From Unbalanced to Perfect: Implementation of Low Energy Stream Ciphers

Jikang Lin[1,2], Jiahui He[1,2], Yanhong Fan[1,2,3] (✉), and Meiqin Wang[1,2,3]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
{linjikang,hejiahui2020}@mail.sdu.edu.cn, {yanhongfan,mqwang}@sdu.edu.cn
[2] School of Cyber Science and Technology, Shandong University, Qingdao, China
[3] Quan Cheng Shandong Laboratory, Jinan, China

Abstract. Low energy is an important aspect of hardware implementation. For energy-limited battery-powered devices, low energy stream ciphers can play an important role. In `IACR ToSC 2021`, Caforio et al. proposed the Perfect Tree energy model for stream cipher that links the structure of combinational logic circuits with state update functions to energy consumption. In addition, a metric given by the model shows a negative correlation with energy consumption, i.e., the higher the balance of the perfect tree, the lower the energy consumption. However, Caforio et al. didn't give a method that eliminate imbalances of the unrolled strand tree for the existing stream ciphers.
In this paper, based on the Perfect Tree energy model, we propose a new redundant design model that improve the balances of the unrolled strand tree for the purpose of reducing energy consumption. In order to obtain the redundant design, we propose a search algorithm for returning the corresponding implementation scheme. For the existing stream ciphers, the proposed model and search method can be used to provide a low-power redundancy design scheme. To verify the effectiveness, we apply our redundant model and search method in the stream ciphers (e.g., `Trivium` and `Kreyvium`) and conducted a synthetic test. The results of the energy measurement demonstrate that the proposed model and search method can obtain lower energy consumption.

Keywords: Low Energy · Stream Cipher · Hardware Implementation · `Trivium`.

1 Introduction

Hardware implementations of symmetric ciphers focus on several hardware performance index such as low latency, low area, low power and low energy consumption. For battery-powered devices, such as portable devices, medical implant devices or RFID tags, the implementation of low energy consumption plays an important role.

Power and energy are correlated physical variables, energy is essentially the integral of power over time, and power is the amount of energy consumed per

unit of time, i.e.,

$$E = \int P dt.$$

The energy consumption of semiconductor circuits reflects the total work done by the voltage source during the execution of any operation. The low energy consumption design reduces battery consumption, which is important in battery-driven devices with limited energy supply. In the last few years, there have been a series of works [4,20,7,8,12,21,22,14,2,3,9] on the energy consumption of symmetric ciphers.

For block ciphers, the authors in [2] investigated the architectural design of each component (S-box, MixColumn), the clock frequency and the impact of serialization or unrolling design strategy on energy consumption. To explain the relation between the degree of unrolling and energy efficiency, the authors gave a model that illustrated the energy consumption in each clock cycle as a quadratic function of the degree of unrolling r, with the following expression

$$E(r) = \left(Ar^2 + Br + C\right) \cdot \left(1 + \left\lceil \frac{R}{r} \right\rceil\right),$$

where $\left(Ar^2 + Br + C\right)$ denotes the energy consumed per cycle and $\left(1 + \lceil \frac{R}{r} \rceil\right)$ is the total clock cycles required for encryption[4]. Based on this model, the degree of unrolling of the energy-efficient optimal block ciphers implementation can be predicted. They concluded that the degree of unrolling $r = 2$ was the optimal configuration for lightweight block ciphers (e.g., `Present` [6], `TWINE` [24] and `Simon` [5]), while for other block ciphers (e.g., `AES` [12], `Noekeon` [11] and `Piccolo` [23]), the degree of unrolling $r = 1$ was optimal. Based on the above model of energy consumption in any r-round unrolled block cipher architecture, the authors developed energy-efficient linear and non-linear layers, and proposed a block cipher for low energy called `Midori` in [1].

For encrypting significantly large data, the stream cipher is a better scheme than the block cipher. In [3], the authors showed that an unrolled stream cipher circuit was more energy-efficient when the encryption of multiple data blocks was considered instead of a single block. The authors found that a `Trivium` [13] implementation at degree of unrolling $r = 160$ was about 9 times more energy efficient than any block cipher-based large data encryption scheme, implying that unrolled stream ciphers generally outperformed block ciphers.

For optimizing the energy consumption of stream ciphers, the literature [9] proposed an energy model (i.e., Perfect Tree model) that links the underlying algebraic structure of the state update function to the energy consumptive characteristic. The authors divided the whole circuit into smaller circuit strands, which mainly comprise of the logic functions related to one register update. Since these strands are interconnected, they seem like a tree. By observing the variation of the power consumption of the circuit strands in the above tree, the

[4] Where A, B, C are constants, R is the number of iterations of the round function specified in the design of the cipher, and r is the degree of unrolling of the cipher.

authors found that power consumption was related to the balance degree of the tree, i.e., if the balance degree of the tree is higher, the corresponding energy consumption of the tree circuit is lower.

However, the implementation in [9] did not take into account the elimination of imbalances in the unrolled strand tree for the exiting stream ciphers, the imbalances can lead to more glitches. In this paper, we propose a redundant design model for reducing these glitches, and give a method to search the redundant design scheme.

Our Contributions. In this paper, based on a new redundant design model, we propose a search algorithm for implementation scheme of the stream cipher, and reimplement r-round unrolled stream cipher circuits under the redundant scheme. Moreover, we conduct a synthetic test to obtain the results of the energy measurements. We now list our contributions as follows.

1. **A redundant design model for reducing glitches:** We discuss the factors influencing energy consumption in semiconductor circuits, of which toggle rate is the one focused on in this paper, and illustrate the relation between glitches and toggle rate. These discussions demonstrate that we can improve energy efficiency by reducing glitches. The glitches in the circuit are produced by inconsistent input delays of the combinational logic circuit modules, so we can reduce the glitches by balancing the delays of all the inputs of the modules.
   According to the definition of Perfect Tree energy model in [9], each circuit strand (combinational logic module) corresponds to an unrolled strand tree, and the balance of this tree corresponds to the input delay balance of the strand. If we want to convert an unbalanced tree into a balanced one, a natural idea is to add child nodes to the unbalanced tree. Corresponding to the circuit, this means adding additional combinational logic modules to the circuit, which we call redundant modules.
2. **A search algorithm for implementation schemes:** The essence of our redundant design is to optimize the input ports with lower delay, i.e., to connect these input ports to the redundant modules. The optimization method of the ports depend on the parameters of the update function of the ciphers. We build a set of mappings of the circuit strand to their input ports as a way to present the entire scheme of implementation. We present our generic search algorithm, which takes the tap locations of the register as input and the implementation scheme of the circuit strand as output.
3. **Apply our model and search method in the stream ciphers:** To verify the efficiency of our model and algorithm, we apply them to `Trivium` [13] and `Kreyvium` [10]. We obtain the implementation scheme with redundant design using the search algorithm, and describe the circuit in VHDL. Then we use Synopsys Design Compiler and Synopsys VCS to complete the synthesis, post-synthesis simulation and energy analysis. As shown in Table 1, the results of the energy measurements demonstrate our redundant design model and search method can obtain lower energy consump-

tion. VHDL codes for describing redundant design schemes are available at https://github.com/JKLinsdu/RedundantDesign.

Lib	Cipher	Design	Total Power (uW)	Energy (nJ/Mbit)
Lib.1	`Trivium`	w/o	2552.9	93.0
	($r = 288$)	w/	2394.7	87.2
	`Kreyvium`	w/o	2848.4	116.7
	($r = 256$)	w/	2723.5	111.6
Lib.2	`Trivium`	w/o	2758.1	100.4
	($r = 288$)	w/	2577.9	93.9
	`Kreyvium`	w/o	2637.0	108.1
	($r = 256$)	w/	2454.5	100.6

Table 1. Part of power/energy measurements in this paper. w/: With redundant design, w/o: Without redundant design (corresponding to the implementation in [9]), Lib.1: TSMC 90 nm, Lib.2: UMC 55 nm

Outline of the Paper. The rest of the paper is organized as follows. Section 2 gives the specification of Perfect Tree energy model including the definition of circuit strand, unrolled strand tree and perfect m-ary tree, in addition, the energy consumption in semiconductor circuits is also discussed. In Section 3, we discuss the glitches in semiconductor circuits and gives a redundant design model that reduces glitches. Section 4 gives an algorithm for searching for ports to which the outputs of redundant modules should be connected. In Section 5, We apply the redundant design to stream ciphers and give some test result to illustrate the effectiveness of the proposed model and search method. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this section, we first review the definition of circuit strand and illustrate the role of the Restricted directive for circuits, using `Trivium` as an example. We then further review the relation between the circuit strand and tree to introduce the concepts of Perfect m-ary Tree in Perfect Tree energy model. Finally, we describe the energy consumption in semiconductor circuits, and point out the direction for further optimization.

2.1 Circuit Strand

Definition 1 (Circuit Strand [9]). *Update functions of a stream cipher correspond to different combinational logic circuit modules. the combinational logic circuit module can be denoted as circuit strand.*

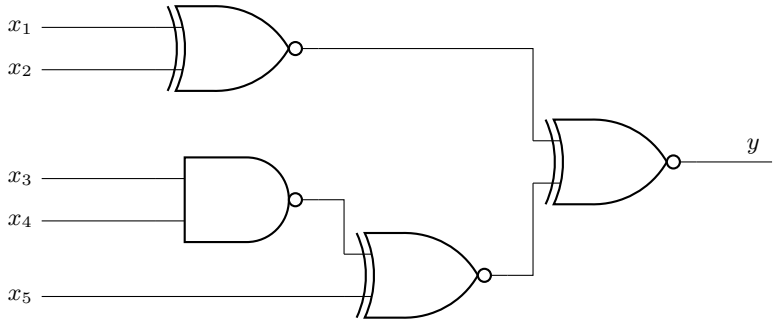
In the case of `Trivium`, it consists of the following three update functions

$$t_1 \leftarrow s_{65} + (s_{90} \cdot s_{91}) + s_{92} + s_{170}$$
$$t_2 \leftarrow s_{161} + (s_{174} \cdot s_{175}) + s_{176} + s_{263}$$
$$t_3 \leftarrow s_{242} + (s_{285} \cdot s_{286}) + s_{287} + s_{68},$$

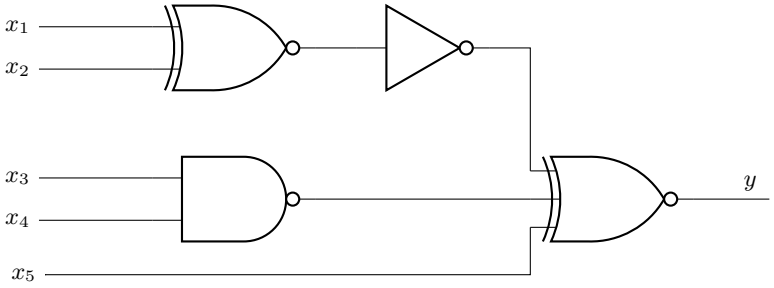
where s_i $(1 \leq i \leq 288)$ are the bits in the state register.

Thus, the update functions of `Trivium` correspond to three independent circuit strands (combinational logic circuit modules). These circuit strands can be represented as

$$x_1 + x_2 + (x_3 \cdot x_4) + x_5.$$



(a) 1 NAND2, 3 XNOR2



(b) 1 NAND2, 1 XNOR2, 1 XNOR3, 1 NOT

Fig. 1. The combinational logic circuit module that corresponds to the circuit strand.

For the circuit strand above, there are many circuit implementation schemes. Fig. 1 gives two schemes with four gates (i.e., Fig. 1a and Fig. 1b). Take a `Trivium` implementation at degree of unrolling $r = 288$ as an example, when using the Restricted directive of Synopsys Design Compiler, there will be $288 \times 3 = 864$ of such modules at the gate-level netlist. And these modules are connected to each other, but without breaking individual boundaries.

The Synopsys Design Compiler has a variety of compilation directives, which choose different mappings and optimization for the circuit. The experiments in [9] show that for the circuits compiled under the Restricted directive, the power consumption was much lower than for circuits compiled under the Regular or Ultra directives. When using the Restricted directive, the state update circuit for $r = 1$ is simply replicated for higher degrees of unrolling, i.e., each circuit strand has the same gates and structure, and the output of each strand may be the input of other modules and the input of each strand may be the output of other modules.

2.2 Unrolled Strand Tree and Perfect m-ary Tree

In [9], an energy model applied to stream ciphers is proposed which uses a tree structure to portray energy efficiency, i.e., a good shaped tree corresponds to high energy efficiency, while a bad shaped tree corresponds to low energy efficiency. According to Definition 1, the update function of `Trivium` consists of three strands

$$\begin{aligned} t_1 &= s_{66} + s_{93} + (s_{91} \cdot s_{92}) + s_{171} \\ t_2 &= s_{162} + s_{177} + (s_{175} \cdot s_{176}) + s_{264} \\ t_3 &= s_{243} + s_{288} + (s_{286} \cdot s_{287}) + s_{69}. \end{aligned}$$

Definition 2 (Strand in the r-th Unrolled Round [9]). *$t_i(r)$ denotes the strand for equation t_i in the r-th unrolled round, where $i \in \{1, 2, 3\}$, and when $r \in \{1, \ldots, 288\}$, $t_i(r)$ can be written in recursive form for `Trivium` as*

$$\begin{aligned} t_1(r) &= t_3(r-66) + t_3(r-93) + [t_3(r-91) \cdot t_3(r-92)] + t_1(r-78) \\ t_2(r) &= t_1(r-69) + t_1(r-84) + [t_1(r-82) \cdot t_1(r-83)] + t_2(r-87) \\ t_3(r) &= t_2(r-66) + t_2(r-111) + [t_2(r-109) \cdot t_2(r-110)] + t_3(r-69), \end{aligned}$$

where $t_1(r) = s_{94-r}$[5], $t_2(r) = s_{178-r}$ and $t_3(r) = s_{1-r}$ when $r \leq 0$.

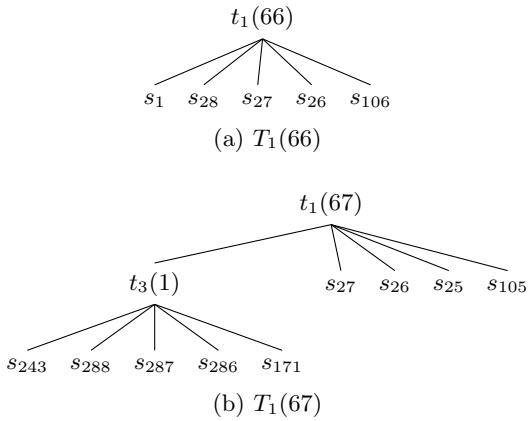
Definition 3 (Unrolled Strand Tree [9]). *A strand $t_i(r)$ can be written in a recursive form, i.e., a strand $t_i(r)$ can be represented as a unrolled strand tree $T_i(r)$ with the root node as the output bit whose subtrees are other unrolled strand trees or leaf nodes.*

In general, the leaf nodes of the unrolled strand tree represent the states in the register, and the non-leaf nodes represent the outputs of strands. As shown in Fig. 2, we take the example of $t_1(66)$ and $t_1(67)$, whose corresponding unrolled strand trees are $T_1(66)$ and $T_1(67)$.

Definition 4 (Perfect m-ary Tree [9]). *A perfect m-ary tree is a tree in which all non-leaf nodes have m children and all leaf nodes are at the same depth.*

[5] We denote the state bits in the register by s_i $(1 \leq i \leq 288)$.

It is noted that not all unrolled strand trees are perfect m-ary trees. The model in [9] determines whether the circuit strand corresponding to the unrolled strand tree is highly energy efficient precisely by distinguishing whether the unrolled strand tree is a perfect m-ary tree, in other words, the perfect m-ary tree is the tree of good shape in the model.



(a) $T_1(66)$

(b) $T_1(67)$

Fig. 2. The unrolled strand tree that corresponds to the circuit strand. $T_1(66)$ is perfect tree, but $T_1(67)$ is not. This means that $t_1(66)$ is energy-efficient and $t_1(67)$ is inefficient.

Since a strand corresponds to an unrolled strand tree, we can estimate the total number of perfect trees in any r-round unrolled implementation of `Trivium`. The total number of perfect trees is a metric given by [9], and this metric shows a negative correlation with energy consumption. Therefore, we can improve energy efficiency by increasing this metric, and more specifically, we can convert an unbalanced unrolled strand tree into a perfect tree.

2.3 Energy Consumption in Semiconductor Circuits

In semiconductor circuits, there are two mainly reasons for producing energy consumption:

(a) *Dynamic dissipation.* Dynamic dissipation is due to the charging and discharging of load capacitances and the short-circuit current in semiconductor circuits. on the frequency of the clock driving the circuit.
(b) *Static dissipation.* Static dissipation is due to leakage current and other current drawn continuously from the power supply.

In detail, the total energy dissipation for a CMOS gate can be written as

$$E = E_{switching} + E_{internal} + E_{leakage},$$

where $E_{internal}$ is internal energy, $E_{switching}$ is switching energy and $E_{leakage}$ is leakage energy. And dynamic dissipation consists of $E_{internal}$ and $E_{switching}$, and static dissipation mainly contains $E_{leakage}$.

Each $0 \rightarrow 1/1 \rightarrow 0$ transition contributes to $E_{switching}$, it is the energy dissipated for charging and discharging the capacitive load of a CMOS gate when output transitions occur. For the capacitive load C_L, the switching power is given as

$$P_{switching} = \frac{1}{2} C_L V_{DD}^2 T_r,$$

where V_{DD} is the internal operating voltage of the device and T_r is the toggle rate (The amount of $0 \rightarrow 1/1 \rightarrow 0$ transitions per unit time). $E_{internal}$ is due to the short-circuit current in a CMOS gate, i.e., the pn Junction loses its unidirectional conductivity and becomes a non-resistive circuit during a $0 \rightarrow 1/1 \rightarrow 0$ transition, and the internal power is given as

$$P_{internal} = \frac{1}{2}(P_{rise} + P_{fall})T_r,$$

where P_{rise} and P_{fall} depend on the input transition time and the total output net capacitance load.

$E_{leakage}$ is due to leakage current, and it increases with any increase in the physical time required to complete an operation, this means that if we decrease the frequency, $E_{leakage}$ will increase (the frequency is inversely proportional to the clock cycle). When the frequency is high, the value of $E_{leakage}$ is very small and negligible compared with the values of $E_{switching}$ and $E_{internal}$. Thus, our energy optimization is mainly centered on dynamic dissipation.

According to the expressions $E_{switching}$ and $E_{internal}$, both of these physical variables are related to the toggle rate. Therefore, we can optimize the energy by reducing the toggle rate.

3 Redundant Design for Reducing Glitches

The Perfect Tree energy model in [9] used the total number of perfect trees as a metric, which is negatively correlated with energy consumption, and proposed low energy variants of the stream ciphers based on this metric by changing the tap locations. In this section, an alternative idea for reducing energy consumption based on this metric is presented, we analyze the relation between the glitches and the unbalanced unrolled strand tree, and propose a redundant design to reduce glitches.

3.1 Glitches and Unbalanced Unrolled Strand Tree

According to Definition 3, a circuit strand $t_i(r)$ corresponds to an unrolled strand tree $T_i(r)$, the vertex of an unrolled strand tree $T_i(r)$ is the output value of $t_i(r)$, and the child nodes of $t_i(r)$ correspond to the inputs to $t_i(r)$. And $t_i(r)$ has five different inputs from the register storing the state values s_i or other strands.

We classify the type of input according to the distance to the register (i.e., the height of the subtree corresponding to $t_i(r)$), and different types of input have different delays. When the types of the inputs to $t_i(r)$ are different, this can make the delay unbalanced, which is the reason why glitches are produced.

According to Definition 1, each circuit strand corresponds to a combinational logic circuit module, and for `Trivium`, the expressions of all the corresponding strands are exactly the same, which means that these strands have the same gates and structure and, more importantly, the same delay. When using the Synopsys Design Compiler's Restricted directive to prevent the optimization between strands, the boundary of each strand is respected and the internal structure is maintained. Thus, we can convert the problem related to the delay about the entire combinatorial logic circuit consisting of these strands into the problem of computing the height and depth of an unrolled strand tree. For example, there are two typical computational problems related to the delay.

(a) *Estimating the delay of the circuit strand.* First, we compute the delay of a single combinational logic circuit module. Then, the height of the unrolled strand tree corresponding to the circuit strand is observed. According to the product of the above delay and height, we obtain the overall delay of the circuit strand. When two different circuit strands correspond to unrolled strand trees of the same height, it means that the two circuit strands have the same delay.

(b) *Judging whether the input delay of the circuit strand is balanced.* We observe whether all the nodes of the unrolled strand tree corresponding to the circuit strand meet the definition of the perfect m-ary tree (Definition 4). If an unrolled strand tree satisfies the definition of the perfect m-ary tree, it means that all the input delays of its corresponding circuit strand are balanced.

For combinational logic circuits, at any given moment, the output state is determined only by the combination of all the input states at the same moment, independent of the previous state of the circuit, and independent of the state at any other time. This means that for a combinational logic circuit module, if one of the inputs arrives faster or slower than the others, it may temporarily change the output of it, which is not the output after the circuit is stabilized, we claim that glitches are produced in the circuit at this time. The glitches reflect the increase in toggle rate, which leads to an increase in internal energy consumption and switching energy consumption.

In a word, the unbalanced unrolled strand tree, the glitches, and the power consumption are the same phenomenon from different perspectives. As shown in Fig. 2, $T_1(66)$ is a perfect tree, while $T_1(67)$ is not. This means that the input delay of $t_1(67)$ is unbalanced and the input $t_3(1)$ arrives slower than the other inputs, which will cause the output of a to be in an unstable state. Therefore $t_1(67)$ has more glitches, resulting in greater power consumption for $t_1(67)$ than $t_1(66)$, which is shown in Fig. 5.

3.2 Redundant Modules with the Same Delay

According to the analysis results in Section 3.1, glitches are the cause of the increased power consumption. Therefore, we can try to reduce glitches. From the viewpoint of Perfect Tree energy model, not all unrolled strand trees satisfy the definition of perfect tree. The reason for the imbalance of the unrolled strand tree is that its leaf nodes are not at the same depth, which means that the input delays of the corresponding circuit strand are inconsistent. Therefore, the problem of reducing glitches is equivalent to the problem of how to convert an unbalanced unrolled strand tree $T_i(r)$ into a perfect tree. A natural idea is to add child nodes for the leaf nodes with smaller depths until all the leaf nodes are at the same depth.

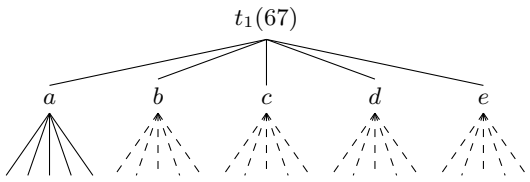


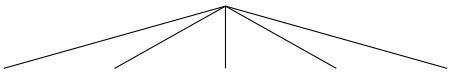
Fig. 3. The adjusted $T_1(67)$. The dashed lines connect the newly added child nodes, corresponding to the five inputs of the redundant module.

According to Definition 3, the leaf nodes of $T_i(r)$ represent the state values in the register, and the non-leaf nodes represent the output values of strands. As shown in Fig. 3, taking the circuit strand $t_1(67)$ as an example, we can adjust the unbalanced unrolled strand tree $T_1(67)$ into a perfect tree by adding child nodes, where a, b, c, d and e denote the five child nodes of $t_1(67)$. This means that the input ports corresponding to b, c, d and e change from direct connections to the states of the register into connections to the outputs of other strands (combinational logic circuit modules). For reducing glitches and improving energy efficiency, the following requirements for redundant module design are required.

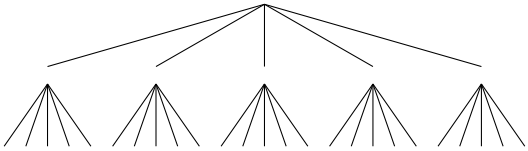
(a) The value of the signal arriving at the input port of $t_i(r)$ cannot be changed, only the arrival time is changed, i.e., the outputs of the redundant modules have to be the same as the state values in the register.
(b) The delays of the redundant modules have to be consistent with the delays of the other circuit strands, which ensures that the input delays of the circuit strands are consistent.
(c) The addition of redundant modules will bring additional leakage of power consumption, and it is important to make this part of the power consumption as small as possible.

An intuitive scheme for the above requirements is to use the structure in Fig. 1 and set its input values to ensure that its output value is the same as the

state value in the register. If we set $x_1 = s_i$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$ and $x_5 = 0$, where s_i denotes the state in the register, the output value will be $y = s_i$. We apply this redundant module to $t_1(67)$, this redundant module design scheme uses the same gates and structure as the other circuit strands, ensuring that the signal of a, b, c, d and e arrive at the input port at the same time. In addition, this scheme makes sure that the values b, c, d and e of the signal arriving at the input port is not changed. These redundant modules added between the input ports with low delays and the register states avoid the delay imbalance caused by the direct connection of the two, thus achieving the goal of reducing glitches.



(a) The subtree of height 2



(b) The subtree of height 3

Fig. 4. Optimizing an input port with lower delay corresponding to a tree of height 4 is equivalent to converting a subtree of height 2 to a subtree of height 3. This adjustment requires an additional 25 child (leaf) nodes and generates an additional 5 non-leaf nodes, implying the requirement of five redundant modules.

For other unbalanced unrolled strand trees, we can also optimize input ports with lower delay of circuit strands using redundant modules. For the input ports that need to be optimized, we propose a search algorithm in Section 4 to obtain these ports with lower delay. In general, we only adjust unbalanced unrolled strand trees of height 3. After adjusting the tree of height 3, for a tree $T_i(r)$ of height 4, the subtrees of strand $t_i(r)$ are trees of height 2 or 3 and meet the definition of the perfect tree. As shown in Fig. 4, a subtree corresponds to an input port of $t_i(r)$, so in order to adjust an input port with lower delay for a circuit strand corresponding to a tree of height 4, we need to add five additional redundant modules, while for a tree of height 3 only one redundant module is needed. And so on, adjusting the tree for heights greater than 4 requires more redundant modules. The addition of redundant modules is not cost-free, but brings an increase in area and leakage power consumption. The essence of our redundant design is to trade an increase in area for a decrease in total power consumption, and we need to make trade-offs for each hardware metric.

In addition, the corresponding input ports to be optimized are different for different circuit strands, i.e., all input ports have to be selected between the state in the register and the output of the redundant module. When considering the hardware implementation, the layout of the circuit needs to be chosen in a targeted way. VHDL's `if-generate` statement allows us to conditionally include blocks of code in the design, we can use this statement to make a selection on the input port.

4 Search Algorithm

In this section, we propose a generic search algorithm, which is used to search the input ports of the circuit strands that need to be connected to the outputs of the redundant modules. According to Definition 3, the vertex of an unrolled strand tree $T_i(r)$ is the corresponding circuit strand $t_i(r)$'s output value, and the child nodes of $t_i(r)$ correspond to the inputs to $t_i(r)$, which possibly have different delays. By filtering the ports of these circuit strands, we can determine the corresponding implementation rules and the input types of circuit strands in different classes which can ensure that the optimized unrolled strand tree will be balanced, i.e., the optimized unrolled strand tree will meet the definition of the perfect tree. In other words, the implementation rules for $t_i(r)$ indicate the type of all the input ports. Based on the type of input ports, we can determine where to add the redundant modules to reach the goal of balancing delays and reducing glitches.

Our search algorithm takes the tap locations involved in the recursive expressions (in Definition 2) of circuit strand as inputs, and outputs the implementation rule of the circuit strand. For different strands, the corresponding search algorithm can also be derived by changing the tap locations parameters.

Before introducing the search method, we give the notations used in this section as follows.

- r: The degree of unrolling.
- X_i: The tap locations involved in the recursive expressions of circuit strand.
- x_i: The input ports for circuit strand.
- $\mathcal{T}$: The set of register's tap locations.
- $\mathcal{M}$: The mapping set of register's tap locations to input ports for corresponding circuit strand.
- $\mathcal{E}$: The mapping set of circuit strands to their input ports.

Our search method is applicable to stream ciphers. In stream ciphers, their update functions can be described using a series of tap location parameters. Definition 2 gives the recursive form of the circuit strand, from the correspondence between the expression for $t_i(r)$ and the unrolled strand tree, we can observe that the shape of the tree depends on the numerical comparisons of the degree of unrolling and the tap locations involved in the recursive expressions. And according to Section 3.2, we only need to adjust the unrolled strand trees with

height 3 for the trade-off of each hardware metric. The essence of the implementation scheme of the redundant design is to optimize the input ports with lower delays, therefore, we propose Algorithm 1 to search for these input ports.

Algorithm 1: $\mathcal{E} = \texttt{SearchPorts}(\mathcal{T}, \mathcal{M})$

Input: The sets $\mathcal{T}$ and $\mathcal{M}$
Output: The set $\mathcal{E}$ of ports with redundant modules as inputs with the tap locations in $\mathcal{M}$

```
1  E ← φ
2  for i in range(1,r) do
3  |  w ← 0
4  |  foreach element X in T do
5  |  |  if i > X then
6  |  |  |  w ← w + 1
7  |  |  end
8  |  end
9  |  v ← sizeof(T)
10 |  if w ≠ 0 and w ≠ v then
11 |  |  foreach element X in T do
12 |  |  |  if i ≤ X then
13 |  |  |  |  x ← mapping(X, M)
14 |  |  |  |  E ← E ∪ {(i, x)}
15 |  |  |  end
16 |  |  end
17 |  end
18 end
19 return E
```

Algorithm 1 consists of two steps: (a) Step 1. Determine whether the delays of the inputs of $t_i(r)$ are balanced. This determination can be done by the above numerical comparison used to observe the shape of the tree, and this step corresponds to **Line 2-8** in Algorithm 1. (b) Step 2. Filter the input ports with lower delays corresponding to $t_i(r)$ with delay imbalance in Step 1, this step also uses numerical comparisons and corresponds to **Line 10-17** in Algorithm 1. Next, we explain Algorithm 1 line by line:

Line 1 Initialize the set $\mathcal{E}$ as an empty set.

Line 2 For the circuit at degree of unrolling r, there are r strands for equation t_i being iterated.

Line 3 Initialize the counter w as 0. We denote the result of a numerical comparison by 0 and 1, and accumulate the results. The total result after accumulation is related to the balance of the unrolled strand tree, i.e., the tree is balanced when the counter has a value of 0 or the number of elements in

the set $\mathcal{T}$. In a word, this counter reflects the balance of the unrolled strand tree corresponding to the circuit strand.

Line 4-8 Iterate through the five taps to get the result of the comparison with the numerical value of i, and accumulate the results.

Line 9 The `sizeof()` is used to return the number of elements in the set.

Line 10 Filter unbalanced unrolled strand trees.

Line 11-16 The input ports are classified in more detail, and the ports that need to be optimized are added to the set $\mathcal{E}$. Where `mapping()` is used to return the mapping value of an element X in the set $\mathcal{M}$.

Line 19 Return the set $\mathcal{E}$ of mappings of circuit strands to their inputs.

Taking `Trivium` as an example, as shown in Fig. 1, a circuit strand corresponds to five input ports, so we can create a mapping from the circuit strand to the input ports that need to be optimized to present a specific implementation scheme. When considering an implementation of the cipher, we connect the input ports of the mappings in the set to redundant modules.

Example: Implementation rule for $t_1(67)$. As shown in Fig. 2, $T_1(67)$ is an unbalanced tree, thus there are a portion of input ports with lower delays, and we use Algorithm 1 to filter them. It is noted that the tap locations involved in the recursive expressions of the circuit strand $t_1(67)$ are $X_1 = 66$, $X_2 = 93$, $X_3 = 91$, $X_4 = 92$ and $X_5 = 78$, where the value of X_1 is less than 67, so $w = 1$. Because $w \neq 0$ and $w \neq 5$, we further filter the input port for $t_1(67)$. Since the values of X_2, X_3, X_4 and X_5 are greater than 67, and $(X_1, x_1) \in \mathcal{M}$, $(X_2, x_2) \in \mathcal{M}$, $(X_3, x_3) \in \mathcal{M}$, $(X_4, x_4) \in \mathcal{M}$, $(X_5, x_5) \in \mathcal{M}$, so $(67, x_2) \in \mathcal{E}$, $(67, x_3) \in \mathcal{E}$, $(67, x_4) \in \mathcal{E}$ and $(67, x_5) \in \mathcal{E}$. Thus, the input ports x_2, x_3, x_4 and x_5 of $t_1(67)$ is to be connected to the redundant module.

5 Applications to Stream Ciphers

In this section, we apply the proposed model and search method in two `Trivium`-like stream ciphers: `Trivium` [13] and `Kreyvium` [10]. The most significant characteristic of `Trivium`-like ciphers is that the strands corresponding to the update functions are approximately the same. For `Trivium`-like ciphers, our redundant design model can play a positive role in energy optimization. In order to verify the effectiveness of our model and algorithm, we conduct synthetic tests and give the experimental results.

We take the tap location parameters of the stream cipher as the input of the search algorithm and get the mappings of circuit strands to its ports as the implementation scheme. Based on these mappings, we describe the layout of the circuit in VHDL, i.e., the input ports of each circuit strand should be connected to which part of the entire circuit. Then, we use Synopsys Design Compiler

to convert the circuit described by VHDL into a gate-level netlist based on the standard cell library, and the two standard cell libraries we used in our experiments are TSMC 90 nm and UMC 55 nm. We then use Synopsys VCS to run post-synthesis simulation of gate-level netlist, with the aim of collecting the switching activity of each gate of the circuit and generating SAIF files containing the information about the toggle counts. In the last step, the SAIF files are sent back to the synthesis tool together with the gate-level netlist from the initial synthesis to run power analysis.

5.1 Application to `Trivium`

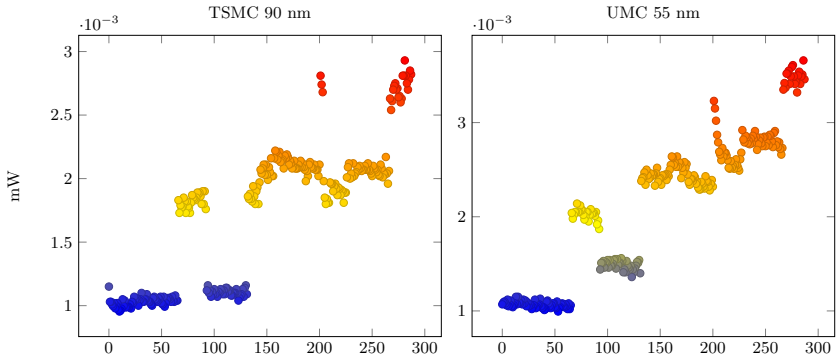
`Trivium` [13] is a hardware-oriented synchronous stream cipher, which still has a large margin of security [19,25,18,17], its update functions and their corresponding circuit strands are given in Section 2.1, and these circuit strands are exactly the same.

We use the Synopsys design compiler's restriction directive to compile the circuit to ensure that the boundaries of the circuit strands are not broken, so we can conduct a separate energy analysis for each circuit strand. Power measurements for all circuit strands $t_i(r)$ without redundant design of the two standard cell libraries (TSMC 90 nm and UMC 55 nm) are shown in Fig. 5. And we can observe that the power consumption of the circuit strands corresponding to the perfect trees are relatively low, and the power consumption of the circuit strands corresponding to the unbalanced unrolled strand trees are relatively high, where data points with colors close to blue correspond to circuit strand with low energy consumption, while data points with colors close to red are circuit strand with high energy consumption. Take the circuit strands $t_1(r)$ $(1 \leq r \leq 93)$ as examples, where the unrolled strand trees corresponding to circuit strands $t_1(r)$ for $1 \leq r \leq 66$ are perfect trees, while the unrolled strand trees corresponding to circuit strands $t_1(r)$ for $67 \leq r \leq 93$ are imbalanced trees. We can observe a sudden rise between the data points corresponding to the two parts of the circuit strands mentioned above in Fig. 5a, more specifically, there is a significant discontinuity between $t_1(66)$ and $t_1(67)$. Another phenomenon of interest is that the data points corresponding to the circuit strands do not rise monotonically as r rises, but there are sudden drops, and the data points that drop suddenly correspond to the circuit strands corresponding to the perfect trees as well, which reminds us that converting an unbalanced tree into a perfect one is an idea for optimizing power consumption, specifically, this can be achieved by introducing redundant design to the circuit.

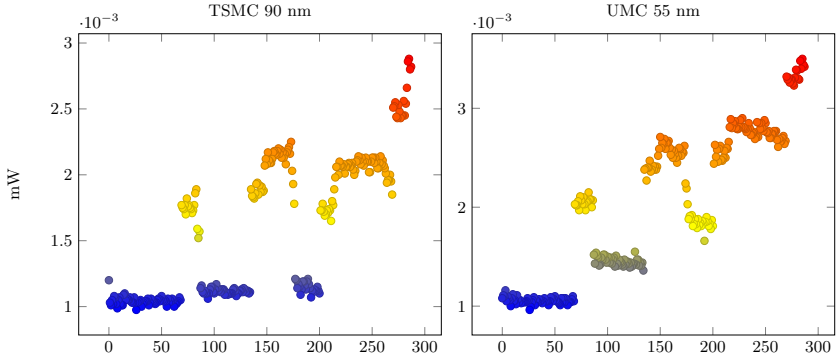
For redundant design, we also synthesize the circuit using Synopsys Design Compile's Restriction directive. From the gate-level netlist, the circuit with redundant design has some more combinational logic circuit modules (redundant modules) than the circuit without redundant design. Thanks to Restriction directive, the boundary of each circuit strand is not broken, so we can also use the power analysis tool to measure the power consumption of each circuit strand of the circuit with redundant design. Based on the power measurement figures of the circuit with and without redundant design, we can observe the role of the

redundant modules by comparing the two. Note that according to the analysis in Section 3, we only optimize for unbalanced unrolled strand trees of height 3. As shown in Fig. 5 and Fig. 6, the values of the data points corresponding to all circuit strands are reduced to different degrees, where the decrease in the values of the data point corresponding to circuit strands $t_1(r)$ for $67 \leq r \leq 93$ is particularly significant. The main reason for this significant decrease is that the power consumption of the circuit strands corresponding to the unbalanced unrolled strand trees of height 3 is reduced due to the fact that we have adjusted these trees to satisfy the definition of the perfect tree. Taking $t_1(66)$ and $t_1(67)$ as examples, in Fig. 5, there is a obvious discontinuity between the two, while in Fig. 6, there is a continuity between the two, which implies that the redundant modules play the role in reducing glitches. And compared with Fig. 5, the sudden rise that exists between circuit strands $t_1(r)$ for $1 \leq r \leq 66$ and circuit strands $t_1(r)$ for $67 \leq r \leq 93$ is eliminated.

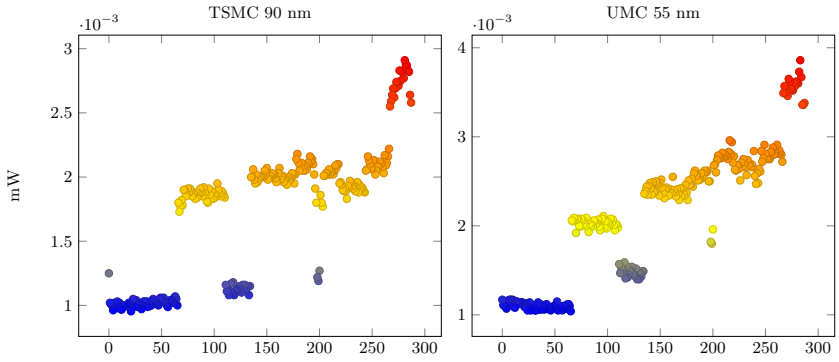
Then, we conducted energy measurements on `Trivium`'s circuit with and without redundant design, and compared the results in these two cases. In Fig. 7, we render the energy measurement of the process of encrypting 1 Mbit data results by using two standard cell libraries (TSMC 90 nm and UMC 55 nm) over 100 Mhz, and the energy consumption is computed as the product of the average power and the total time required for the encryption process. Our energy measurement figure shows the energy consumed to encrypt 1 Mbit of data at different degree of unrolling. It is noted that for lower degree of unrolling, since the number of glitches is small and the effect of redundant design is not obvious, the two lines in the figure overlap. As shown in Fig. 7, the energy measurement figure first drops sharply and then tends to be stable. Considering throughput and the impact of the degree of unrolling r on energy consumption, we take $r = 288$ as the optimal parameter. Take the circuit based on TSMC 90 nm as an example, at degree of unrolling $r = 288$, the energy consumption of the implementations with and without redundant design are 87.2 nJ/Mbit (Corresponding dynamic power consumption is 2335.5 uW and leakage power consumption is 59.2 uW) and 93.0 nJ/Mbit (Corresponding dynamic power consumption is 2497.7 uW and leakage power consumption is 55.2 uW) as shown in Fig. 7 and Table 2, respectively, which indicates a reduction in energy consumption of about 6.2%. Another example is based on UMC 55 nm, at degree of unrolling $r = 288$, the energy consumption of the implementations with and without redundant design are 93.9 nJ/Mbit (Corresponding dynamic power consumption is 2569.1 uW and leakage power consumption is 8.8 uW) and 100.4 nJ/Mbit (Corresponding dynamic power consumption is 2749.9 uW and leakage power consumption is 8.2 uW), respectively, which indicates a reduction in energy consumption of about 6.5%. According to the comparison results of dynamic and leakage power consumption in Table 2, we can find that the redundant design increases leakage power consumption, but reduces dynamic power consumption. The reason for this is that our redundant scheme reduces glitch by adding redundant circuits to balance the dynamic and leakage power consumption, so as to reduce the power consumption of the whole scheme.



(a) $t_1(r)$

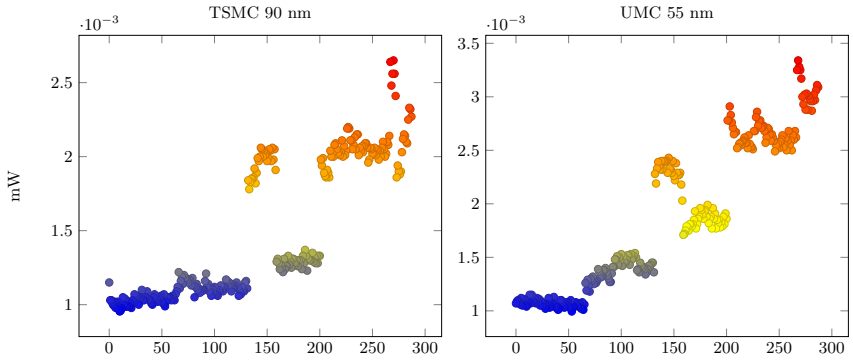


(b) $t_2(r)$

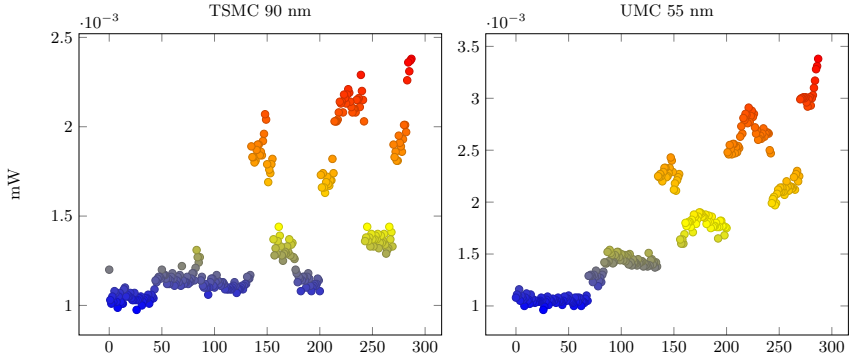


(c) $t_3(r)$

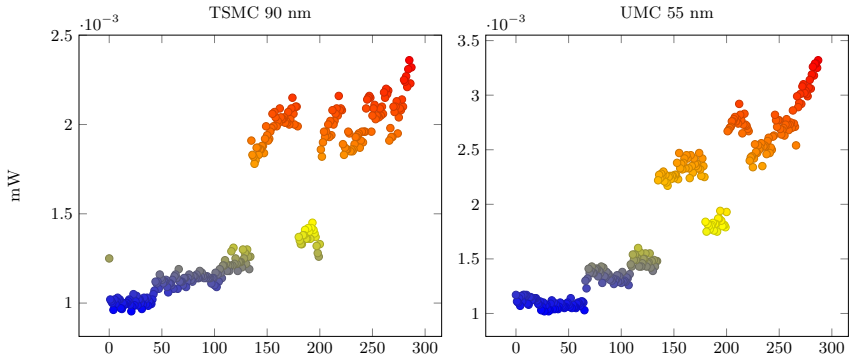
Fig. 5. `Trivium` power measurements for all the circuit strands for the TSMC 90 nm and UMC 55 nm cell libraries without redundant design.



(a) $t_1(r)$

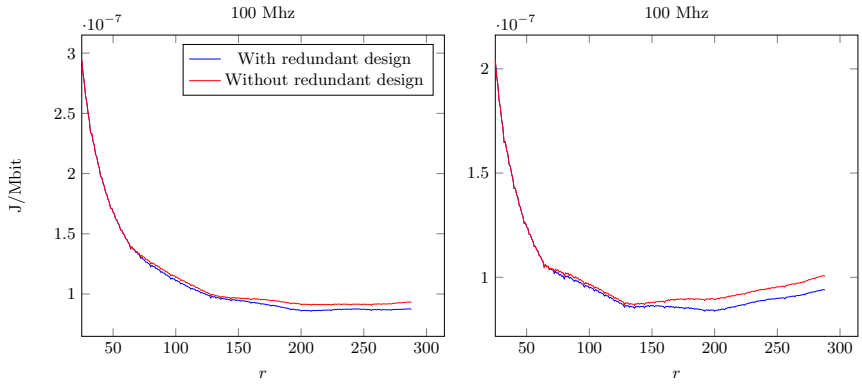


(b) $t_2(r)$



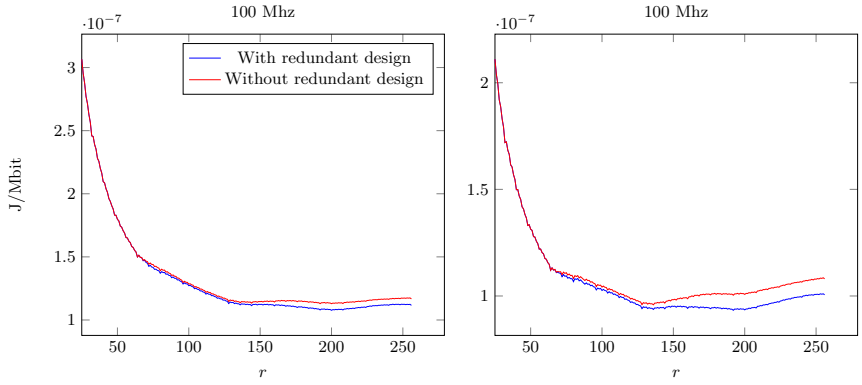
(c) $t_3(r)$

Fig. 6. `Trivium` power measurements for all the circuit strands for the TSMC 90 nm and UMC 55 nm cell libraries with redundant design.



(a) TSMC 90 nm (b) UMC 55 nm

Fig. 7. `Trivium` energy measurements for the TSMC 90 nm and UMC 55 nm cell libraries. For 100 Mhz, the implementation with redundant design is more advantageous than the implementation without redundant design.



(a) TSMC 90 nm (b) UMC 55 nm

Fig. 8. `Kreyvium` energy measurements for the TSMC 90 nm cell libraries.

5.2 Application to `Kreyvium`

`Kreyvium` [10] is a stream cipher designed to be applied to fully homomorphic encryption schemes. Compared with `Trivium`, `Kreyvium`'s security [15,16,17] is improved from 80-bit to 128-bit. Although `Kreyvium` and `Trivium` have the same structure and taps, the three circuit strands corresponding to `Kreyvium`'s update

functions are not exactly the same. According to Definition 1 and the description of `Kreyvium`, the update functions of `Kreyvium` consists of three strands

$$t_1 = s_{66} + s_{93} + (s_{91} \cdot s_{92}) + s_{171} + IV_0^*$$
$$t_2 = s_{162} + s_{177} + (s_{175} \cdot s_{176}) + s_{264}$$
$$t_3 = s_{243} + s_{288} + (s_{286} \cdot s_{287}) + s_{69} + K_0^*,$$

where IV_0^* and K_0^* denote the bits from IV and key.

Lib	Cipher	Design	Total Power (uW)	Dynamic Power (uW)	Leakage Power (uW)	Energy (nJ/Mbit)
Lib.1	`Trivium`	w/o	2552.9	2497.7	55.2	93.0
	($r = 288$)	w/	2394.7	2335.5	59.2	87.2
	`Kreyvium`	w/o	2848.4	2791.7	56.7	116.7
	($r = 256$)	w/	2723.5	2662.8	60.7	111.6
Lib.2	`Trivium`	w/o	2758.1	2749.9	8.2	100.4
	($r = 288$)	w/	2577.9	2569.1	8.8	93.9
	`Kreyvium`	w/o	2637.0	2628.7	8.3	108.1
	($r = 256$)	w/	2454.5	2445.6	8.9	100.6

Table 2. Part of power/energy measurements at 100 Mhz. w/: With redundant design, w/o: Without redundant design (corresponding to the implementation in [9]), Lib.1: TSMC 90 nm, Lib.2: UMC 55 nm

We apply the proposed model and search method to the `Kreyvium` to give a redundant design scheme and describe it using VHDL. Based on the Synopsys Design Compiler and two cell libraries (i.e., TSMC 90 nm and UMC 55 nm), we synthesise the hardware implementation scheme, and the results of our energy analysis based on encrypted 1 Mbit data are shown in Fig. 8 and Table 2. We can also observe that as the degree of unrolling r increases, the energy line first has a significant drop, and then fluctuates in a small range in Fig. 8. And as the degree of unrolling r increases, the advantages of our redundant design become more and more apparent. Considering the throughput and the impact of degree of unrolling on energy (when greater than a certain value, the impact of degree of unrolling on energy consumption is small), we take $r = 256$ as the optimal. At degree of unrolling $r = 256$, for TSMC 90 nm, the energy consumption of the implementations with and without redundant design are 111.6 nJ/Mbit (Corresponding dynamic power consumption is 2662.8 uW and leakage power consumption is 60.7 uW) and 116.7 nJ/Mbit (Corresponding dynamic power consumption is 2791.7 uW and leakage power consumption is 56.7 uW), respectively, which indicates a reduction in energy consumption of about 4.4%; for UMC 55nm, the energy consumption of the implementations with and without redundant design are 100.6 nJ/Mbit (Corresponding dynamic power consumption is 2445.6 uW and leakage power consumption is 8.9 uW) and 108.1 nJ/Mbit (Corresponding dynamic power consumption is 2628.7 uW and leakage power

consumption is 8.3 uW), respectively, which indicates a reduction in energy consumption of about 6.9%. These results indicate that our redundant design trades a portion of the increase in leakage power consumption for a reduction in dynamic power consumption, and that our design shows an advantage when the latter reduction is greater than the former increase.

6 Conclusion

In this paper, we investigated the hardware implementation of low energy stream ciphers. Based on Perfect Tree energy model, we propose a redundant design model for reducing glitches. In fact, the intuitive scheme in this paper is not the only scheme for redundant module, as long as the combined logic circuit modules that meet requirements of delay balance can be used as redundant modules. We have not yet found a better design scheme for redundant modules, so there remains an open problem of finding a better delay balancing scheme. In addition, we present a search algorithm used to return the implementation scheme. And we reimplemented the different stream ciphers using the redundant design, the experimental results show that the energy consumption of the circuit with redundant design was reduced.

From the experimental results, we can find that applying the redundant design to `Trivium`-like ciphers worked well because the strands of the `Trivium`-like cipher are approximately the same, which means that the redundant module is easier to construct. Therefore, from the perspective of energy optimization, we suggest that cipher designers consider setting the circuit strands corresponding to the update functions of the stream ciphers to be approximately the same.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (Grant No. 62272273, Grant No. 62002201, Grant No. 62032014), the National Key Research and Development Program of China (Grant No. 2018YFA0704702), and the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025).

References

1. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 411–436. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_17
2. Banik, S., Bogdanov, A., Regazzoni, F.: Exploring energy efficiency of lightweight block ciphers. In: Dunkelman, O., Keliher, L. (eds.) Selected Areas in Cryptography – SAC 2015. pp. 178–194. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_10

3. Banik, S., Mikhalev, V., Armknecht, F., Isobe, T., Meier, W., Bogdanov, A., Watanabe, Y., Regazzoni, F.: Towards low energy stream ciphers. IACR Transactions on Symmetric Cryptology **2018, Issue 2**, 1–19 (2018). https://doi.org/10.13154/tosc.v2018.i2.1-19
4. Batina, L., Das, A., Ege, B., Kavun, E.B., Mentens, N., Paar, C., Verbauwhede, I., Yalçın, T.: Dietary recommendations for lightweight block ciphers: power, energy and area analysis of recently developed architectures. In: Radio Frequency Identification: Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers 9. pp. 103–112. Springer (2013). https://doi.org/10.1007/978-3-642-41332-2_7
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Paper 2013/404 (2013), https://eprint.iacr.org/2013/404
6. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007. pp. 450–466. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
7. Caforio, A., Balli, F., Banik, S.: Energy analysis of lightweight aead circuits. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security. pp. 23–42. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_2
8. Caforio, A., Balli, F., Banik, S., Regazzoni, F.: A deeper look at the energy consumption of lightweight block ciphers. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 170–175. IEEE (2021). https://doi.org/10.23919/DATE51398.2021.9474018
9. Caforio, A., Banik, S., Todo, Y., Meier, W., Isobe, T., Liu, F., Zhang, B.: Perfect trees: Designing energy-optimal symmetric encryption primitives. IACR Transactions on Symmetric Cryptology **2021**(4), 36–73 (Dec 2021). https://doi.org/10.46586/tosc.v2021.i4.36-73
10. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In: Peyrin, T. (ed.) Fast Software Encryption. pp. 313–333. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_16
11. Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: Nessie proposal: Noekeon. In: First open NESSIE workshop. pp. 213–230 (2000), http://gro.noekeon.org/Noekeon-spec.pdf
12. Daemen, J., Rijmen, V.: The design of Rijndael, vol. 2. Springer (2002). https://doi.org/10.1007/978-3-662-60769-5
13. De Cannière, C., Preneel, B.: Trivium, pp. 244–266. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_18
14. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: Aes implementation on a grain of sand. IEE Proceedings-Information Security **152**(1), 13–20 (2005). https://doi.org/10.1049/ip-ifs:20055006
15. Hao, Y., Jiao, L., Li, C., Meier, W., Todo, Y., Wang, Q.: Links between division property and other cube attack variants. IACR Transactions on Symmetric Cryptology **2020, Issue 1**, 363–395 (2020). https://doi.org/10.13154/tosc.v2020.i1.363-395

16. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 466–495. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_17
17. He, J., Hu, K., Preneel, B., Wang, M.: Stretching cube attacks: Improved methods to recover massive superpolies. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology – ASIACRYPT 2022. pp. 537–566. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-22972-5_19
18. Hu, K., Sun, S., Todo, Y., Wang, M., Wang, Q.: Massive superpoly recovery with nested monomial predictions. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021. pp. 392–421. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_14
19. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 446–476. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_15
20. Kerckhof, S., Durvaux, F., Hocquet, C., Bol, D., Standaert, F.X.: Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint. In: Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14. pp. 390–407. Springer (2012). https://doi.org/10.1007/978-3-642-33027-8_23
21. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: A very compact and a threshold implementation of aes. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. pp. 69–88. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_6
22. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact rijndael hardware architecture with s-box optimization. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 239–254. Springer Berlin Heidelberg, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_15
23. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2011. pp. 342–357. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_23
24. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. pp. 339–354. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_22
25. Ye, C.D., Tian, T.: Algebraic method to recover superpolies in cube attacks. IET Information Security **14**(4), 430–441 (2020). https://doi.org/10.1049/iet-ifs.2019.0323