + \Delta d_2 \\ i_3 + \Delta d_3 \\ i_4 + \Delta d_4 \\ 1 \end{bmatrix}$$

The combined operation for permutation and translation can be expressed with matrix S, which represents the compete behavior of a switch block. Corresponding matrix S created with above P and T is in the following form:

$$S = TP = egin{bmatrix} 0 & 1 & 0 & 0 & \Delta {
m d}_1 \ 1 & 0 & 0 & 0 & \Delta {
m d}_2 \ 0 & 0 & 1 & 0 & \Delta {
m d}_3 \ 0 & 0 & 0 & 1 & \Delta {
m d}_4 \ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Delay values $\Delta d_1, \ldots, \Delta d_4$, and the order of the bases vectors are functions of the challenge of the switch block. These functions can be expressed using 16 distinct delay parameters belonging to 16 paths.

The combined behavior of N switch blocks can be expressed by multiplying each matrix S, belonging to every switch block. The resulting matrix is in the

following form:

$$\prod_{n=1}^{N} \boldsymbol{S}_{n} = \begin{vmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & D_{1} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & D_{2} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} & D_{3} \\ p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} & D_{4} \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

where $p_{m,n}$ are elements in the combined permutation matrix and constitutes of distinct standard basis vectors and D_1, \ldots, D_4 are combined delay values.

The input of the first switch block, the stimuli, can be defined as $s = [0, 0, 0, 0, 1]^T$ since there is no accumulated delay. Then, the equation that binds the stimuli to the output of the last switch block is as follows:

$$\begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & D_1 \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & D_2 \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} & D_3 \\ p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} & D_4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ 1 \end{bmatrix}$$

The model The final delays D_1, \ldots, D_4 going to the arbiter can be expressed as follows:

$$\begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ 1 \end{bmatrix} = \begin{pmatrix} N \\ \prod_{n=1}^N \boldsymbol{S}_n \end{pmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

This model can be applied to APUFs with switch blocks of arbitrary size easily by changing the matrix sizes.

Extension for arbiter paths The influence of the paths from the last switch block to the arbiter flip-flops can be added to the model with a matrix \mathbf{F} that transforms the 4 delays D_1, \ldots, D_4 into 6 delay differences $\Delta D_{1,2}, \Delta D_{2,3}, \Delta D_{3,4}, \Delta D_{1,3}, \Delta D_{2,4}, \Delta D_{1,4}$ going to the flip-flops as follows:

$\Delta D_{1,2}$		[1	-1	0	0	$e_{1,2}$	
$\Delta D_{2,3}$		0	1	-1	0	$e_{2,3}$	$\begin{bmatrix} D_1 \end{bmatrix}$
$\Delta D_{3,4}$		0	0	1	$^{-1}$	$e_{3,4}$	D_2
$\Delta D_{1,3}$	=	1	0	-1	0	$e_{1,3}$	D_3
$\Delta D_{2,4}$		0	1	0	$^{-1}$	$e_{2,4}$	D_4
$\Delta D_{1,4}$		1	0	0	$^{-1}$	$e_{1,4}$	1
1		0	0	0	0	1	
		_		-			
				F			

where $e_{m,n}$ is error introduced by the flip-flop that compares paths m and n. The extended model is as follows:

$$\begin{bmatrix} \Delta D_{1,2} \\ \Delta D_{2,3} \\ \Delta D_{3,4} \\ \Delta D_{1,3} \\ \Delta D_{2,4} \\ \Delta D_{1,4} \\ 1 \end{bmatrix} = \boldsymbol{F} \left(\prod_{n=1}^{N} \boldsymbol{S}_{n} \right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

4 Data Collection & Analysis

Data is collected as challenge-response pairs. One pair consists of the response an APUF generates for a particular challenge. A predefined set of challenges was applied to different APUF setups. In this thesis, the process of applying the set of challenges for a particular setup is called an "experiment". The parameters defining a setup are (1) number of stages of the APUF, (2) the unique FPGA chip on which APUF is implemented. Also each experiment is performed multiple times to perform reliability analysis.

1, 2, 3, 4, and 24 stages are used in the experiments. All possible challenges were applied to APUFs with 1, 2, 3, and 4 number of stages, while a randomly chosen set of challenges was applied to a 24 stage APUF since considering a challenge set of such large size is not realizable. All these experiments were repeated for two different FPGA chips.

24 stage APUF is the main focus during data collection and analysis. CW only supports 128-bit plaintext that is used to communicate the challenge. The largest number of stages having a challenge that can fit into 128 bits is 24 (120-bit challenge). This selection was to keep the communication with the FPGA simple, at the same time, having large enough number of challenges to make brute force attacks infeasible.

4.1 Format of Responses

All of the analysis is performed on individual responses, rather than a concatenation of multiple responses. In a real-world application, it may be necessary to combine multiple responses of an APUF to achieve a larger response (such as a 128-bit response) for utility purposes.

Most of the analyses were performed on 6 mutual order bits. For some of the analysis, we translated the mutual order bits into an integer in the interval [0, 23] representing a particular permutation of incoming signals to the arbiter.⁴

4.2 Influence of Arbiter Paths

During our experiments, an anomaly in the responses was discovered: Delay differences within the path pairs, leading from the last switch block to each flipflop in the arbiter, cause some of the mutual order bits to be incorrect. This

⁴The translation is performed on a computer.

sometimes results in responses that cannot be translated into the permutation information. These responses are called "illegal" in this thesis. And others are called "legal".

This concept can be demonstrated with an example: Let response bits (a, b, c, d, e, f) represent the comparison between output pairs of the last switch block, respectively, (o_1, o_2) , (o_2, o_3) , (o_3, o_4) , (o_1, o_3) , (o_2, o_4) , and (o_1, o_4) . If p_1 , the path leading to o_1 , is faster than p_2 , and p_2 is faster than p_3 , then, p_1 is concluded to be faster than p_3 . So, responses, where $a = b \neq d$, are illegal. Yet, during experiments some of the responses we got were illegal.

This phenomenon was not analyzed in [8] and [10]. The mathematical model proposed in [10] only includes delays of the paths within the switch blocks. The model provided in this thesis considers flip-flop paths, as well as switch block paths.⁵

4.2.1 Response Correction

Some of the illegal responses can be corrected as such: Number of legal responses is 24 (4! is the number of possible permutations of 4 paths), and the number of illegal responses is 40 ($64 = 2^6$ in total). 24 of the illegal responses have only one legal response that is 1 Hamming distance away, while 16 of them have 3 legal responses that are 1 Hamming distance away. Under the assumption that the probability of paths leading to the arbiter causing multiple bit changes is lower enough than only a single bit change, 24 of illegal responses can be corrected.

4.2.2 Analysis of Illegal Responses

Illegal responses were analyzed by looking at the distribution of all 40 illegal responses, their Hamming weight distribution, and most importantly transition (transition from legal to illegal) of individual bits during response correction. Transition information helps to spot the problematic bits in the response.

4.3 Uniformity Analysis

In uniformity analysis, the distribution of responses of one particular APUF to a set of random challenges is evaluated. It is performed intra-APUF. We performed it on data collected from one experiment consisting of a 24 stage APUF implemented on one particular FPGA chip. The challenge set in the experiment consists of 13824 challenges picked randomly in a uniform way.

We looked at the average Hamming weights (distribution of 1s and 0s) in each 6 mutual order bits, and also overall. In the theoretical case, where responses are perfectly uniform, average Hamming weights of each mutual order bit should be 0.5, and the overall Hamming weight of responses should be 3.

The rest of the uniformity analysis was performed 2 times; first, by ignoring the illegal responses; second, by correcting them.

⁵These additional delays can also cause a legal response to flip to another legal response.

We translated mutual order bits into integers in the interval [0, 23], and looked at their distribution.

We also looked at the Hamming weight distribution of the permutation order after dividing it by 3. This is done in [10] inside arbitration to directly achieve uniform bit distribution. After division, the interval [0, 23] is reduced to [0, 7], which uses all the bits uniformly when represented in binary with 3 bits.

4.4 Reliability Analysis

In theory, an ideal APUF should generate the same responses when the same challenge is applied over and over again. However, this is not the case in the real world. Our experiments show that APUF responses are nondeterministic. In other words, the application of the same challenge over and over again does not always produce the same response.

In this thesis, we define the reliability of a challenge to whether it always generates the same response or not. Challenges that always generate the same response are called "reliable challenges", others "unreliable challenges".

Reliability analysis is performed to analyze the reliabilities of every challenge for one particular APUF. It is performed intra-APUF. We ran one experiment 407 times on a 24 stage APUF implemented on one particular FPGA chip. The challenge set in the experiment consists of 13824 challenges picked randomly in a uniform way (the same set used in uniformity analysis).

We looked at the distribution of the responses per every challenge across multiple applications of the same experiment. In the ideal case, where all the challenges are reliable, every challenge generates one single response. So, the ideal distribution is (13824, 0, 0, ...), where there are 407 entries. The n^{th} entry is the number of challenges that generate n different responses over 407 repeated experiments, n = 1, 2, ..., 407. The first entry is the number of challenges that generate one single response during all 407 experiments. Its percentage over the sum of all the entries gives the percentage of reliable challenges.

4.5 Uniqueness Analysis

The essence of APUF is that it should produce different results when implemented on different chips. In other words, responses to a particular set of challenges should be unique for every chip. Uniqueness analysis analyzes the degree of uniqueness. It is performed inter-APUF.

Uniqueness analysis is performed by running one experiment 407 times on 2 different FPGA chips on which a 24 stage APUF is implemented. The challenge set in the experiment consists of 13824 challenges picked randomly in a uniform way (the same set used in uniformity analysis). Later on, analyses are performed on every 407 pairs of experiments (2 experiments for different chips in a pair). Finally, the results for every pair are averaged.

We are defining the theoretically ideal case as such: responses are perfectly random, illegal responses are assumed not to occur, and responses are perfectly reliable. In this ideal case, the probability for each legal response to occur is $\frac{1}{24}$.

The probability of responses to the same challenge from 2 APUFs implemented on different chips to be different, also the expected number of challenges that produce different results on 2 different chips is $\frac{23}{24} = 95.8333\%$. During the analysis, the actual number of challenges that produced different results on 2 different chips are compared with this ideal value.

There is one complication due to unreliable challenges. When responses from different chips differ, it is difficult to understand whether the cause is unreliability or manufacturing differences. Furthermore, as our experiments show, unreliable challenge sets are highly different for every chip. As a solution we also calculate uniqueness by excluding the union of unreliable challenges.

5 Results

Results are for 24 stage 4×4 APUF unless stated otherwise.

5.1 Uniformity Analysis Results

The number of challenges in the experiment was 13758 and 66 of them were illegal. The percentage of legal responses is 99.52%.

Among 66 illegal responses, none was corrected.

Following results are calculated by ignoring the presence of illegal responses, because the influence of illegal response is negligible.

Response distribution is shown in Fig. 9. The Hamming weight distribution of responses divided by 3 is shown in Fig. 10.



Figure 9: Response histogram for 24 stage 4×4 APUF (ignoring illegal responses).



Figure 10: Hamming weight distribution for responses divided by 3 for 24 stage 4×4 APUF (ignoring illegal responses).

5.1.1 Distribution of Mutual Order Bits

Average of every mutual order bit is shown in Table 1.

Bit	1	2	3	4	5	6
Aver.	0.5579	0.4862	0.5207	0.5097	0.4331	0.4901

Table 1: Average of every mutual order bit.

The sum of these numbers, in other words, the average Hamming weight of mutual order bits is 2.9978.

The Hamming weight distribution of legal responses (in mutual order bits format) is shown in Fig. 11.



Figure 11: Hamming weight distribution of legal responses (mutual order bits) for 24 stage 4×4 APUF.

5.2 Reliability Analysis Results

At the result of repeated applications of the same experiment, the distribution of the number of different responses for 2 FPGA chips is shown in Table 2.

# of responses	1	2	3	4	5	6	
Chip 1	12835	957	22	10	0	0	
Chip 2	12714	1063	30	16	1	0	

Table 2: The distribution of the number of different responses for 2 FPGA chips.

The percentage of the challenges that produced only a single response throughout the repeated experiments (first entry of Table 2 divided by the sum of all the entries), in other words, the percentage of reliable challenges is shown in Table 3.

Chip 1	92.8458%
Chip 2	91.9704%

Table 3: The percentage of reliable challenges.

The probabilities of a randomly selected challenge to produce its n^{th} likely response are shown in Table 4, $n = 1, 2, \ldots$

Responses	1^{st}	2^{nd}	$3^{\rm rd}$	4^{th}	5^{th}	
Chip 1	99.0599%	0.9297%	0.0087%	0.0017%	0%	
Chip 2	98.8828%	1.1015%	0.0133%	0.0023%	0%	

Table 4: The probabilities of a randomly selected challenge to produce its n^{th} likely response, $n = 1, 2, \ldots$

On average for all chips, the probability of a randomly selected challenge to produce its most likely response is 98.9714%. These probabilities are calculated by analyzing response histograms for individual challenges.

5.3 Uniqueness Analysis Results

Percentage of unreliable challenges (calculated using results of reliability analysis) for every FPGA chip is shown in Table 5.

Chip 1	7.1542%
Chip 2	8.0295%

Table 5: The percentage of unreliable challenges.

Percentage of the union of unreliable challenges for 2 chips: 13.5923%

The percentage of challenges that produced different results on 2 different chips (illegal challenges included) is 15.1520%.

The same percentage when the union of unreliable challenges from 2 chips are excluded is 9.3261%.

5.4 Illegal Response Analysis

The distribution of illegal responses is shown in Fig. 12. Every 40 slot on the x-axis represents one of the illegal responses.



Figure 12: Illegal responses histogram for 24 stage 4×4 APUF.

The Hamming weight distribution of illegal responses (in mutual order bits format) is shown in Fig. 13.



H. W. Distribution of Illegal Responses (mutual order bits)

Figure 13: Hamming weight distribution of illegal responses (mutual order bits) for 24 stage 4×4 APUF.

Interesting results from 4 stage 4×4 APUF When a random set of challenges are applied to our 4 state APUF, the percentage of illegal responses were much higher: 24.6672%.

This is supposed to be the result of paths between the last switch block and the arbiter flip-flops. In that particular implementation, because of design compiler decisions, look-up table locations, routing, etc., the difference between some pairs of paths to the flip-flops must have turned out to be high, which caused some mutual order bits to stuck at a value for a large number of challenges.

This result is shown in Fig. 14 that shows bitwise transitions of legal responses to illegal responses. The problematic flip-flops are number 0 and 1. This plot is created by correcting 3410 out of 10414 illegal responses.



Figure 14: Bitwise transitions from legal to illegal responses for 4 stage 4 \times 4 APUF.

Although not presented in this thesis, all other analyses for 4 stage APUF are indicating this behavior.

5.5 Area Comparison

The area comparison of 4×4 APUF, implemented in this thesis, to different variants of 2x2 APUFs is presented in Table 6. Table columns are, respectively, APUF type, number of stages, challenge length in bits, response length in bits, number of 6 Input Loop-Up Tables (6-LUTs), number of flip-flops, number of 6-LUTs per one bit of response, number of flip-flops per 1 bit of response.

APUF type	Stages	Ch. bits	Res. len.	6-LUT	FF	6-LUT/res. len	FF/res. len
2x2 APUF [16]	128	128	1	256	1	256	1
8-XOR APUF [16]	128	128	1	2050	8	2050	8
Interpose APUF [19]	128	128	1	514	2	514	2
CRC-APUF [9]	128	128	1	320	1	320	1
4×4 APUF	28	140	4.58	224	6	48.86	1.31

Table 6: Area comparison of 4×4 APUF to other various 2x2 APUF variants.

The number of stages of 4×4 APUF is selected as 28 so that the number of possible challenges is equal or greater than 2^{128} .⁶

Compared to others, 4×4 APUF generates 5-bit response that represents a number in the range [0, 23], rather than a single bit in the range [0, 1]. Since the interval [0, 23] cannot be represented with an integer number of bits without redundancy, we are calculating the response length of 4×4 APUF as $\log_2 24 \cong$ 4.58. This is an advantage because to achieve the same amount of output, other APUFs should be run more than once or more than one instance of them should be implemented in parallel. It should also be underlined that critical path (path from the input stimuli, through all of the switch block, to the arbiter) of 4×4 APUF is shorter than 2x2 APUF, enabling response generation with a higher throughput.

6 Discussion

Repeating the research question: Is APUF with 4×4 switch blocks realizable on FPGAs with a sufficient amount of exploitation of manufacturing differences to be used in real-world applications?

First of all, our 4×4 APUF design extracts some amount of manufacturing differences. This means our design can be useful for some real-world applications.

At first look, our design seems to have low uniqueness. Nevertheless, it should be kept in mind that our results are for responses of small length (4.58 bits). The overall uniqueness can be improved by combining multiple responses to create a larger one. Though, while doing so, non-ideal reliability should be taken into account.

6.1 Our 4×4 APUF in the Real World

We are proposing several methods to make better use of our 4×4 APUF and applications to which these methods can be applied.

In all of these methods multiple responses are "combined" to produce a larger one. We are defining this combination as such: Let R be the combination of $r_1, \ldots r_m$, individual responses taken from the APUF.

 $^{^{6}}$ Keeping the number of challenge bits just above 128 would be wrong because challenge bits are redundant: 5-bit challenge of a switch block can only take values in the interval [0, 23], rather than [0, 31].

$$R = 24^0 r_1 + \dots + 24^{m-1} r_m$$

We are performing the combination this way to prevent redundancy in the combined response.

Combining Responses of Randomly Selected Challenges This is the most straightforward method. According to our results, the probability of a randomly selected challenge to produce different results on 2 different chips is 15.1520%. Let's call it *p*. Accordingly, the probability, *u*, of 2 different combined responses created from 2 different chips to be different can be expressed as follows.

$$u = 1 - (1 - p)^m \tag{1}$$

where m is the number of responses that are combined.

Let's generalize the case and call the same probability for n different chips P(n), where P(2) = u. If we assume that number of PUFs is negligible compared to the possible number of combined responses $(n \ll 24^m)$, P(n) can be expressed as follows.

$$P(n) = u^{\binom{n}{2}} = u^{\frac{n(n-1)}{2}}, \quad n \ge 2$$
⁽²⁾

If we combine equations 1 and 2, we can express P as a function of both n and m.

$$P(n,m) = \left[1 - (1-p)^m\right]^{\frac{n(n-1)}{2}}$$
(3)

A contour diagram of P is shown in Fig. 15. The sufficient number of response length for the required number of PUFs can be seen on the diagram. For example, if we want to uniquely identify 10000 PUFs with 99% probability, we need a 600-bit response.



Figure 15: Contour diagram for P (uniqueness probabilities).

The problem with this method is that unreliability of responses is not handled. If the same combined response is tried to be generated using the same challenges, the results will most probably differ, especially for higher m values. However, this unpredictability can be exploited by applications where True Random Number Generators (TRNGs) are necessary.

Detection of Reliable Challenges Beforehand This method involves using only reliable challenges by detecting them beforehand. Detection can be performed either after fabrication before distribution or while runtime. This way, it will be possible to reproduce the same responses. This enables identification and authentication applications.

Equation 3 can also be used for this method with, this time, p = 9.3261%. Need for larger responses (higher m) and extra cost for reliable challenge detection are traded with repeatability.

Better uniqueness would improve usage costs of our 4×4 APUF design by reducing the number of required responses to combine. However, there is a possibility that higher uniqueness can also cause lower reliability because when PUF delays are closer to each other, the results will be more sensitive to factors like noise.

7 Future Work

First of all, the results of our 4×4 APUF do not have desirable uniqueness. So, the hardware design should be improved to achieve higher uniqueness. This may be achieved by focusing on symmetrical placement and routing of switch blocks and the arbiter. Once an acceptable uniqueness is achieved, the security of the 4×4 APUF should be evaluated. Evaluating its resistance to ML modeling attacks has the priority since APUFs are generally vulnerable to these attacks. Resistance to side-channel analysis can also be evaluated.

Real-world applications require at least 128-bit keys. Accordingly, APUF responses should be securely transformed into desired length. Also statistical analysis of this transformation should be performed.

In this thesis, we provided an area comparison of our 4×4 APUF to other APUFs in the literature. This comparison can be extended. After security analysis, the size of different APUFs (like the number of stages) can be selected to provide a similar amount of security, then areas can be compared. This way, area requirement can be related to security every APUF provides. Furthermore, the maximum response generation throughput should be determined experimentally and play a role in the comparison.

8 Conclusion

In this thesis, we implemented 4×4 APUF proposed in [8] on FPGAs and performed statistical analysis to evaluate its capabilities to be used in real-world applications.

According to the analysis, the capabilities of our design are adequate to enable many real-world applications, such as identification, authentication, encryption, and random number generation. However, since uniqueness is lower than desirable, these applications are associated with runtime performance penalty.

Our design can be improved by focusing on making switch block delays closer to each other, which may increase uniqueness. Other possible future work includes performing security evaluation of 4×4 APUF, most importantly evaluating its resistance to ML modeling attacks, and finding better methods that enable the usage of 4×4 APUF with lower operation costs.

Appendices

A Hardware Schematics for 4×4 **APUF**

Figure 16: Vivado schematics for 4×4 APUF.



Figure 17: Vivado schematics for 4×4 APUF (Front closeup).



Figure 18: Vivado schematics for 4×4 APUF (Back closeup).



Figure 19: Vivado schematic for 4×4 APUF permutation component.



Figure 20: Vivado schematics for 4×4 APUF switch block.



Figure 21: Vivado schematics for 4×4 APUF switch block multiplexer.



Figure 22: Vivado schematics for 4×4 APUF arbiter.

B 4×4 **APUF** Net Delays

The output of the Tcl script written to get net delays in Vivado. Using default placement.

Manually replaced repetitive parts of the net names with three dots "...".

source ~/Downloads/get_net_delays.tcl
puts "NAME FAST_MAX FAST_MIN SLOW_MAX SLOW_MIN" # puts "NAME FAST_MAK FAST_MIN SLOW_MAN SLOW_MIN" NAME FAST_MAK FAST_MIN SLOW_MAX SLOW_MIN # proc put_delays [net) { # foreach net_delay [get_net_delays -of_objects [get_nets %net]] { # set fast_max [get_property FAST_MAX %net_delay] # set fast_max [get_property FAST_MIN %net_delay] # set slow_max [get_property SLOW_MAX %net_delay] # set slow_max [get_property SLOW_MAX %net_delay] # set slow_min [get_property SLOW_MIN %net_delay] # set slow_min [get_property SLOW_MIN %net_delay] puts "\$net_delay \$fast_max \$fast_min \$slow_max \$slow_min" # proc switch_block_delays {net_base} {
for {set idx 0} {\$idx < 4} {incr idx} {
put_delays \$net_base[\$idx]</pre> # switch block delays apuf driver/APUF TEST UNIT/FIRST SWITCH BLOCK/input # switch_block_delays apuf_driver/APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/input ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[1]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[2]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[3]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[3]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[5]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[6]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...apuf_response_mutual_order_reg_reg[6]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[0]_to_...FBM_onehot_state_reg[3]/D 611 524 1123 944 FIRST_SWITCH_BLOCK/input[0]_to_...FSM_onehot_state[3]/D 611 524 1123 944 ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ ISS_ONENC_STATE[0]_1/10 277 234 555 464 Lusy_INST_0/IO 277 234 555 464 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/IO 716 610 1304 1089 ...FIRST_SWITCH_BLOCK/input[0]_to .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/IO 712 606 1303 1087 ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ APUT_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/IO 961 822 1767 1471 APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/IO 1038 899 1832 1547 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/IO 1688 594 1252 1054 ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ AUU_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_UUX/Q_INST_0/15 1038 899 1832 1547 AUU_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_UUX/Q_INST_0/15 1038 899 1832 1547 AUU_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_UUX/Q_INST_0/15 807 761 1492 1273 AUU_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_UUX/Q_INST_0/15 807 761 1492 1273 ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ ...FIRST_SWITCH_BLOCK/input[0]_to_ APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/12 961 837 1685 1429 APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/12 961 837 1685 1429 APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/12 961 837 1685 1429 APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 961 837 1685 1429 ...FIRST_SWITCH_BLOCK/input[0]_to_. ...FIRST_SWITCH_BLOCK/input[0]_to_. ...FIRST_SWITCH_BLOCK/input[0]_to_. ...FIRST_SWITCH_BLOCK/input[0]_to_. .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/11 419 365 713 607 ...FIRST_SWITCH_BLOCK/input[0]_to_...APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/11 419 365 713 607 ...FIRST_SWITCH_BLOCK/input[0]_to_...APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/11 620 524 1130 940 ...FIRST_SWITCH_BLOCK/input[0]_to_...APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/11 410 356 704 598 ...FIRST_SWITCH_BLOCK/input[1]_to_...apuf_response_mutual_order_reg_reg[1]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[1]_to_...apuf_response_mutual_order_reg_reg[3]/CE 524 449 1016 850 ...FIRST_SWITCH_BLOCK/input[1]_to_...Apur_response_mutual_order_reg_reg[1/CE 524 449 1016 680 ...FIRST_SWITCH_BLOCK/input[1]_to_...Apuf_response_mutual_order_reg_reg[5]/CE 524 449 1016 680 ...FIRST_SWITCH_BLOCK/input[1]_to_...FSM_onehot_state_reg[3]/C 611 524 1123 944 ...FIRST_SWITCH_BLOCK/input[1]_to_...FSM_onehot_state[0]_i_1/10 277 234 555 464 ...FIRST_SWITCH_BLOCK/input[1]_to_...APUF_ITST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 880 774 1506 1287 ...FIRST_SWITCH_BLOCK/input[1]_to_...APUF_ITST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 880 774 1506 1287 ...FIRST_SWITCH_BLOCK/input[1]_to_...APUF_ITST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_0/I1 4936 713 607 FIRST_SWITCH_BLOCK/input[1]_to_...APUF_ITST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_0/I1 4936 713 607 FIRST_SWITCH_BLOCK/input[1]_to_...APUF_ITST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_0/I1 4936 713 607 .APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/11 419 365 713 607 APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/11 620 524 1130 940 .APUF_TESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/11 410 356 704 598 .APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/15 1038 899 1822 1647 .APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/15 1038 899 1822 1647 .APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/15 1038 899 1822 1647 .APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/15 1038 899 1602 1647 .APUF_IESI_UNIT/FIRSI_SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/15 107 665 1407 1185 ...FIRST SWITCH BLOCK/input[1] to ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ ARUT_IEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I5 867 761 1492 1273 APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 963 839 1688 1432 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I2 961 837 1685 1429 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/12 962 838 1667 1431 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 961 837 1685 1429 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/10 716 610 1304 1089 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/10 712 606 1303 1087 ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ ...FIRST_SWITCH_BLOCK/input[1]_to_ .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATH5[1].EQUAL_PATH_MUX/Q_INST_0/10 712 606 1303 1087 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATH5[2].EQUAL_PATH_MUX/Q_INST_0/10 961 822 1767 1471 .APUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATH5[3].EQUAL_PATH_MUX/Q_INST_0/10 1038 899 1832 1547 .apuf_response_mutual_order_reg_reg[1]/CE 524 449 1016 850 .apuf_response_mutual_order_reg_reg[3]/CE 524 449 1016 850 .apuf_response_mutual_order_reg_reg[5]/CE 524 449 1016 850 .apuf_response_mutual_0rder_reg_reg[5]/CE 524 449 1016 850 .apuf_response_mutual_0rder_reg_reg[5]/CE 524 449 1016 850 .FIRST_SWITCH_BLOCK/input[1]_to_ .FIRST_SWITCH_BLOCK/input[1]_to_ .FIRST_SWITCH_BLOCK/input[2]_to_ ...FIRST_SWITCH_BLOCK/input[2]_to_ ...FIRST_SWITCH_BLOCK/input[2]_to_. ...FIRST_SWITCH_BLOCK/input[2]_to_. ...FIRST_SWITCH_BLOCK/input[2]_to_. ...FIRST_SWITCH_BLOCK/input[2]_to_. ...FIRST_SWITCH_BLOCK/input[2]_to_.. ...FIRST_SWITCH_BLOCK/input[2]_to_.. ...FIRST_SWITCH_BLOCK/input[2]_to_.. FSM_onehot_state_reg[3]/D 611 524 1123 944 .FSM_onehot_state[0]_i_1/10 277 234 555 464 .busy_INST_0/I0 277 234 555 464 ...FIRSI_SWITCH_BLOCK/INput[2]_to_...APUF_TEST_UNIT/FIRSI_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INSI_0/I1 880 774 1506 1287 ...FIRSI_SWITCH_BLOCK/INput[2]_to_...APUF_TEST_UNIT/FIRSI_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INSI_0/I1 4936 713 607 ...FIRSI_SWITCH_BLOCK/INput[2]_to_...APUF_TEST_UNIT/FIRSI_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INSI_0/I1 420 524 130 940 ...FIRSI_SWITCH_BLOCK/INput[2]_to_...APUF_TEST_UNIT/FIRSI_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INSI_0/I1 420 567 745 98

FIRST_SWITCH_BLOCK/input[2]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/IO 716 610 1304 1089 FIRST_SWITCH_BLOCK/input[2] toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EDUAL_PATH_MUX/Q_INST_0/IO 712 606 1303 1087
FIRST SWITCH BLOCK/input[2] to APUF TEST UNIT/FIRST SWITCH BLOCK/EQUAL PATHS[1].EQUAL PATH MUX/Q INST 0/10 712 606 1303 1087
FIRST SWITCH BLOCK (input [2] to ADDE TEST INIT/FIRST SWITCH BLOCK (FOUND DATHS[2] FOUND DATH MUY (O INST 0/10 061 822 1757 1471
ETBET SUITCH BLOCK (input [2] to ADDE TEST INIT/ FIRST SUITCH BLOCK/EQUAL ADDE ADDE ADDE ADDE ADDE ADDE ADDE AD
ETDET GUTGU DLOW / INPUT [2] to AND INTERNATION DUTCH DLOW / FOUND DATE (DLOW / FOUND DATE (DLOW / LOW / COUND DATE (DLOW / LOW / COUND DATE (DLOW / LOW
rikal_switch_block/input[2]_tokrof_ibal_wrif/rikal_switch_block/coukt_rkinc[0].Edukt_rkin=m0/(d_ikal_vki_cout) 0.000/(d_ikal_vki_cout) 0.000/
rikal_Switch_BLOCK/input[2]_toRFOF_IS3_UNIT/FIRST_SWITCH_BLOCK/EQUAL_FAID[1].E
FIRST_SWITCH_BLUCK/INput[2]_toAPUF_IEST_UNIT/FIRST_SWITCH_BLUCK/EQUAL_PAIHS[2]_EQUAL_PAIH_MUX/Q_INST_0/12 962 838 1687 1431
FIRST_SWITCH_BLOCK/input[2]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 961 837 1685 1429
FIRST_SWITCH_BLOCK/input[2]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I5 689 594 1252 1054
FIRST_SWITCH_BLOCK/input[2]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I5 1038 899 1832 1547
FIRST_SWITCH_BLOCK/input[2]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/15 770 665 1407 1185
FIRST_SWITCH_BLOCK/input[2]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/15 867 761 1492 1273
FIRST_SWITCH_BLOCK/input[3]_toapuf_response_mutual_order_reg_reg[1]/CE_524_449_1016_850
FIRST SWITCH BLOCK/input[3] toapuf response mutual order reg reg[2]/CE 524 449 1016 850
FIRST SWITCH BLOCK/input[3] to apuf response mutual order reg reg[3]/CF 524 449 1016 850
FIRST SWITCH BLOCK/input[3] to april response mutual order reg reg[4]/CF 524 449 1016 850
The Strong block input [2] to applie second strong and a second strong strong strong s
rinsi_switch_bLock/input[5]_t0apui_response_mutuat_order_reg_reg[5]/CE 524 449 1010 650
FIRST_SWITCH_BLUCK/input[3]_toapuf_response_mutual_order_reg_reg[6]/CE_524_449_1016_850
FIRST_SWITCH_BLUCK/input[3]_toFSM_onehot_state_reg[3]/D 611 524 1123 944
FIRST_SWITCH_BLOCK/input[3]_toFSM_onehot_state[0]_i_1/I0 277 234 555 464
FIRST_SWITCH_BLOCK/input[3]_tobusy_INST_0/I0 277 234 555 464
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 880 774 1506 1287
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 419 365 713 607
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I1 620 524 1130 940
FIRST SWITCH BLOCK/input[3] toAPUE TEST UNIT/FIRST SWITCH BLOCK/EQUAL PATHS[3].EQUAL PATH MUX/Q INST 0/11 410 356 704 598
FIRST SWITCH BLOCK/input [3] to APUE TEST INIT/FIRST SWITCH BLOCK/FOULDL PATHS[0] FOULDL PATH MUX/0 INST 0/10 716 610 1304 1089
FIRST SUTTOR BLOCK / INDUST 1 + ADIE TEST INIT/FIRST SUTTOR BLOCK/FOILAL DATHS[1] FOILAL DATH MILY (I INST 0/10 T12 FIG 1303 1087
FIRST SUITCH BLOCK / imput[3] to _ ADIE TEST INIT/FIRST SUITCH_DEGON/ EQUAL: A ADIE 1 - EQUAL: A ADIE - COM - 1000 - 1007 - 1000
The second secon
rinoi_writor_buows/input[0]_toArvr_ito]_writ/rino_bu/k/24U4k_YAIHS[3].EU4AL_FAIH_MUX/LINS_0/10 1038 899 1832 154/
FIRST_SWITCH_BLUCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/15 689 594 1252 1054
FIRST_SWITCH_BLUCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I5 1038 899 1832 1547
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 770 665 1407 1185
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 867 761 1492 1273
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/12 963 839 1688 1432
FIRST_SWITCH_BLOCK/input[3]_toAPUF_TEST_UNIT/FIRST_SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/12 961 837 1685 1429
FIRST SWITCH BLOCK/input[3] to APUF TEST UNIT/FIRST SWITCH BLOCK/EQUAL PATHS[2]. EQUAL PATH MUX/Q INST 0/12 962 838 1687 1431
FIRST SWITCH RLOCK/input [3] to APUE TEST INIT/FIRST SWITCH RLOCK/FOULDL PATHS [3] FOULDL PATH MUX/O INST 0/12 961 837 1685 1429
suitch block_dalaus anuf driver/ADHE TEST INIT/FIRST SUITCH BLOCK/output
FIGE SUTTON BLOCK (AUTON) 10 - SUTTON BLOCKS[1] SUTTON BLOCK (BUILD ATHS[0] FOLD AT
THE STORE DIOK OUT OF THE STORE DIOK STORE DIOK STORE DIOK STORE DIOK STORE DIOK STORE STORE DIOK S
TIRSTSWICE BLOCK/ULPUC (0]_00SWICE BLOCK/EULPAINS[], EQUAL_FAINS[], EQUAL_FAINS[], ULVOL_(11, 301, 203, 302, 322, 322, 322, 322, 322, 322
FIRSI_SWITCH_BLUCK/OUTPUT[0]_toSWITCH_BLUCKS[1].SWITCH_BLUCK/EUVAL_FAIHS[2].EUVAL_FAIH_MUX/Q_INSI_0/II 412 343 /88 649
FIRST_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/II 417 312 799 554
FIRST_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I0 264 227 500 422
FIRST_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I0 237 201 449 376
FIRST_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I0 344 284 680 557
FIRST_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I0 351 253 694 463
FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I5 408 340 785 646
FIRST SWITCH BLOCK/output[2] toSWITCH BLOCKS[1].SWITCH BLOCK/EQUAL PATHS[1].EQUAL PATH MUX/Q INST 0/15 408 340 787 648
FIRST SWITCH BLOCK/output[2] to SWITCH BLOCKS[1], SWITCH BLOCK/EQUAL PATHS[2], EQUAL PATH MUX/Q INST 0/15 306 258 600 498
FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 258 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_MUX/Q_INST_0/I5 306 258 600 498
FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 258 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310
FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 258 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 438 361 863 705
FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 528 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 437 637 527 FIRST_SWITCH_BLOCK/OUtput[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0]_HOUL_PATH_MUX/Q_INST_0/I2 437 637 527 FIRST_SWITCH_BLOCK/OUtput[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATH_MUX/Q_INST_0/I2 437 637 527
FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 258 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 449
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_MUX/Q_INST_0/I5 306 288 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 328 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I2 289 215 674 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 258 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 549 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 549 FIRST_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 259 215 547 549 FIRST_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358 # for {set sb 1} {\$sb <= 23} {inr sb} {</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 306 288 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 328 361 563 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 327 363 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 289 215 47 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358 # for [set sb 1] {\$sb <= 23} {lincr sb} { switch_block_delays apuf_ctiver/APUF_IEST_UNIT/SWITCH_BLOCKS[\$sb].SWITCH_BLOCK/output</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_WUX/Q_INST_0/I5 306 258 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_WUX/Q_INST_0/I2 259 215 547 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358 # for {set sb 1} {\$sb <= 23} {incr sb } { # switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK[\$sb].SWITCH_BLOCK/output # }</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_WUX/Q_INST_0/I5 306 228 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 437 861 763 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 259 215 674 499 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 249 195 464 358 # for {set sb }} {set ch } {s</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHE[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 258 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 328 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_0/I2 327 363 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_0/I2 327 363 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_WUX/Q_INST_0/I2 243 195 464 358 # for [set sb 1] {\$sb << 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[\$sb].SWITCH_BLOCK/output * } SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/eQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 572</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 363 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 363 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 249 195 464 358 # for [set 31] {\$8b < 23} {inc sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358 switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/S[\$b].SWITCH_BLOCK/output syitch_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCK/S[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].BQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].BQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTPUT[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUA</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHE[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 258 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I5 306 128 301 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 327 363 757 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 327 363 757 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 259 215 644 358 # for [set sb 1] {\$sb <= 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[\$sb].SWITCH_BLOCK/output #SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 362 304 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SW</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_WUX/Q_INST_0/I2 327 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_WUX/Q_INST_0/I2 327 373 577 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 375 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 259 215 474 49 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 249 195 464 358 # for [sat s1] {8sb <23} {inc sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWI</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 328 361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 259 215 674 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 249 215 464 368 # for [set sb l] {\$bb <= 23} {lincr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWI</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358 # for [set sb 1] {8bc <23} {inc sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[\$sb].SWITCH_BLOCK/output * } SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS/1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS/1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS/1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 328 361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 269 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 243 195 464 368 # for {set sb 1} {\$bc < 23} {incr sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_0/I1 363 304 695 77 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I1 363 304 695 77 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/I2 243 195 464 358 # for [set sb 1] {8bc <23} {inc sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS/I].SWITCH_BLOCK/OUTPUT[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS/I].SWITCH_BLOCK/OUTPUT[</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I5 306 328 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 328 361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 259 215 646 358 # for {set sb }} {\$sb < 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS/UIL_PATH_MUX/Q_INST</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 122 373 10 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 328 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 259 215 674 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 368 # for [set sb l] {\$sb <= 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[\$sb].SWITCH_BLOCK/cUUL_PATHS[0].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 362 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 362 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 362 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_WUX/Q_INST_O/I1 363 304 694 572 SWITC</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 228 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I5 306 123 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 329 215 636 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 329 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I2 243 195 464 368 # for [set b] 1 {\$bb < 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS[1]_SWITCH_BLOCK/outp</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 725 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 289 215 644 368 # for [set sb 1] {\$bcSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 368 # for [set sb 1] {\$bc + 23} {lincr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/I1 365 306 695 562 SWITCH_BLOCKS[1].SWITCH_BLOCK/outpu</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 437 8361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 368 # for [set b] 1 {\$bb < 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/o</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 287 2173 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 289 215 464 368 # for [set sb l] {sb c = 23} {incr sb {</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I5 306 126 370 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 259 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 358 # for [set 3b] {sbc <23} {inc sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 692 577 SWITCH_BLOCKS/UNTCH_BLOCK/OUTPUI</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 205 172 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 438 361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 249 215 644 368 # for [set sb 1] {\$bc <= 23} {Incr sb} {</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 437 367 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 249 215 464 358 # for [set s1] {sbc <23} {inc sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 363 306 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 363 306 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0]_DUAL_PATH_MUX/Q_INST_O/I1 363 306 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/O</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 437 83 61 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 287 215 647 449 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 358 # for [set sb 1] {\$bc < 23} {Incr sb } {</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 358 # for [set s1] {Sbt <23} {inc sb { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[\$sb]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 437 267 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 249 125 464 358 # for {set ab }} {\$\$b < = 23} {incr ab} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/outpu</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I5 306 7267 537 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 249 215 644 358 # for [set s1] {Sbt <23} {inter_sbt <23} {inter_sbt <452 3} {inter_sbt <23} {inter_sbt <24} {inter_sbt <23} {inter_sbt <24} {inter_sbt <23} {inter_sbt <24} {inter_</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 122 373 10 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 289 215 464 368 # for [set sb 1] {\$bc < 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 437 37 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 437 367 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 434 195 464 358 # for [set st] { \$ sbitch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[\$ sb].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 362 303 692 572 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS/II.SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS/II.SWITCH_BLOCK/OUTPUT[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_HUX/Q_INST_O/I1 363 304 694 573 SWITCH_BLOCKS/II.SWITCH_BLOCK/OUTPUT[</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I5 306 123 373 310 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 327 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 289 215 646 358 # for (set sb 1) {\$sb < 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_O/I2 243 195 464 358 # for (set sb 1) {\$sb < 23} {incr sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I5 306 2268 600 498 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_WUX/Q_INST_O/I2 438 361 863 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[0]_EQUAL_PATH_WUX/Q_INST_O/I2 437 37 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 427 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_WUX/Q_INST_O/I2 434 195 464 358 # for {set sb }] {\$bc < 23} {inc sb }</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_WUX/Q_INST_0/I5 306 228 600 488 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 438 361 663 705 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 237 273 637 527 FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 249 215 644 368 # for (set sol) {\$sbc < 23} {incSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I2 243 195 464 368 # for (set sol) {\$sbc < 23} {incSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_WUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_WUX/Q_INST_0/I1 362 303 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 306 692 577 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 694 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 363 304 692 573 SWITCH_BLOCKS[1].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2].SWITC</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/IS 06 258 600 498FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I2 438 361 863 705FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I2 37 273 637 527FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_MUX/Q_INST_O/I2 243 195 464 358 # for {set sh 1} {set sh 2} for {set sh 1} {set sh 2} set sh 1 {set sh 4} * switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTput[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 304 694 573SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PAT</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/15 306 258 600 498FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/12 438 361 863 705FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/12 438 361 863 705FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/12 249 215 547 449FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/12 249 136 464 388 # for {star bl } {starch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[sb]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/11 362 303 692 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_O/11 362 303 692 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_O/11 362 303 692 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_O/11 362 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_O/11 363 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/11 363 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTput[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/11 363 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTput[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/11 363 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTput[3]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/11 363 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/11 363 304 694 57SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTPUT</pre>
<pre>FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/IS 06 258 600 488FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I2 438 361 863 705FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I2 37 273 637 527FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[2]_EQUAL_PATH_MUX/Q_INST_O/I2 289 215 547 449FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I2 243 195 464 358 # for {set sh 1} {set s2} {inter_sb} { switch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUtput[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTput[1]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 303 692 572SWITCH_BLOCKS[1]_SWITCH_BLOCK/OUTPUT[3]_toSWITCH_BLOCKS[2]_SWITCH_BLOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_O/I1 362 304 694 572</pre>
<pre>FIRST_SWITCH_BLOCK/output[2].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/IS 306 258 600 498FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/IZ 307 273 310FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/IZ 307 273 637 567FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_O/IZ 307 273 637 567FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_O/IZ 307 273 637 567FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/IZ 303 105 647 449FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 362 303 692 67SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[0].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/II 362 303 692 67SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[0].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[0].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[0].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[1].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[1].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[1].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[3].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 57SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTPUT[1].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].</pre>
<pre>FIRST_SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/CQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/IS 306 258 600 498FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/CQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/IS 306 258 600 498FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/IS 307 637 637 527FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_O/IS 237 273 637 637 527FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_O/IS 249 215 647 449FIRST_SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_O/II 243 195 464 386 # or fact ab 1} { sb < 233 { lor cr b} { lor cr b} { lor cr b} { lor cr b} { switch_BLOCKS[1].SWITCH_BLOCKS[2].SWITCH_BLOCK/CUUL_PATHS[0].EQUAL_PATH_MUX/Q_INST_O/II 362 303 692 577 .SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/CUUL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 362 303 692 577 .SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/CUUL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 362 303 692 577 .SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTput[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/CUUL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 362 303 692 577 .SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 577 .SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTPUT[0]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 577SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTPUT[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 577SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTPUT[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 577SWITCH_BLOCKS[1].SWITCH_BLOCK/OUTPUT[1]_toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_O/II 363 304 694 577SWITCH_BLOCKS[</pre>
<pre>FiRST_SWITCH_BLOCK/output[3]_to</pre>
<pre>FIRST_SWITCH_BLOCK/output[2].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/15 206 172 373 310FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/15 206 172 373 310FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 237 273 637 527FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 237 273 637 527FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 237 218 547 449FIRST_SWITCH_BLOCK/output[3].toSWITCH_BLOCKS[1].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 243 195 464 358 # for {set sol } { soitch_block_delays apuf_driver/APUF_TEST_UNIT/SWITCH_BLOCK/S[sb].SWITCH_BLOCK/OUTput # }SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCKS[2].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 577SWITCH_BLOCKS[1].SWITCH_BLOCK/output[0].toSWITCH_BLOCK/EQUAL_PATHS[2].EQ</pre>
<pre>FIRST_SWITCH_BLOCK/output[3]_to</pre>
<pre>FiRST_SWITCH_ELOCK/output[2]_coSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/15 306 288 600 488FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/12 438 361 963 705FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/12 259 215 547 449FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/12 259 215 547 449FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/12 243 195 464 358 # for (set ab 1] {\$bb < 23} {incr sb} { switch_ElocKs[1].SWITCH_ELOCK/ID_LOCK_S[2].SWITCH_ELOCK/CQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 303 692 577SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 303 692 577SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 303 692 577SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 363 304 694 573SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 363 304 694 573SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 363 304 694 573SWITCH_ELOCKS[1].SWITCH_ELOCK/output[1]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 303 3692 577SWITCH_ELOCKS[1].SWITCH_ELOCK/output[1]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 308 303SWITCH_ELOCKS[1].SWITCH_ELOCK/output[1]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 308 309SWITCH_ELOCKS[3].SWITCH_ELOCK/OUTPUT[3]_toSWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_NUX/Q_INST_0/11 362 308 309SWITCH_ELOCKS[3].SWITCH_ELOCK/OUTPUT[3]_toSWITC</pre>
<pre>FIRST_SWITCH_ELOCK/output[2]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUL_PATHS[3].EQUA_PATH_MUX/Q_INST_0/15 306 288 600 498FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUA_PATH_MUX/Q_INST_0/15 287 273 637 567FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[1].EQUA_PATH_MUX/Q_INST_0/12 289 215 547 449FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 289 215 547 449FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[1].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 289 215 547 449FIRST_SWITCH_ELOCK/output[3]_toSWITCH_ELOCKS[2].SWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 289 215 547 449SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/13 280 3869 547SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 477SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCK/EQUAL_PATHS[0]_EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 477SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_0/11 362 303 692 477SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCK/EQUAL_PATHS[1]_EQUAL_PATH_MUX/Q_INST_0/11 362 304 694 573SWITCH_ELOCKS[1].SWITCH_ELOCK/output[0]_toSWITCH_ELOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/11 353 304 694 573SWITCH_ELOCKS[1].SWITCH_ELOCK/output[1]_toSWITCH_ELOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/11 353 466 453SWITCH_ELOCKS[1].SWITCH_ELOCK/output[1]_toSWITCH_ELOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/11 353 466 456SWITCH_ELOCKS[1].SWITCH_ELOCK/OUTPUT[1]_toSWITCH_ELOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/11 282 204 44SWITCH_ELOCKS[1].SWITCH_ELOCK/OUTPUT[1]_toSWITCH_ELOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/11 282 204 44SWITCH_ELOCKS[1].SWITCH_ELOCK/OUTPUT[1]_toSWITCH_ELOCK/EQUAL_PATHS[3]_EQUAL_PATH_MUX/Q_INST_0/11 282 204 44SWITCH_EL</pre>

SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS [4]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/I1	273	229 E	549 453
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[4]	.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/I1	277	204 5	557 381
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[4]	.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/10	440	363 8	354 698
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[4]	.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/I0	439	362 8	351 696
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[1]_to		.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/I0	444	366 8	358 701
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[4]	.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/I0	236	194 4	137 358
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[2]_to		.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/15	119	102 2	233 197
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[4]	.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/I5	364	309 6	376 564
SWITCH_BLOCKS[3]	.SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS [4]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/15	278	240 4	199 422
SWITCH BLOCKS[3]	SWITCH BLOCK/output[2] to	SWITCH BLOCKS [4]	SWITCH BLOCK/EQUAL PATHS[3]	EQUAL PATH MUX/Q INST 0/15	280	231 F	503 395
SWITCH BLOCKS[3]	SWITCH BLOCK/output[3] to	SWITCH BLOCKS [4]	SWITCH BLOCK/FOULAL PATHS[0]	FOILAL PATH MIX /O INST 0/12	413	344 8	818 673
SWITCH BLOCKS [3]	SWITCH BLOCK (output [3] to	SWITCH BLOCKS [4]	SWITCH BLOCK/FOUNT DATHS[1]	FOUNT DATH MUY/O INST 0/12	240	202 /	170 307
CUITCH BLOCKS [2]	SWITCH BLOCK/output[2] to	CUITCU DI OCVC [4]	SWITCH BLOCK/EQUAL_IATHS[1]	FOUNT DATE MUX/0 INST 0/12	240	202 -	100 106
CUITOU DLOCKS[3]	GUITOU DI OCK/output [3] +-	GUITON DLOCKS [4]	CUITCH_BLOCK/EQUAL_FAINS[2]	EQUAL_FAIN_MUX/Q_INSI_0/12	242	203 4	190 400
SWITCH_BLUCKS[3]	.SWITCH_BLUCK/output[3]_to	SWIICH_BLUCKS[4]	SWITCH_BLUCK/EQUAL_PATHS[3]	EQUAL_PAIN_MUX/Q_INSI_0/12	350	200 0	300 409
SWITCH_BLUCKS[4]	SWITCH_BLUCK/output[0]_to		SWITCH_BLUCK/EQUAL_PAINS[U]	EQUAL_PAIN_MUX/Q_INSI_0/II	401	213 1	20 591
SWIICH_BLUCKS[4]	.SWIICH_BLUCK/output[0]_to	SWIICH_BLUCKS[5]	.SWIICH_BLUCK/EQUAL_PAIHS[1]	EQUAL_PATH_MUX/Q_INSI_0/11	421	353 8	312 669
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[2]	.EQUAL_PATH_MUX/Q_INST_0/I1	460	381 8	365 707
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[3]	.EQUAL_PATH_MUX/Q_INST_0/I1	467	350 8	379 613
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[0]	.EQUAL_PATH_MUX/Q_INST_0/I0	225	190 4	166 390
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[1]	.EQUAL_PATH_MUX/Q_INST_0/I0	318	266 6	67 551
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[2]	.EQUAL_PATH_MUX/Q_INST_0/I0	387	325 7	782 648
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[3]	.EQUAL_PATH_MUX/Q_INST_0/I0	391	300 7	790 576
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/15	160	140 2	273 232
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[2]_to		.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/15	158	138 2	269 229
SWITCH_BLOCKS[4]	.SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[5]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/I5	158	138 2	269 229
SWITCH BLOCKS [4]	.SWITCH BLOCK/output[2] to .	.SWITCH BLOCKS [5]	.SWITCH BLOCK/EQUAL PATHS[3]	EQUAL PATH MUX/Q INST 0/15	270	206 5	519 363
SWITCH BLOCKS [4]	SWITCH BLOCK/output[3] to	SWITCH BLOCKS [5]	SWITCH BLOCK/EQUAL PATHS[0]	EQUAL PATH MUX/0 INST 0/12	377	313 7	707 581
SWITCH BLOCKS [4]	.SWITCH BLOCK/output[3] to	.SWITCH BLOCKS [5]	.SWITCH BLOCK/EDUAL PATHS[1]	EQUAL PATH MUX/0 INST 0/12	370	308 6	398 573
SWITCH BI OCKO [4]	SWITCH BLOCK/output[3] +o	SWITCH BIOCKS [5]	SWITCH BLOCK/FOUNT DATHS[1]	FOILAL PATH MILY /O TNST 0/12	366	305 4	589 566
CUITCH BLOCKS [4]	SWITCH BLOCK/output[2] to	CUITCU DI OCVO [6]	SWITCH BLOCK/EQUAL_IATHS[2]	FOUNT DATE MUX/0 INST 0/12	271	074 7	700 471
CUITOU DI OCKO [4]	GUITOU DI OGK (autout [0] to	QUITON_BLOCKS[0]	GUITOU DLOCK/EQUAL_FAINS[3]	EQUAL_FAIN_MUX/Q_INSI_0/12	571	402 0	200 4/1
SWITCH_BLUCKS[5]	.SWITCH_BLUCK/output[0]_t0	SWIICH_BLUCKS[0]	SWITCH_BLUCK/EQUAL_PATHS[U]	EQUAL_PAIN_MUX/Q_INSI_0/II	505	423 8	330 101
SWIICH_BLUCKS[5]	.SWIICH_BLUCK/output[0]_to	SWIICH_BLUCKS[6]	.SWIICH_BLUCK/EQUAL_PAIHS[1]	EQUAL_PATH_MUX/Q_INSI_0/II	433	361 8	311 666
SWIICH_BLUCKS[5]	.SWIICH_BLUCK/output[0]_to	SWIICH_BLUCKS[6]	.SWIICH_BLUCK/EQUAL_PAIHS[2]	EQUAL_PATH_MUX/Q_INSI_0/11	407	337 1	/45 593
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/I1	334	274 6	302 472
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[0]	.EQUAL_PATH_MUX/Q_INST_0/10	242	208 4	117 352
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[1]	.EQUAL_PATH_MUX/Q_INST_0/I0	282	242 4	186 410
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[2]	.EQUAL_PATH_MUX/Q_INST_0/I0	579	449 1	1061 77
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[3]	.EQUAL_PATH_MUX/Q_INST_0/I0	246	206 4	125 353
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/15	498	417 9	919 758
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[2]_to		.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/15	323	272 6	316 510
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/I5	222	186 3	389 322
SWITCH_BLOCKS[5]	.SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[6]	.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/15	503	386 9	330 663
SWITCH BLOCKS [5]	.SWITCH BLOCK/output[3] to .	.SWITCH BLOCKS [6]	.SWITCH BLOCK/EQUAL PATHS[0]	EQUAL PATH MUX/Q INST 0/12	466	399 8	389 745
SWITCH BLOCKS[5]	SWITCH BLOCK/output[3] to	SWITCH BLOCKS [6]	SWITCH BLOCK/EQUAL PATHS[1]	EQUAL PATH MUX/0 INST 0/12	426	355 8	301 658
SWITCH BLOCKS[5]	SWITCH BLOCK/output[3] to	SWITCH BLOCKS [6]	SWITCH BLOCK/FOULAL PATHS[2]	FOULAL PATH MUX/O INST 0/12	356	271 6	681 483
SWITCH BLOCKS[5]	SWITCH BLOCK (output [3] to	SWITCH BLOCKS[6]	SWITCH BLOCK/FOUNT DATHS[3]	FOUNT DATH MUY/O INST 0/12	473	383 0	201 600
CUITCH BLOCKS [0]	SWITCH BLOCK (output [0] to	CUITCU DI OCVC [7]	SWITCH BLOCK/EQUAL_IATHS[0]	FOUNT DATE MUX/0 INST 0/11	106	2000 0	017 672
SWITCH_BLUCKS[0]	GUITOU DI OGK (autout [0] to	QUITON_BLOCKS[7]	CUITCH_BLOCK/EQUAL_FAINS[0]	EQUAL_FAIN_MUX/Q_INSI_0/II	420	200 0	317 073
SWITCH_BLUCKS[6]	.SWITCH_BLUCK/output[0]_t0	SWIICH_BLUCKS[7]	SWITCH_BLUCK/EQUAL_PATHS[I]	EQUAL_PAIN_MUX/Q_INSI_0/II	420	300 0	311 013
SWIICH_BLUCKS[6]	.SWIICH_BLUCK/output[0]_to	SWIICH_BLUCKS[7]	.SWIICH_BLUCK/EQUAL_PAIHS[2]	EQUAL_PATH_MUX/Q_INSI_0/II	243	203 4	100 385
SWITCH_BLUCKS[6]	.SWITCH_BLUCK/output[0]_to	SWITCH_BLUCKS[7]	.SWITCH_BLUCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/11	244	204 4	110 388
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[0]	.EQUAL_PATH_MUX/Q_INST_0/I0	310	258 5	585 480
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[1]	.EQUAL_PATH_MUX/Q_INST_0/I0	313	260 5	589 484
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[2]	.EQUAL_PATH_MUX/Q_INST_0/I0	279	228 5	551 448
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[3]	.EQUAL_PATH_MUX/Q_INST_0/I0	279	228 5	553 450
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[0]	.EQUAL_PATH_MUX/Q_INST_0/I5	188	160 3	338 285
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[1]	.EQUAL_PATH_MUX/Q_INST_0/I5	189	161 3	342 287
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/15	146	127 2	248 212
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[2]_to		.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/15	362	301 6	388 565
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[3]_to		.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/12	357	296 6	379 557
SWITCH_BLOCKS[6]	.SWITCH_BLOCK/output[3]_to	.SWITCH_BLOCKS[7]	.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/12	319	269 6	318 512
SWITCH BLOCKS[6]	.SWITCH BLOCK/output[3] to .	.SWITCH BLOCKS [7]	.SWITCH BLOCK/EQUAL PATHS[2]	EQUAL PATH MUX/Q INST 0/12	280	234 5	531 439
SWITCH BLOCKS[6]	.SWITCH BLOCK/output[3] to .	.SWITCH BLOCKS [7]	.SWITCH BLOCK/EQUAL PATHS[3]	EQUAL PATH MUX/Q INST 0/12	185	157 3	336 281
SWITCH BLOCKS[7]	.SWITCH BLOCK/output[0] to .	.SWITCH BLOCKS[8]	.SWITCH BLOCK/EQUAL PATHS[0]	EQUAL PATH MUX/Q INST 0/11	578	488 1	1085 90
SWITCH BLOCKS [7]	.SWITCH_BLOCK/output[0] to		.SWITCH_BLOCK/EQUAL PATHS[1]	EQUAL_PATH_MUX/0 INST 0/11	260	223 4	174 399
SWITCH BLOCKS [7]	.SWITCH_BLOCK/output[0] to		.SWITCH_BLOCK/EQUAL PATHS [2]	EQUAL_PATH_MUX/0 INST 0/11	504	424 9	337 777
SWITCH BLOCKS [7]	.SWITCH_BLOCK/output[0] to	SWITCH_BLOCKS [8]	.SWITCH_BLOCK/EQUAL PATHS[3]	EQUAL_PATH_MUX/O INST 0/11	292	254 F	505 430
SWITCH BLOCKS [7]	.SWITCH BLOCK/output[1] to	.SWITCH BLOCKS [8]	SWITCH BLOCK/EDUAL PATHS [0]	EQUAL PATH MUX/0 INST 0/10	232	204 4	108 350
SWITCH BLOCKS [7]	.SWITCH BLOCK/output[1] to	.SWITCH BLOCKS [8]	.SWITCH BLOCK/EDUAL PATHS[1]	EQUAL PATH MUX/0 INST 0/10	333	284 F	334 530
SWITCH BLOCKS [7]	SWITCH BLOCK/output[1] to	SWITCH BLOCKS [8]	SWITCH BLOCK/FOILAL PATHS[2]	EQUAL PATH MUX/0 INST 0/10	437	367 9	330 687
SWITCH BLOCKS[7]	SWITCH BLOCK (output [1] to	SWITCH BLOCKS [8]	SWITCH BLOCK/FOUNT DATHS[3]	FOUNT DATH MUY/O INST 0/10	437	367 8	832 689
SWITCH BI OCKO [7]	SWITCH BLOCK/output[2] +o	SWITCH BI OCKG [0]	SWITCH BLOCK/FOUNT DATHS[0]	FOILAL PATH MILY /O TNST O/TE	442	370 9	835 600
CUITCH BLOCKS [7]	SWITCH BLOCK/output[2]_to_	CUITCU DI OCVC [0]	SWITCH BLOCK/EQUAL_IATHS[0]	FOUNT DATE MUX/0 INST 0/15	127	267 0	200 002
CUTTON DE DOVO [7]	CUTTON DI DOV ([0] .	GUITTON DLOUKS[8]	CUITCU DI OCV (DOUAL PAIRS[1]	FUINT DATE MILY (0 THOT C (15	220	201 0	,JZ 009
SWIIGEBLUCKS[7]	SWITCH_BLUCK/OUTPUT[2]_to		SWITCH_BLUCK/EQUAL_PAIHS[2]	EQUAL_PAIN_MUX/Q_INSI_0/15	332	203 t	130 527
SWIICH_BLUCKS[7]	.Swiich_BLUCK/output[2]_to		.SWITCH_BLUCK/EQUAL_PATHS[3]	EQUAL_PAIH_MUX/Q_INST_0/15	200	229 4	192 415
SWIICH_BLUCKS[7]	.SWIICH_BLUCK/output[3]_to		.SWIICH_BLUCK/EQUAL_PATHS[0]	EQUAL_PAIH_MUX/U_INST_0/I2	282	245 5	JU9 432
SWITCH_BLOCKS[7]	.SWIICH_BLUCK/output[3]_to	SWITCH_BLOCKS[8]	.SWITCH_BLUCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/12	362	307 6	J87 573
SWITCH_BLOCKS[7]	.SWITCH_BLOCK/output[3]_to		.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/12	363	308 6	389 575
SWITCH_BLOCKS[7]	.SWITCH_BLOCK/output[3]_to		.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/12	362	307 6	587 573
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[0]	.EQUAL_PATH_MUX/Q_INST_0/I1	200	173 3	366 310
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/I1	199	172 3	363 308
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/I1	240	204 4	155 381
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/I1	240	204 4	154 380
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/10	237	201 4	158 385
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[1]	EQUAL_PATH_MUX/Q_INST_0/10	302	254 5	599 497
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[9]	.SWITCH_BLOCK/EQUAL_PATHS[2]	EQUAL_PATH_MUX/Q_INST_0/10	410	341 8	304 662
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[1]_to		.SWITCH_BLOCK/EQUAL_PATHS[3]	EQUAL_PATH_MUX/Q_INST_0/10	411	342 8	308 665
SWITCH_BLOCKS[8]	.SWITCH_BLOCK/output[2]_to .		.SWITCH_BLOCK/EQUAL_PATHS[0]	EQUAL_PATH_MUX/Q_INST_0/15	346	287 6	393 569
SWITCH BLOCKS [8]	.SWITCH_BLOCK/output[2] to		.SWITCH_BLOCK/EQUAL PATHS[1]	EQUAL_PATH_MUX/Q INST 0/15	345	286 6	393 569
SWITCH BLOCKS [8]	.SWITCH_BLOCK/output[2] to		.SWITCH_BLOCK/EQUAL PATHS [2]	EQUAL_PATH_MUX/0 INST 0/15	240	202 4	188 405
SWITCH BLOCKS [8]	.SWITCH BLOCK/output[2] to	.SWITCH BLOCKS [9]	SWITCH BLOCK/EDUAL PATHS[3]	EQUAL PATH MUX/0 INST 0/15	241	203 4	192 408
GUITCH DI DOVG [0]	SWITCH BLOCK / output [2] _to	QUITCH DI NOVO [0]	SWITCH BLOCK/EQUAL_FAIRS[5]	FOUNT DATH MUY/O TNOT 0/10	481	402 4	135 770
· · · · · · · · · · · · · · · · · · ·				/ w// 12	-01	-02 3	

SWITCH_BLOCKS[8].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [9].SWITCH_BLOCK/EQUAL_PATHS [1].EQUAL_PATH_MUX/Q_INST_0/12	481 402 935 772
SWITCH_BLOCKS[8].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [9] . SWITCH_BLOCK/EQUAL_PATHS [2] . EQUAL_PATH_MUX/Q_INST_0/12	318 264 632 521
SWITCH_BLOCKS[8].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [9] . SWITCH_BLOCK/EQUAL_PATHS [3] . EQUAL_PATH_MUX/Q_INST_0/12	341 294 654 550
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[10].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1	347 288 677 557
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS [10] .SWITCH_BLOCK/EQUAL_PATHS [1] .EQUAL_PATH_MUX/Q_INST_0/I1	242 204 480 398
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [2] . EQUAL_PATH_MUX/Q_INST_0/I1	348 289 679 558
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [3] . EQUAL_PATH_MUX/Q_INST_0/I1	252 208 525 432
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS [10] .SWITCH_BLOCK/EQUAL_PATHS [0] .EQUAL_PATH_MUX/Q_INST_0/IC	136 119 251 214
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS [10] .SWITCH_BLOCK/EQUAL_PATHS [1] .EQUAL_PATH_MUX/Q_INST_0/IC	347 288 682 560
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS [10] .SWITCH_BLOCK/EQUAL_PATHS [2] .EQUAL_PATH_MUX/Q_INST_0/IC	242 204 477 396
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS [10] .SWITCH_BLOCK/EQUAL_PATHS [3] .EQUAL_PATH_MUX/Q_INST_0/IC	243 205 481 399
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [0] . EQUAL_PATH_MUX/Q_INST_0/IE	240 202 479 397
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [1] . EQUAL_PATH_MUX/Q_INST_0/IE	345 286 676 556
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [2] . EQUAL_PATH_MUX/Q_INST_0/I5	133 116 248 212
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [3] . EQUAL_PATH_MUX/Q_INST_0/I5	136 119 252 215
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [0] . EQUAL_PATH_MUX/Q_INST_0/12	238 202 446 375
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [1] . EQUAL_PATH_MUX/Q_INST_0/12	239 203 450 377
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [10] .SWITCH_BLOCK/EQUAL_PATHS [2] .EQUAL_PATH_MUX/Q_INST_0/12	411 342 792 652
SWITCH_BLOCKS[9].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS [10] . SWITCH_BLOCK/EQUAL_PATHS [3] . EQUAL_PATH_MUX/Q_INST_0/12	412 343 796 655
SWITCH_BLOCKS[10].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS [11] .SWITCH_BLOCK/EQUAL_PATHS [0] .EQUAL_PATH_MUX/Q_INST_0/I	1 360 298 687 563
SWITCH BLOCKS[10].SWITCH BLOCK/output[0] to .	.SWITCH BLOCKS[11].SWITCH BLOCK/EQUAL PATHS[1].EQUAL PATH MUX/Q INST 0/I	1 148 128 255 216
SWITCH BLOCKS[10] SWITCH BLOCK/output[0] to	SWITCH BLOCKS[11] SWITCH BLOCK/EQUAL PATHS[2] EQUAL PATH MUX/Q INST 0/1	1 443 371 844 696
SWITCH BLOCKS[10] SWITCH BLOCK/output[0] to	SWITCH BLOCKS[11] SWITCH BLOCK/FOUAL PATHS[3] FOUAL PATH MUX/O INST 0/I	1 439 367 843 694
SWITCH BLOCKS[10] SWITCH BLOCK/output[1] to	SWITCH BLOCKS[11] SWITCH BLOCK/FOUAL PATHS[0] FOUAL PATH MUX/0 INST 0/1	0 367 305 690 567
SWITCH BLOCKS[10] SWITCH BLOCK/output[1] to	SWITCH BLOCKS[11] SWITCH BLOCK/FOUAL PATHS[1] FOUAL PATH MUX/0 INST 0/1	0 367 305 692 569
SWITCH BLOCKS[10] SWITCH BLOCK/output[1] to	SWITCH BLOCKS[11] SWITCH BLOCK/EQUAL_FAILS[1] LQUAL_FAIL_NOK/Q_INCT_0/I	0 269 228 497 414
SUITCH BLOCKS[10] SWITCH BLOCK/Output[1] to	SWITCH_BECORD[11] SWITCH_BECORD EQUAL TATES[2] EQUAL TATE NOV Q_INDI_0/I	0 165 145 070 022
SUITCH BLOCKS[10] SUITCH BLOCK/output[1]_t0	SUTTCH RIDCKS[11] SUTTCH RIDCK/FOILAT DATUS[0] EQUAL_FAIL_RUA/Q_INSI_0/1	5 106 167 250 005
SUITCH BLOCKS[10] SUITCH BLOCK/OUtput[2]_t0	SUTTCH RIDCKS[11] SUTTCH RIDCK/EDIMI DATUS[1] EDIMI DATU MUX/Q_INSI_0/1	5 106 167 251 004
CUITCH PLOCKS[10] SWITCH_BLOCK/OUtput[2]_to_	SWITCH_BLOCKS[II].SWITCH_BLOCK/EQUAL_FAINS[I].EQUAL_FAIN_MOX/Q_INSI_0/I	5 190 107 331 294
CUTTCH BLOCKS[10] CUTTCH BLOCK/OUTPUC[2]_TO	CUTTON DECONDENSION DECON EQUAL FAIRS[2]. EQUAL FAIR_NUX/Q_INSI_0/1	5 233 240 040 404 5 000 042 544 450
SWITCH_BLUCKS[10].SWITCH_BLUCK/output[2]_to	.SWITCH_BLOCKS[11].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INSI_0/I	5 290 243 544 450
SWITCH_BLUCKS[10].SWITCH_BLUCK/output[3]_to	.SWITCH_BLUCKS[11].SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MOX/Q_INSI_0/I	2 282 235 537 443
SWITCH_BLUCKS[10].SWITCH_BLUCK/output[3]_to	.SWITCH_BLOCKS[11].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INS1_0/1	2 332 281 636 528
SWITCH_BLUCKS[10].SWITCH_BLUCK/output[3]_to	.SWITCH_BLUCKS[11].SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	2 441 369 837 690
SWITCH_BLUCKS[10].SWITCH_BLUCK/output[3]_to	.SWITCH_BLUCKS[11].SWITCH_BLUCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	2 197 168 352 295
SWITCH_BLUCKS[11].SWITCH_BLUCK/output[0]_to	.SWITCH_BLUCKS[12].SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	1 405 343 785 655
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	1 319 275 599 506
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	1 456 386 907 755
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	1 280 207 559 383
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	0 431 360 824 678
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	0 280 234 531 438
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	0 252 213 500 416
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	0 151 127 273 228
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	5 457 378 860 704
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	5 415 347 796 656
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[12].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	5 520 434 1012 833
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS [12] .SWITCH_BLOCK/EQUAL_PATHS [3] .EQUAL_PATH_MUX/Q_INST_0/I	5 289 234 551 429
SWITCH_BLOCKS[11].SWITCH_BLOCK/output[3]_to	.SWITCH_BLOCKS [12] .SWITCH_BLOCK/EQUAL_PATHS [0] .EQUAL_PATH_MUX/Q_INST_0/I	2 268 230 451 380
SWITCH BLOCKS [11] . SWITCH BLOCK/output [3] to .	.SWITCH BLOCKS [12] .SWITCH BLOCK/EQUAL PATHS [1] .EQUAL PATH MUX/Q INST 0/I	2 272 234 456 384
SWITCH BLOCKS[11]. SWITCH BLOCK/output[3] to .	.SWITCH BLOCKS[12].SWITCH BLOCK/EQUAL PATHS[2].EQUAL PATH MUX/Q INST 0/I	2 466 388 881 724
SWITCH BLOCKS[11], SWITCH BLOCK/output[3] to	SWITCH BLOCKS[12] SWITCH BLOCK/EQUAL PATHS[3] EQUAL PATH MUX/Q INST 0/1	2 470 357 884 623
SWITCH BLOCKS[12] SWITCH BLOCK/output[0] to	SWITCH BLOCKS[13] SWITCH BLOCK/FOUAL PATHS[0] FOUAL PATH MUX/0 INST 0/1	1 231 203 408 349
SWITCH BLOCKS[12] SWITCH BLOCK/output[0] to	SWITCH BLOCKS[13] SWITCH BLOCK/FOUAL PATHS[1] FOUAL PATH MUX/O INST 0/1	1 343 271 658 483
SWITCH BLOCKS[12] SWITCH BLOCK/output[0] to	SWITCH BLOCKS[13] SWITCH BLOCK/FOUAL PATHS[2] FOUAL PATH MUX/0 INST 0/1	1 333 285 628 526
SWITCH BLOCKS [12] SWITCH BLOCK /output [0] to	SWITCH BLOCKS[13] SWITCH BLOCK/EQUAL_FAILS[2] EQUAL_FAIL_NOK/Q_INCT_0/I	1 438 369 830 688
SWITCH BLOCKS [12] SWITCH BLOCK / output [1] to	SWITCH BLOCKS[13] SWITCH BLOCK/EQUAL_FAILS[0] EQUAL_FAIL_NOX/Q_INCT_0/I	0 443 372 827 685
SWITCH BLOCKS [12] SWITCH BLOCK /output [1] to	SWITCH BLOCKS[13] SWITCH BLOCK/EQUAL_FAILD[0].LQOKL_FAIL_NOK/Q_INDI_0/I	0 403 314 813 599
SUITCH BLOCKS[12] SWITCH BLOCK/Output[1] to	SWITCH_BECORD[13] SWITCH_BECORD EQUAL TATES[1] EQUAL TATE NOV Q_INDI_0/I	0 400 260 202 620
SUITCH BLOCKS[12] SWITCH BLOCK/Output[1] to	SWITCH_BECORD[10]. SWITCH_BECORD EQUAL TATIS[2]. EQUAL TATI NOV Q_INST_0/I	0 440 270 207 625
CUITCH DIOCKS[12] SWITCH_BLOCK/OUtput[1]_to_	SWITCH_BLOCKS[13].SWITCH_BLOCK/EQUAL_FAINS[3].EQUAL_FAIN_MOX/Q_INSI_0/I	E 444 370 827 885
CUITCH PLOCKS[12] SWITCH_BLOCK/OUtput[2]_to_	SWITCH_BLOCKS[13].SWITCH_BLOCK/EQUAL_FAINS[0].EQUAL_FAIN_MOX/Q_INS1_0/I	5 444 370 819 075
CUITCH PLOCKS[12] SWITCH_BLOCK/OUtput[2]_to_	SWITCH_BLOCKS[13].SWITCH_BLOCK/EQUAL_FAINS[1].EQUAL_FAIN_MOX/Q_INS1_0/I	5 449 339 830 380 E 442 260 910 67E
SWITCH_BEDCKS[12].SWITCH_BEDCK/OUtput[2]_t0	SWITCH_BLOCKS[13].SWITCH_BLOCK/EQUAL_FAINS[2].EQUAL_FAIN_MOX/Q_INSI_0/I	5 443 305 815 075
SUITCH BLOCKS[12] SWITCH BLOCK/ Output[2] to	SUITCH RIDCKS[13] SUITCH RIDCK/EDIMI DATUS[0] EDUML FAIL NUX/U_INSI_0/I	2 370 311 401 570
SUITCH BLOCKS[12] SWITCH BLOCK/ Output[2] +-	SUTTOR BLOCKS[13] SUTTOR BLOCK/EQUINT DATUS[1] EQUAL_FAIL_RUA/U_INSI_0/I	2 3/0 311 031 0/2 2 3/0 311 031 0/2
SWITCH BLOCKS[12] SWITCH BLOCK/output[2] +0	SWITCH RINCKS[13] SWITCH RINCK/FRIMAT DATUS[3] FRIMATI MUV/0 THOT 0/7	2 367 308 676 561
SUITCH BLOCKS[12] SWITCH BLOCK/ Output[2] +-	SUTTOR BLOCKS[13] SUTTOR BLOCK/EQUINT DATUS[2] EQUAL_TAIL_NUA/U_INSI_0/I	2 302 260 526 452
SUITCH BLOCKS[13] SUITCH BLOCK/ OUTPUT[3] +-	SUTTCH RIDCKS[14] SUTTCH RIDCK/EQUAL_FAIRS[5].EQUAL_FAIR_MUX/Q_INSI_0/1	1 281 235 507 402
SUITCH BLOCKS[13] SWITCH BLOCK/OUTPUC[0]_TO	SUTTCH RIDCKS[1/] SUTTCH RIDCK/EQUAL_FAIDS[0]. EQUAL_FAID_NUX/Q_INSI_0/1	1 201 200 007 420
SWITCH_BEDCKS[13].SWITCH_BEDCK/OUtput[0]_t0	SWITCH_BLOCKS[14] SWITCH_BLOCK/EQUAL_FAINS[1].EQUAL_FAIN_MOX/Q_INSI_0/I	1 450 239 514 420
SWITCH_BLUCKS[IS].SWITCH_BLUCK/OUtput[0]_t0	. SWITCH_BLUCKS[14] . SWITCH_BLUCK/EQUAL_PAINS[2] . EQUAL_PAIN_MOX/Q_INSI_0/I	1 452 373 844 690
SWITCH_BLUCKS[13].SWITCH_BLUCK/output[0]_to	.SWITCH_DLUCKS[14].SWITCH_BLUCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	1 409 342 858 596
SWIICH_BLUCKS[13].SWITCH_BLUCK/output[1]_to	.SWITCH_DEUCKS[14].SWITCH_BEUCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	0 309 305 693 569
SWIICH_BLUCKS[13].SWITCH_BLUCK/output[1]_to	.SWITCH_DEUCKS[14].SWITCH_BEUCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	0 302 300 684 561
SWIICH_BLUCKS[13].SWITCH_BLOCK/output[1]_to	.swiich_blucks[14].swiich_bluck/equal_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	0 284 238 547 451
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[1]_to	.SWITCH_BLUCKS[14].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	0 288 213 555 379
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[2]_to	.SWITCH_BLUCKS[14].SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	5 152 132 259 220
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	5 150 130 255 217
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	5 358 297 675 554
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[2]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	5 262 198 505 351
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[3]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	2 352 293 668 549
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[3]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	2 453 374 847 693
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[3]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	2 245 207 429 357
SWITCH_BLOCKS[13].SWITCH_BLOCK/output[3]_to	.SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	2 249 205 437 358
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	1 361 302 707 583
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	1 281 237 544 452
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I	1 366 307 712 588
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[0]_to	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I	1 281 237 544 452
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I	0 358 299 700 577
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[1]_to	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I	0 320 272 609 508
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[1] to .	.SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/O INST 0/I	0 350 296 664 553
SWITCH BLOCKS[14] SWITCH BLOCK/output[1] to	.SWITCH BLOCKS [15].SWITCH BLOCK/EQUAL PATHS [3].EQUAL PATH MUX/Q INST 0/I	0 156 139 282 243
		0 100 100 101 110

	SWITCH BLOCKS[15] SWITCH BLOCK/FOUNT DATHS[0] FOUNT DATH MUY/O INST 0/15 198 171 358 30	4
SWITCH BLOCKS [14].SWITCH BLOCK/output [2] to .	SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_NOX/Q_INST_0/15 366 307 711 580	т В
SWITCH BLOCKS [14]. SWITCH BLOCK/output [2] to .	SWITCH BLOCKS[15].SWITCH BLOCK/EQUAL PATHS[2].EQUAL PATH MUX/Q INST 0/15 269 231 517 43	4
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/15 366 307 711 58	в
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/12 289 245 565 47	1
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I2 194 168 370 31	3
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I2 208 182 385 320	в
SWITCH_BLOCKS[14].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 204 178 380 32	3
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 250 213 456 384	4
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 253 216 460 38	7
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I1 529 442 1031 8	52
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/11 256 218 457 38	3
SWITCH_BLUCKS[15].SWITCH_BLUCK/output[1]_to	SWITCH_BLUCKS[16].SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_M0X/Q_INST_0/10/333/2/9/10/583	9
SWIICH_BLUCKS[15].SWIICH_BLUCK/OUTput[1]_to	SWITCH_BLUCKS[16].SWITCH_BLUCK/EQUAL_AATHS[1].EQUAL_PAIH_MUX/Q_INS1_0/10/438/363/912/5	1
SWITCH BLOCKS[15] SWITCH BLOCK/output[1] to_	SWITCH_BLUCKS[16] .SWITCH_BLUCK/EUVAL_FAIDS[2].EUVAL_FAID_MUK/U_INSI_0/10 465 366 950 /6 SWITCH_BLUCKS[16] SWITCH_BLUCK/EDVINI DATHMUK/I DATHMUK/O INST_0/IO 257 222 558 A/A	5
SWITCH BLOCKS[15] SWITCH BLOCK/output[2] to	SWITCH BLOCK [16] SWITCH BLOCK FOILL PATHOLOJ FOILL PATH MUX/0 INST 0/15 400 331 789 644	R
SWITCH BLOCKS[15]. SWITCH BLOCK/output[2] to .	SWITCH BLOCK [16] SWITCH BLOCK/EQUAL PATHS[1] EQUAL PATH MUX/Q INST 0/15 296 248 593 493	1
SWITCH BLOCKS[15]. SWITCH BLOCK/output[2] to .		6
SWITCH BLOCKS[15]. SWITCH BLOCK/output[2] to .	SWITCH BLOCKS[16].SWITCH BLOCK/EQUAL PATHS[3].EQUAL PATH MUX/Q INST 0/15 389 322 804 66	1
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/12 323 269 645 53	3
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/12 322 268 643 53	1
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/12 338 282 635 529	5
SWITCH_BLOCKS[15].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[16].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 335 279 631 52:	1
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 496 416 915 75	Б
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 388 327 727 603	3
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I1 282 244 487 41	1
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I1 426 355 804 66	1
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/IO 281 241 491 41	3
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I0 335 286 598 50	1
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[1]_to	SWITCH_BLUCKS[17].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/IO 422 356 777 64	ь -
SWIICH_BLUCKS[16].SWITCH_BLUCK/output[1]_to	SWITCH_BLUCKS[17], SWITCH_BLUCK/EQUAL_PATHS[3], EQUAL_PATH_MUX/Q_INST_0/IO 255 217 464 38	1
SWIICH_BLUCKS[16].SWIICH_BLUCK/output[2]_to	SWITCH_BLUCKS[17].SWITCH_BLUCK/EQUAL_AIHS[0].EQUAL_PAIH_MUX/Q_INS1_0/15 366 313 694 58.	1
SWITCH_BLUCKS[16] SWITCH_BLUCK/OUtput[2]_to	SWITCH_DLUCKS[1/].SWITCH_DLUCK/EUVAL_FAINS[1].EUVAL_FAIN_MUX/U_INSI_0/15/200/226/459/35.	2
SWITCH BLOCKS[16] SWITCH BLOCK/output[2] to_	SWITCH_BLOCKS[17] SWITCH_BLOCK/EQUAL_FAINS[2] EQUAL_FAIN_MOX/Q_INGI_0/10 407 034 079 72	6
SWITCH BLOCKS[16] SWITCH BLOCK/output[3] to	SWITCH_BLOCKS[17] SWITCH_BLOCK/FOILL_PATHS[0] FOILL_PATH MIX/0 INST 0/12 469 395 885 733	3
SWITCH BLOCKS[16] SWITCH BLOCK/output[3] to	SWITCH_BLOCKS[17] SWITCH_BLOCK/FOILL_PATHS[1] FOILL_PATH MUX/0 INST 0/12 391 333 748 622	3
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[3] to .		5
SWITCH_BLOCKS[16].SWITCH_BLOCK/output[3]_to	SWITCH_BLOCKS[17].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I2 449 381 839 700	0
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I1 197 171 362 30	7
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I1 77 61 140 113	
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I1 369 310 704 58:	2
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[0]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I1 370 311 708 58	5
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I0 460 384 861 710	0
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/IO 385 278 738 494	4
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I0 248 214 433 364	6
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I0 251 217 437 369	9
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[2]_to	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/15_263_225_501_419	9
SWITCH_BLUCKS[17].SWITCH_BLUCK/output[2]_to	SWITCH_BLUCKS[18].SWITCH_BLUCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/15 298 211 575 360	6
SWITCH BLOCKS[17] SWITCH BLOCK/output[2] to_	SUITCH RICCKS[18] SUITCH BICCK/EQUAL_PAIRS[2].EQUAL_PAIR_MOX/Q_INSI_0/10 309 310 700 57	9
SWITCH BLOCKS[17] SWITCH BLOCK/output[3] to	SWITCH_BLOCK_[18] SWITCH_BLOCK/FOILL_PATHE[5] FOILL_PATH MIX/0 INST 0/12 370 311 718 59	4
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3] to .	SWITCH BLOCK [18] SWITCH BLOCK/EQUAL PATHS[1] EQUAL PATH MUX/Q INST 0/12 341 271 668 49	4
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3] to .		-
	SWIICH BLUCKSIISI.SWIICH BLUCK/EUUAL PAIHSIZI.EUUAL PAIH MUX/U INSI 0712-265-227-513-430	0
SWITCH BLOCKS[17].SWITCH BLOCK/output[3] to .	SWIICH_BLUCKS[18].SWIICH_BLUCK/EQUAL_PAIHS[2].EQUAL_PAIH_MUX/Q_INS1_0/12 265 227 513 430SWIICH BLOCKS[18].SWIICH BLOCK/EQUAL PATHS[3].EQUAL PATH MUX/Q INST 0/12 266 228 517 433	0 3
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to	Swilch_BLUCKS[18].Swilch_BLUCK/EQUAL_PAINS[2].EQUAL_PAIH_RUX/Q_INS1_0/12 266 227 513 434 SWITCH_BLUCKS[18].SWITCH_BLUCK/EQUAL_PAINS[3].EQUAL_PAIH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PAINS[0].EQUAL_PAIH_MUX/Q_INST_0/11 484 407 922 761	0 3 5
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to	Swiller_BLOCKS[19].Swiller_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INS1_0/12_265_227_513_43 Swiller_BLOCKS[19].Swiller_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12_266_228_517_433 Swiller_BLOCKS[19].Swiller_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/11_402_340_772_64	0 3 5 4
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to	SWIICH_BLUCKS[19].SWIICH_BLUCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INSI_0/12 265 227 513 43 SWIICH_BLUCKS[19].SWIICH_BLUCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INSI_0/12 266 228 517 43 SWIICH_BLUCKS[19].SWIICH_BLUCK/EQUAL_PATIS[0].EQUAL_PATH_MUX/Q_INSI_0/11 440 792 276 SWIICH_BLUCKS[19].SWIICH_BLUCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INSI_0/11 402 340 772 64 SWIICH_BLUCKS[19].SWIICH_BLUCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INSI_0/11 425 356 809 661	0 3 5 4 8
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to	SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INS]_0/12 265 227 513 437 SWITCH_BLUCKS[18].SWITCH_BLUCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/11 484 407 922 761 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/11 484 407 922 761 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/11 482 556 809 661 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/11 383 321 732 564	0 3 5 4 8 4
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_	SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INSI_0/12 265 227 513 433 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INSI_0/12 265 228 517 433 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[0].EQUAL_PATH_MUX/Q_INSI_0/11 402 340 772 64 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INSI_0/11 402 340 772 64 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INSI_0/11 426 356 809 666 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INSI_0/11 426 3521 735 265 SWITCH_BLUCKS[19].SWITCH_BLUCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INSI_0/11 321 733 264	0 3 5 4 8 4 3
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[0].EQUAL_PATH_MUX/Q_INST_0/11 402 340 792 761 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 428 232 1732 584 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 393	0 3 5 4 8 4 3 3
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 484 407 922 761 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[1].EQUAL_PATH_MUX/Q_INST_0/11 484 407 922 761 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 484 407 922 761 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 483 931 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 393 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 393 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 485 403 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATIS[3].EQUAL_PATIS_0/10 249 208 485 403 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATIS[3].EQUAL_PATIS[3].SWITCH_PATIS[0 3 5 4 8 4 3 3 3 1
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 352 173 264 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 211 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 211 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 485 400 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 485 400 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 128 122 258 122 258 124 228 521 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 2258 124 255 124 2258 124 255 124 124 255 124 255 124 255 124 124 255 124 255 124 255 124 124 255 124 255 124 124 124	0 3 5 4 8 4 3 3 1 3 4
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to .SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to .SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[0].EQUAL_PATH_MUX/Q_INST_0/11 402 340 792 76 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 428 556 609 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 428 352 173 265 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[1].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 39 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 39 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 148 122 268 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 148 122 258 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/157 156 333 331 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/157 156 333 331 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATIS[2].EQUAL_PATIS[2].SWITCH_BLOCKS[20].SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATIS[2].SWITCH_BLOCKS[20].SWITCH_SUCR_PATIS[2].EQUAL_PATIS[20].PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWITCH_SUCR_PATIS[20].SWI	0 3 5 4 8 4 3 3 1 3 4 4 7
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 222 fol 43 SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 439 321 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 204 465 40 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 142 2268 213 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/154 343 363 381 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/154 349 765 397 675 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/154 349 765 397 675 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 434 363 301 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 343 663 406 69 69 69 69 69 69 69 69 69 69 69 69 69	0 3 5 4 8 4 3 3 1 3 4 7 8
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_	SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 661 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 33 118 249 211 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 211 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 485 400 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 148 122 258 211 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 346 363 831 664 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 346 363 316 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 297 679 567 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 364 297 307 567 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 368 297 679 567 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 364 307 307 567 SWITCH_BLOCKS[14].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 368 371 669 373 377 567 SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 368 297 679 567 SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 363 273 757 575 SWITCH_BLOCKS[15].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 363 373 677 575 575 575 575 575 575 575 575 575	0 3 5 4 4 3 3 1 3 4 4 7 8 3
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 428 556 809 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 428 351 809 661 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 39 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 236 198 475 39 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/10 148 122 258 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/16 436 333 168 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/15 362 397 679 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/15 483 404 066 744 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/15 483 73 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 483 73 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PA	0 3 5 4 8 8 4 3 3 3 3 4 7 8 8 3 4
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 438 321 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 25 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 22 585 13 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 138 237 679 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 297 679 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 293 573 444 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 391 765 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 352 35 573 444 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_P	0 3 5 4 8 4 3 3 3 1 3 4 7 8 3 4 4 4
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_toSWITCH_BLOCKS [18]_SWITCH_BLOCKS [18]_SWITCH_BLOCKS [18]_SWITCH_SUUCKS [18]_SWITCH_BLOCKS [18]_SWITCH_BLO	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 265 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 366 809 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 335 113 249 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 486 400 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 146 122 258 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 348 363 381 68: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 348 237 679 557 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 348 273 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 282 238 537 444 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 288 238 537 445 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 245 547 547	0 3 5 4 8 4 3 3 3 1 3 4 7 8 3 4 4 4 7
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 43; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 43; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 439 321 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 58 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[1].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[1].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 143 163 465 40 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 142 208 465 40 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 433 436 331 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 438 304 966 74 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 189 170 350 29 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 425 353 474 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 128 173 550 29 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 128 173 550 29 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 281 221 255 537 4000000000000000000000000000000000000	0 3 5 4 8 4 3 3 1 3 4 7 8 3 4 4 7 5
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_SLOCKS[18].SWITCH_BLOCK/oUTput[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTput[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTput[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTput[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTput[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTput[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTPUT[3]_to_ SWITCH_SLOCKS[18].SWITCH_SLOCK/OUTPUT[3]_to_ SWITCH_SLOCKS[18].SWITCH_SL	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 433 SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 420 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 420 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 420 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 609 66 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 138 118 249 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 138 118 249 22: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 485 40 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 434 363 831 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 434 469 807 44 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 438 373 167 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 438 373 176 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 438 373 176 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 438 373 176 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 537 447 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 537 547 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 47 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 47 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 47 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 47 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PA	0 3 5 4 8 4 3 3 3 1 3 4 7 7 8 3 4 4 7 5 8 8
SWITCH_BLOCKS[17].SWITCH_BLDCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLDCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLDCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLDCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLDCK/output[0]_to_	SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 43; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 517 43; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 366 809 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 809 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 35 118 249 21; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 22; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 486 40; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 146 122 258 21; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 148 22 363 316 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 346 363 316 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 297 679 555 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 917 655 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 282 238 537 444 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 282 238 537 445 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 288 238 537 457 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 477 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 576 477 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 576 477 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 287 243 576 477 SWITCH_BLOCKS[20].SWITCH_	0 3 5 4 4 3 3 1 3 4 7 8 3 4 4 7 5 8 8 1
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 613 43; SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 617 43; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 138 118 248 24; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 148 128 258 13; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/16 343 436 381 68; SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 483 404 906 74 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 483 404 906 74 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 379 769 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 379 165 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 379 165 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 379 165 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 379 165 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 478 399 17 65 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 478 399 176 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 478 399 197 55 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 478 399 197 55 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL	0 3 5 4 8 8 4 3 3 1 3 4 7 8 3 4 4 7 5 8 1 6
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 631 43: SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 617 43: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 689 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 689 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 689 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 426 356 689 666 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21: SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 221 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 249 208 485 40 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 434 368 381 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 434 366 381 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 483 737 165 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 488 373 107 650 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 483 737 165 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 129 170 350 29 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 128 724 575 477 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 127 128 583 537 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 128 734 565 477 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 128 734 565 475 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 247 398 937 465 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 247 386 876 477 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 424 316 628 570 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQ	0 3 5 4 8 4 3 3 1 3 4 7 5 8 3 4 4 7 5 8 1 6 9
SWITCH_BLOCKS[17].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[1]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[2]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[18].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[3]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_ SWITCH_BLOCKS[19].SWITCH_BLOCK/output[0]_to_	SWITCH, BLOCKS [13].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 265 227 513 43; SWITCH, BLOCKS [13].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 265 228 517 43; SWITCH, BLOCKS [13].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 426 356 809 66 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 135 118 249 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 148 122 258 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 148 122 258 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/16 148 238 238 63 31 68; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/16 148 232 258 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/16 348 363 31 68; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/16 348 373 917 65; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 288 273 7917 65; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/17 288 239 537 44; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/17 288 239 537 44; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/17 288 239 537 44; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/17 287 243 575 477 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/17 281 248 340 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/17 291 218 583 400 SWITCH, BLOCKS [19].SWIT	0 3 5 4 4 8 4 3 3 1 3 4 7 8 3 4 4 7 5 8 8 1 6 9 7 7
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_	SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/12 265 227 613 433 SWITCH_BLOCKS[18].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/12 266 228 617 433 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[2].EQUAL_PATH_MUX/Q_INST_0/11 402 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 403 340 772 64 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 389 321 732 68 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 25 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/10 135 118 22 583 14 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 297 679 55 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 358 293 573 440 360 741 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/15 350 293 573 440 350 350 SWITCH_BLOCKS[19].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 243 583 340 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 243 585 340 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 473 339 391 765 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 473 339 391 765 SWITCH_BLOCKS[20].SWITCH_BLOCK/EQUAL_PATIS[3].EQUAL_PATH_MUX/Q_INST_0/11 473 339 391 765 SWITCH_BLOCKS[20].SWI	0 3 5 5 4 8 4 3 3 1 3 4 7 8 3 4 4 7 5 8 8 4 4 7 5 8 9 7 9 9
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_	SWITCH_BLOCKS [13]. SWITCH_BLOCK/EQUAL_PATIS [2]. EQUAL_PATH_MUX/Q_INST_0/12 265 227 513 43: SWITCH_BLOCKS [13]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/12 265 228 517 43: SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/11 426 356 609 666 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 21: SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/10 135 118 249 241 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/10 148 122 258 21: SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/10 148 122 258 21: SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/15 158 297 679 55 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/15 168 297 679 55 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/15 488 373 317 65 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/15 282 295 374 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/12 285 239 537 SWITCH_BLOCKS [19]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 477 SWITCH_BLOCKS [20]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/12 287 243 575 477 SWITCH_BLOCKS [20]. SWITCH_BLOCK/EQUAL_PATIS [3]. EQUAL_PATH_MUX/Q_INST_0/11 227 188 487 40 SWITCH_BLOCKS [20]. SWITCH_BLOCK/EQUAL_PATIS	0 3 5 5 4 8 4 3 3 1 3 4 4 7 8 3 4 4 7 5 8 1 6 9 7 9 2 2
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_	SWITCH, BLOCKS [18].SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/12 266 228 517 433 SWITCH, BLOCKS [18].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 266 228 517 433 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [2].EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [2].EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 68 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 68 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 68 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 148 198 475 39; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/16 343 633 816 68 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 483 404 906 74 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 488 379 169 55; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 488 379 169 55; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 289 537 445 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 289 587 485 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 289 587 485 455 455 455 455 455 455 455 455 455	0 3 5 5 4 8 4 3 3 1 3 4 7 5 8 1 6 9 7 9 2 3 9
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_	SWITCH, ELOCKS [18].SWITCH, ELOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/12 266 226 517 433 SWITCH, ELOCKS [18].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 266 226 517 433 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 426 356 809 60 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 138 118 249 21: SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 138 118 249 21: SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 249 208 485 40 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 434 363 831 68 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 434 363 831 68 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 438 373 167 655 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 373 917 655 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 373 917 655 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 575 47 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 575 47 SWITCH, ELOCKS [19].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 575 47 SWITCH, ELOCKS [20].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 575 47 SWITCH, ELOCKS [20].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 575 47 SWITCH, ELOCKS [20].SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 575 47 SWITCH, ELOCKS [0035484333134478344758816997923000
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_	SWITCH, BLOCKS [13].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/12 265 227 513 43; SWITCH, BLOCKS [13].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/11 426 256 717 43; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/11 426 356 809 666 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/11 426 356 809 666 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/11 426 356 809 666 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/11 335 113 249 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/10 135 118 249 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/10 148 122 258 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/10 249 208 485 400 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/10 148 122 258 21; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/15 358 297 679 557. SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/15 358 297 679 557. SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/15 348 373 177 65; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, O/15 488 373 177 65; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATHS [3].EQUAL, PATH, MUX/Q, INST, O/16 488 373 177 65; SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATHS [3].EQUAL, PATH, MUX/Q, INST, O/16 287 245 575 477 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATHS [3].EQUAL, PATH, MUX/Q, INST, O/11 227 188 487 40 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATHS [3].EQUAL, PATH, MUX/Q, INST, O/11 227 188 487 40 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATHS [3].EQUAL, PATH, MUX/Q, INST, O/11 227 188 487 40 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATHS [3].EQUAL, PATH, MUX/Q, INST, O/11 227 188 487 40 SWITCH, BLOCKS [20].SWITC	0 3 5 4 8 4 3 3 1 1 3 4 7 7 8 3 4 4 7 5 8 1 6 9 7 9 2 3 0 3 3
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_	SWITCH, BLOCKS [18]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/12 266 228 517 433 SWITCH, BLOCKS [18]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 266 228 517 433 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 58 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 58 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 138 118 249 21 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 138 118 22 583 13 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 1434 563 381 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 458 240 496 06 74 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 8917 65 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 373 917 65 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 373 917 65 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 282 557 44. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 478 399 917 55 SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 147 359 917 55 SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 473 399 917 55 SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/1	035548433113478334477581669792300332
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_	SWITCH, BLOCKS [18].SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/12 266 226 517 43: SWITCH, BLOCKS [18].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 426 436 647 922 761 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 426 356 649 660 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 426 356 649 660 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 426 356 649 660 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 426 356 649 660 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 211 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 249 208 4455 40 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/10 148 122 258 11 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 434 363 831 66 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 458 297 679 55 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 458 297 676 55 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/15 458 373 177 65. SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 287 243 573 47 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 287 243 576 47 SWITCH, BLOCKS [19].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/12 287 126 553 537 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 247 339 919 75 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 247 316 487 40 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 247 316 487 40 SWITCH, BLOCKS [20].SWITCH, BLOCK/EQUAL, PATIS [3].EQUAL, PATH, MUX/Q, INST, 0/11 247 316 486 460 SWITCH, BLOCKS [20].SWITCH,	0 3 5 4 8 4 3 3 1 3 4 7 5 8 1 6 9 7 9 2 3 0 3 3 2 8
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/OUTPUT [1]_	SWITCH, ELOCKS [18]. SWITCH, ELOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/12 266 228 517 433 SWITCH, ELOCKS [18]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 403 940 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 58 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 148 198 475 39 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 148 208 485 40 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 443 463 318 68 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 483 404 966 74 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 767 55 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 767 55 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 379 767 55 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 379 767 55 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 379 767 55 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 379 767 55 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 370 470 55 SWITCH, ELOCKS [20]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 248 576 77 SWITCH	0 0 3 5 5 4 8 4 3 3 3 1 3 4 4 7 7 5 8 1 6 9 7 9 2 3 0 3 3 3 2 8 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_	SWITCH, ELOCKS [18]. SWITCH, ELOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/12 266 226 517 433 SWITCH, ELOCKS [18]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 402 407 922 f61 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 402 340 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/11 402 340 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/11 403 340 772 64 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/11 389 321 732 68 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/10 135 118 249 21 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/10 136 118 249 52 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/10 136 118 22 583 14 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/16 143 453 351 68 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 448 463 351 68 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 488 373 917 65. SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 488 373 917 65. SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/16 488 373 917 65. SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 247 243 5753 744 SWITCH, ELOCKS [19]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/12 291 218 633 400 SWITCH, ELOCKS [20]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 477 359 9597 SWITCH, ELOCKS [20]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 477 359 9597 SWITCH, ELOCKS [20]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 477 359 9597 SWITCH, ELOCKS [20]. SWITCH, ELOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 474 359 9597 	0 0 3 5 4 8 4 4 3 3 1 1 3 4 4 7 5 8 1 1 6 9 7 9 2 3 0 3 3 2 8 0 2
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_	SWITCH, BLOCKS [13]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 265 227 613 433 SWITCH, BLOCKS [13]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 266 228 517 433 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 266 228 517 433 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 426 356 689 666 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 426 356 689 666 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 428 321 733 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 138 118 249 211 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 148 122 258 21 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 148 122 258 21 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 148 122 258 21 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 348 368 381 66 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 348 373 107 65. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 348 373 107 65. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 644 640 607 44 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 537 44. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 287 243 576 47. SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 247 124 586 576 47. SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 247 136 587 647. SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 247 136 587 647. SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INS	0 0 3 5 5 4 8 4 4 3 3 1 1 3 4 4 7 5 8 8 1 6 9 7 9 2 3 0 3 3 2 8 0 2 9 9
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_	SWITCH, BLOCKS [18]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/12 266 228 517 43: SWITCH, BLOCKS [18]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/11 389 321 732 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 135 118 249 21: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 138 118 249 24: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/10 148 142 268 21: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 1434 363 381 68: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/15 488 379 769 55: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/16 488 379 176 55: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/17 1478 399 917 55: SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, 0/12 249 256 574 4. SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, 0/11 477 389 919 75: SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, I	0 0 3 5 4 8 4 3 3 1 3 4 7 5 8 1 6 9 7 9 2 3 0 3 3 2 8 0 2 9 9
SWITCH_BLOCKS [17].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [0]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [18].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [1]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [2]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_to_ SWITCH_BLOCKS [19].SWITCH_BLOCK/output [3]_	SWITCH, BLOCKS [18]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/12 266 226 517 43: SWITCH, BLOCKS [18]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 402 407 922 f61 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/11 402 340 772 64 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [2]. EQUAL, PATH, MUX/Q, INST, O/11 389 321 732 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 389 321 732 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/10 135 118 249 21: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/10 138 118 249 52: SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/11 349 208 465 40 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 438 363 316 68 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 368 297 679 55 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 368 297 679 55 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 368 297 679 55 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/15 368 373 917 65. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/16 488 373 917 65. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/16 387 3917 65. SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/12 287 243 575 47 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/12 287 243 575 47 SWITCH, BLOCKS [19]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/12 287 243 575 47 SWITCH, BLOCKS [20]. SWITCH, BLOCK/EQUAL, PATIS [3]. EQUAL, PATH, MUX/Q, INST, O/12 287 243 575 47 	003554843331334778344477581697792300332280029999

SWITCH_BLOCKS[20].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[21].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I1 482 381 9	31 697
SWITCH_BLOCKS[20].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[21].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I0 491 423 9	16 772
SWITCH_BLOCKS[20].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[21].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I0 593 502 1	109 923
SWITCH_BLOCKS[20].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[21].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I0 426 358 8	32 689
SWITCH_BLOCKS[20].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[21].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I0 598 471 1	120 828
SWITCH_BLOCKS [20].SWITCH_BLOCK/output [2]_toSWITCH_BLOCKS [21].SWITCH_BLOCK/EQUAL_PATHS [0].EQUAL_PATH_MUX/Q_INST_0/I5 574 482 1	077 891
SWITCH BLOCKS [20], SWITCH BLOCK/output [2] to SWITCH BLOCKS [21], SWITCH BLOCK/EQUAL PATHS [1], EQUAL PATH MUX/Q INST 0/15 342 292 6	07 510
SWITCH BLOCKS[20] SWITCH BLOCK/output[2] to SWITCH BLOCKS[21] SWITCH BLOCK/EDIAL PATHS[2] FOULL PATH MUX/0 INST 0/15 490 411 9	23 764
SWITCH BLOCKS[20] SWITCH BLOCK/output[2] to SWITCH BLOCKS[21] SWITCH BLOCK/EDIAL PATHS[3] FOULL PATH MUX/0_INST_0/15_344_283_6	11 483
Suttch BLOCKS[00] SUTCh BLOCK (Autor) [3] + SUTCH BLOCKS[01] SUTCH BLOCK (BLOCK FOLL DATH MIX (A LINK) (4) INT (A LINK) (4) INT (A LINK) (4) INT (4) I	22 762
GUTTOU DIOOKS[O] SUTTOU DIOOKS (AUTOUR [0] + C) - CUTTOU DIOOKS[O] SUTTOU DIOOKS[O] DIOUS (AUTOUR) AUTOUR (AUTOUR) (AUTOUR [0] + CUTTOU DIOOKS[O] SUTTOU SUTTOU DIOOKS[O] SUTTOU SUTOU S	04 496
SWICE_BLOCK [20] .SWICE_BLOCK (Utplut [3]_00 SWICE_BLOCK [21].SWICE_BLOCK (EQAL_FAIRS[1].EQAL_FAIRS[1].EQAL(FAIRS[1]) [20] [20] [20] [20] [20] [20] [20] [20]	09 920
Swilch_BLUCK0[20].Swilch_BLUCK/OUtput[3]_t0Swilch_BLUCK2[21].Swilch_BLUCK/EQUAL_FAIR5[2].EQUAL_FAIR5[2].EQUAL_FAIR5[2].COM (42107) 200 050 5	92 334
Swilch_BLUCKS[20].Swilch_BLUCK/OUtput[3]_t0Swilch_BLUCK/SUID=BLUCK/EQUAL_FAIDS[3].EQUAL_FAID=NOX/Q_IND=0/12 400 205 30	12 427
SWITCH_BLUCKS[21].SWITCH_BLUCK/OUTPut[0]_toSWITCH_BLUCKS[22].SWITCH_BLUCK/EQUAL_PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].EQUAL[PAIRS[0].	25 689
SWITCH_BLUCKS[21].SWITCH_BLUCK/output[0]_toSWITCH_BLUCKS[22].SWITCH_BLUCKS[22].SWITCH_BLUCK/CEQUAL_PATHS[1].EQUAL_PATHS	32 535
SWITCH_BLUCKS[21].SWITCH_BLUCK/output[0]_toSWITCH_BLUCKS[22].SWITCH_BLUCK/EQUAL_PATHS[2].EQUAL_PATH_MOX/Q_INST_0/11 504 426 9	52 799
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[0]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I1 511 395 9	76 705
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I0 192 166 3	82 323
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I0 190 164 3	78 320
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I0 466 389 9	40 775
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I0 194 162 3	86 321
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I5 316 267 6	52 541
SWITCH_BLOCKS[21].SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[22].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I5 315 266 6	48 538
SWITCH BLOCKS [21], SWITCH BLOCK/output [2] to SWITCH BLOCKS [22], SWITCH BLOCK/EQUAL PATHS [2], EQUAL PATH MUX/Q INST 0/15 265 221 5	53 455
SWITCH BLOCKS [21] SWITCH BLOCK/output [2] to SWITCH BLOCKS [22] SWITCH BLOCK/EQUAL PATHS [3] EQUAL PATH MUX/Q INST 0/15 344 249 6	92 463
SWITCH BLOCKS[21] SWITCH BLOCK/output[3] to SWITCH BLOCKS[22] SWITCH BLOCK/EDIAL PATHS[0] FOILAL PATH MUX/0 INST 0/12 473 393 9	39 775
SWITCH BLOCK[21] SWITCH BLOCK/output[3] to SWITCH BLOCKS[22] SWITCH BLOCK/FOILL PATHS[1] FOILAL PATH MIX/O INST 0/12 466 388 9	30 767
Suttch BLOCKS[01] SUTCh BLOCK (Autor) [3] + SUTCh BLOCKS[02] SUTCh BLOCK (FOLD AUTOR) BLOCK (C) DATH MIX (A 1997) (A 199	99 502
GUTTCH DIOCK/2[1]. DWTTCH DIOCK/GUTTCH DIOCK/2[2]. GUTTCH DIOCK/2[1]. DATUMIX/GING/2[1]. AND () (12 20 20 30 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 40 20 20 20 40 20 20 20 40 20 20 20 40 20 20 20 20 20 20 20 20 20 20 20 20 20	01 505
SWICE_BLOCKC[21].SWICE_BLOCK/OUCPUC[3]_CCSWICE_BLOCK[22].SWICE_BLOCK/EQUAL_FAIRS[3].EQUAL_FAIRS[4].VC_TWOT_V/12_3VICE_BLOCKC[4].	DI 000
Swilch_BLUCK0[22].Swilch_BLUCK/OUtput[0]_t0Swilch_BLUCK2[23].Swilch_BLUCK/EQUAL_FAIRS[0].EQUAL_FAIR_MOX/Q_IND[0/11 193 10/ 3	50 303
SWITCH_BLUCKS[22].SWITCH_BLUCK/OUTPut[0]_toSWITCH_BLUCKS[23].SWITCH_BLUCK/EQUAL_PAIRS[1].	57 302
SWITCH_BLUCKS[22].SWITCH_BLUCK/output[0]_toSWITCH_BLUCKS[23].SWITCH_BLUCKS[22].SWITCH_BLUCK/output[0]_toSWITCH_BLUCKS[22].SWITC	72 317
SWITCH_BLUCKS[22].SWITCH_BLUCK/output[0]_toSWITCH_BLUCKS[23].SWITCH_BLUCK/EUQAL_PATH_MUX/Q_INST_0/11 287 243 5	50 458
SWITCH_BLUCKS[22].SWITCH_BLUCK/output[1]_toSWITCH_BLUCKS[23].SWITCH_BLUCK/EQUAL_PATHS[0].EQUAL_PATH_MOX/Q_INST_0/10 365 306 7	07 584
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I0 155 138 2	81 242
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/IO 160 143 2	85 247
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[1]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/I0 164 147 2	90 251
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I5 221 194 4	21 359
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/I5 365 306 7	10 587
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[2].EQUAL_PATH_MUX/Q_INST_0/I5 291 247 5	67 473
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[2]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[3].EQUAL_PATH_MUX/Q_INST_0/15 365 306 7	10 587
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[0].EQUAL_PATH_MUX/Q_INST_0/I2 281 237 5	57 463
SWITCH_BLOCKS[22].SWITCH_BLOCK/output[3]_toSWITCH_BLOCKS[23].SWITCH_BLOCK/EQUAL_PATHS[1].EQUAL_PATH_MUX/Q_INST_0/12 316 268 6	01 501
SWITCH_BLOCKS [22].SWITCH_BLOCK/output [3]_toSWITCH_BLOCKS [23].SWITCH_BLOCK/EQUAL_PATHS [2].EQUAL_PATH_MUX/Q_INST_0/12 322 274 6	04 505
SWITCH_BLOCKS [22].SWITCH_BLOCK/output [3]_toSWITCH_BLOCKS [23].SWITCH_BLOCK/EQUAL_PATHS [3].EQUAL_PATH_MUX/Q_INST_0/12 197 171 3	73 316
SWITCH BLOCKS[23],SWITCH BLOCK/output[0] toAPUF ARBITER/FF 2 1/0 reg/D 301 251 620 510	
SWITCH BLOCKS [23], SWITCH BLOCK/output [0] to APUF ARBITER/FF 3 1/0 reg/D 363 303 728 600	
SWITCH BLOCKS[23] SWITCH BLOCK/output[0] to APIF ABBITER/FF 4 1/0 reg/D 310 258 605 497	
Suffice BLOCKS[33] SUffice BLOCK(Authorit[1] to ADDE ABBITER/EE 2 1/0 ran/C 323 275 600 503	
SUTCH BLOCK[23] SUTCH BLOCK/OUTDUT[1] to ADIF ARRITR/FE 3 2/0 reg/b 413 249 701 660	
SUITCH BLOCKS [23] SWITCH BLOCK/output[1] to APRITER/F 5 2/0 reg/b 386 324 776 645	
SUTTY BIO(S) SUTT	
CUTIVE_DEGREG 20 CUTIVE_DEGREGATION_DEGREGATION AND A CONTRACT STATEMENT STATEME	
JULIANDER 100005 [20] JANING DULOWA/OULDUL[2] LO APUT ANDITAN/F _0_//Q_TEG/V 319 2/1 053 020	
SHILON DUVIS [23]. SHILON DUVIN [2]. U APUT ANDITAN (T 4) (U 204 235 04/ 404	
SHILD_DUUKS[23]. SHILD_DUUK/OUCDU[3]_U0 APUT_ANDITAN/F 4_1/Q_T89/C 254 214 4/1 392	
SWI1CH_ELUCKS[23].SWI1CH_ELUCK/output[3]_toAPUF_ARBITEK/FF_4_2/U_reg/C 2b4 214 4/1 392	
SWITCH_BLUCKS[23].SWITCH_BLUCK/output[3]_toAPUF_ARBITER/FF_4_3/Q_reg/C 254 214 471 392	

References

- Artix-7. Apr. 1, 2020. URL: https://www.xilinx.com/products/ silicon-devices/fpga/artix-7.html.
- [2] S. V. Sandeep Avvaru, Ziqing Zeng, and Keshab K Parhi. "Homogeneous and Heterogeneous Feed-Forward XOR Physical Unclonable Functions". eng. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 2485–2498. ISSN: 1556-6013.
- [3] Christoph Böhm and Maximilian Hofer. *Physical Unclonable Functions in Theory and Practice*. Springer-Verlag New York, 2013.
- [4] ChipWhisperer Software. Apr. 1, 2020. URL: https://github.com/ newaetech/chipwhisperer.
- [5] Chip Whisperer Wiki. Apr. 1, 2020. URL: https://wiki.newae.com/ Main%5C_Page.
- [6] CW1173 ChipWhisperer-Lite. Apr. 1, 2020. URL: https://wiki.newae. com/CW1173_ChipWhisperer-Lite.
- [7] CW305 Artix FPGA Target. Apr. 1, 2020. URL: https://wiki.newae. com/CW305_Artix_FPGA_Target.
- [8] E. Dubrova. "A Reconfigurable Arbiter PUF with 4 x 4 Switch Blocks". In: 2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL). May 2018, pp. 31–37. DOI: 10.1109/ISMVL.2018.00014.
- E. Dubrova et al. "CRC-PUF: A Machine Learning Attack Resistant Lightweight PUF Construction". In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW). June 2019, pp. 264–271. DOI: 10.1109/EuroSPW.2019.00036.
- [10] Elena Dubrova, Martin Brisfors, and Bernhard Degen. "A Model of Arbiter PUF with 4 x 4 Switch Blocks".
- [11] Ilze Eichhorn, Patrick Koeberl, and Vincent van der Leest. "Logically reconfigurable PUFs: memory-based secure key storage". In: Proc. of the sixth ACM workshop on Scalable Trusted Computing. STC '11. Chicago, Illinois, USA: ACM, 2011, pp. 59–64. ISBN: 978-1-4503-1001-7. DOI: 10. 1145/2046582.2046594. URL: http://doi.acm.org/10.1145/2046582.2046594.
- [12] Yansong Gao and Damith C. Ranasinghe. R³PUF: A Highly Reliable MemRistive Device based Reconfigurable PUF. Tech. rep. ArXive Technical report, Feb. 2017.
- Wei Ge et al. "FPGA implementation of a challenge pre-processing structure arbiter PUF designed for machine learning attack resistance". eng. In: *IEICE Electronics Express* 17.2 (2020). ISSN: 13492543. URL: http://search.proquest.com/docview/2349877475/.

- [14] Rekha Govindaraj, Swaroop Ghosh, and Srinivas Katkoori. "Design, Analysis and Application of Embedded Resistive RAM based Strong Arbiter PUF". eng. In: *IEEE Transactions on Dependable and Secure Computing* (2018), pp. 1–1. ISSN: 1545-5971.
- [15] K. Kursawe et al. "Reconfigurable Physical Unclonable Functions Enabling technology for tamper-resistant storage". In: Proc. of IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'09). July 2009, pp. 22–29. DOI: 10.1109/HST.2009.5225058.
- [16] Daihyun Lim. "Extracting secret keys from integrated circuits". MA thesis. Massachusetts Institute of Technology, 2004.
- M. Majzoobi, F. Koushanfar, and M. Potkonjak. "Testing Techniques for Hardware Security". In: 2008 IEEE International Test Conference(ITC). Vol. 00. Oct. 2009, pp. 1–10.
- [18] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. "Lightweight secure PUFs". eng. In: Proceedings of the 2008 IEEE/ACM International Conference on computer-aided design. ICCAD '08. IEEE Press, 2008, pp. 670–673. ISBN: 9781424428205.
- [19] Phuong Ha Nguyen et al. The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks. Cryptology ePrint Archive, Report 2018/350. 2018.
- [20] Ioannis Papakonstantinou and Nicolas Sklavos. "Physical Unclonable Functions (PUFs) Design Technologies: Advantages and Trade Offs". In: Jan. 2018, pp. 427–442. ISBN: 978-3-319-58423-2. DOI: 10.1007/978-3-319-58424-9_24.
- [21] Durga Prasad Sahoo et al. "A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security". eng. In: *IEEE Transactions* on Computers 67.3 (2018), pp. 403–417. ISSN: 0018-9340.
- [22] Soubhagya Sutar, Arnab Raha, and Vijay Raghunathan. "D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems". In: Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems. CASES '16. Pittsburgh, Pennsylvania: ACM, 2016, 12:1–12:10. ISBN: 978-1-4503-4482-1. DOI: 10.1145/2968455.2968519. URL: http://doi.acm.org/10. 1145/2968455.2968519.
- [23] Vivado. Apr. 1, 2020. URL: https://www.xilinx.com/products/ design-tools/vivado.html.
- [24] Nils Wisiol and Marian Margraf. "Why attackers lose: design and security analysis of arbitrarily large XOR arbiter PUFs". eng. In: *Journal of Cryptographic Engineering* 9.3 (2019), pp. 221–230. ISSN: 2190-8508.

TRITA-EECS-EX-2020:258

www.kth.se