electronics

MDPI

Article

Analysis of Area-Efficiency vs. Unrolling for eSTREAM Hardware Portfolio Stream Ciphers

Fares Alharbi [1], Muhammad Khurram Hameed [2], Anusha Chowdhury [3], Ayesha Khalid [4], Anupam Chattopadhyay [5] and Ibrahim Tariq Javed [6,*]

1 Computer Science Department, Shaqra University, Shaqra 15526, Saudi Arabia; faalhrbi@su.edu.sa
2 Computer Science Department, Bahria University, Islamabad 44000, Pakistan; 01-247182-22@student.bahria.edu.pk
3 Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur 208016, India; anushac28@gmail.com
4 School of Electronics, Electrical Engineering and Computer Science, Queens University, Belfast BT7 1NN, UK; a.khalid@qub.ac.uk
5 School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Ave, Singapore 639798, Singapore; anupam@ntu.edu.sg
6 Lero-The Irish Software Research Centre, University of Limerick, Limerick V94 T9PX, Ireland
* Correspondence: Ibrahimtariq.javed@lero.ie

Received: 22 October 2020; Accepted: 12 November 2020; Published: 17 November 2020

Abstract: The demand for low resource devices has increased rapidly due to the advancements in Internet-of-things applications. These devices operate in environments that have limited resources. To ensure security, stream ciphers are implemented on hardware due to their speed and simplicity. Amongst different stream ciphers, the eSTREAM ciphers stand due to their frugal implementations. This work probes the effect of unrolling on the efficiency of eSTREAM ciphers, including Trivium, Grain (Grain 80 and Grain 128) and MICKEY (MICKEY 2.0 and MICKEY-128 2.0). It addresses the question of optimal unrolling for designing high-performance stream ciphers. The increase in the area consumption is also bench-marked. The analysis is conducted to identify efficient design principles for ciphers. We experimentally show that the resulting performance after unrolling may disagree with the theoretical prediction when the effects of technology library are considered. We report pre-layout synthesis results on 65 and 130 nm ASIC technology as well as synthesis results for Xilinx FPGA platform in support of our claim. Based on our findings, cipher design and implementation suggestions are proposed to aid hardware designers. Furthermore, we explore why and where area-efficiency for these ciphers saturate.

Keywords: cryptography; stream cipher; eSTREAM; trivium; grain; MICKEY; ASIC; FPGA; unrolling; area efficiency

1. Introduction

The rise of the Internet-of-things applications has increased the demand for low resource devices. To ensure security it is essential to protect the data generated from these devices. The devices operate in environments where resources such as energy, memory, computational power and space are very limited. Achieving high levels of security in limited resources is a challenge. While implementing a cipher on the hardware it is essential to enhance throughput and minimize the area consumption while ensuring security [1]. In cryptography, symmetric ciphers are usually used to encrypt data between two parties. Symmetric ciphers can either be block or stream ciphers. In block ciphers, the message is encrypted into fixed-size blocks whereas in stream ciphers the message is encrypted digit by digit using a pseudo-random key bitstream. On resource constraint hardware stream ciphers are usually

preferred due to their speed and simplicity [2]. eSTREAM [3] was an EU-ECRYPT project that allowed submissions of stream cipher proposals. In the four years amongst 34 proposals, 7 were included in the portfolio, 4 in software profile and 3 in the hardware profile. The hardware portfolio comprised of Grain (Now Grain Family: Grain-128AEAD [4], Grain-128a [5], Grain-128 [6], and Grain v.1 [7] ciphers for constrained computing environments) MICKEY [8] and Trivium [9] stream cipher proposals. These ciphers are considered hardware-efficient due to their low cost and high performance [10–12]. Although the security of Trivium was criticized in [13], most importantly they are considered to be secure as there are no successful cryptanalytic attacks till date on Trivium [14], MICKEY [15], and Grain [16]).

As the world is getting closer to the paradigm of pervasive computing, high-performance security for all the increased information exchange is becoming more and more challenging. This need for better performance fuels and justifies academic and industrial efforts in the direction of the design of high performance embedded Application Specific ICs (ASICs) dedicated to a certain cryptographic function. Physical constraints are important while designing ciphers for various computing environments. For efficient hardware implementations, loop unrolling is a micro-architectural configuration that replicates the main module of an iterative process by multiple modules to execute these rounds in one clock cycle, the duplication number is called the unrolling factor. As a direct consequence of unrolling, higher throughput performance is expected, along with an increase in the circuit area and the critical path of the design. In addition to achieving hardware efficiency, unrolling comes with an added benefit for cryptography cores. The unrolled hardware implementations are less vulnerable to side-channel analysis (SCA) attacks since they execute multiple rounds in every clock cycle and hence require a stronger hypothesis for cryptanalysis on multiple values due to deeper diffusion of the secret state [17].

In this paper, we analyze various possible design points in the area-performance curve for VLSI designs of stream ciphers for unrolling factors beyond what the conventional design permits. For block ciphers, the unrolling simply duplicates the round hardware and the area increase is predictable. Whereas for stream ciphers, the numbers are much more interesting since we are only replicating the boolean logic until a certain factor. Hence stream cipher unroll can promise a lightweight solution to performance boost via unrolling. With this motivation, we took up the three hardware portfolio eSTREAM ciphers for HDL implementation, along with their modified versions. In this paper, we perform unrolling by a range of factors ($L = 1$ implies no unrolling, $L = k$ implies full unrolling having loop trip count k). Starting from the synthesizable Verilog HDL and verification, we took up synthesis on various ASIC technology libraries and FPGA platform. The aim is to analyze and benchmark resource utilization, throughput performance and area efficiency due to unrolling optimization in these ciphers. Consequently, a series of interesting results were encountered that are novel and unconventional.

The paper is organized as follows: Section 2 provides the existing work related to the performance analysis of hardware stream ciphers. In section 3, the specifications and design parameters are discussed for Trivium, Grain and MICKEY ciphers. In Section 4, the unrolled hardware implementation is presented. Section 5 states various aspects of imprecise modeling of the effects of unrolling whereas Section 6 guides designers how area-efficiency grows and saturates. Finally, Section 7 concludes the paper.

2. Related Work

In cryptography, the block ciphers require huge gate footprint and memory for implementation. For this reason, they are not implemented over resource-constrained devices. On the other hand, stream ciphers can be used in resource-constraint environments due to their simplicity and throughput. After the widely used stream ciphers were proven to be insecure the eSTREAM project was developed by the European Network of Excellence in Cryptology [18] to create more efficient and robust stream cipher algorithms. There were three rounds where seven algorithms were finalized. In hardware portfolio three algorithms were selected including Grain [7], Mickey [8] and Trivium [9]. Grain family

is proposed to tackle the security issues in old stream cipher algorithms. Mickey 2.0 is designed to ensure security in resource constraint hardware. Whereas Trivium is designed to provide high performance in terms of speed and low gate count.

We discuss some relevant works to the aforementioned stream ciphers. In [1], authors discuss the efficiency of stream ciphers Trivium, Grain and Mickey. A comparison of these lightweight stream ciphers with block ciphers is presented. Block ciphers consume a large footprint of Gate Equivalents (GE) when compared to stream ciphers. For instance, the traditional Advanced Encryption Standard consumes about 2400–3500 GE whereas the lightweight stream ciphers take approximately 2000 GE. The performance of these stream cipher algorithms is further analysed in [19] by conducting simulations using Java programming language. The work suggested the adoption of Grain, Mickey and Trivium for resource-constrained hardware. Profile 2 eSTREAM is adopted in [20] using five leading Phase 2 candidates which are implemented in Spartan 3 Xilinx family and ASIC technology. This work is compared with old stream cipher algorithms based on hardware efficiency criteria. In the Grain, five parallelization factors: 1, 2, 4, 8, 16 are used in the basic architecture with 122 slices under 193 MHz the frequency. In contrast, the implementation of 16× parallel architecture shows a maximum throughput of 2480 with 6.97 Mbps/slice. In the Trivium, seven parallelization factors are used in the basic architecture with 188 slices under 201 MHz the frequency. While in 64x parallel architecture implementation, the maximum throughput 12,160 with 31.34 Mbps/slice is reached.

In the work [21], six different stream cipher algorithms are implemented for comparison using the Xilinx Spartan XC3S700A-4FG484 device. The metrics used to compare algorithms are throughput-to-area ratio and consumed area. In Grain v1, 318 slices are used under 177 MHz the frequency with 0.558 Mbps/slice throughput-to-area while in Mickey 2.0, 98 slices are used under 250 MHz the frequency with 2.55 Mbps/slice throughput-to-area. For the Trivium implementation, 149 slices are used under 326 MHz the frequency with 2.18 Mbps/slice throughput-to-area. However, the consumption of area resources was high in Grain v1 and Trivium. A new FPGA implementation approach is proposed in the work in [22]. In this paper, authors implement Mickey 2.0, Trivium and Grain v1 with two stream ciphers: Lizard and Plantlet using Xilinx's Spartan7 serial and Verilog hardware. In their findings, Trivium has the highest frequency with maximum 416 Mbps, while the Mickey 2.0 has the second-highest frequency with 384 Mbps both in basic versions. In the serial version, Trivium has the smallest consumption of area which is 133 slides, while the Grain v1 has the second smallest which is 26 slices. Trivium achieved the maximum throughput-area ratio which is 165.5 Mbps/Slice in the parallel version.

In the existing literature, the performance of hardware stream ciphers is computed and analyzed in terms of throughput, area efficiency and security. As far as we know to the best of our knowledge, there exists no work that probes the effect of loop unrolling on hardware efficiency for stream ciphers (Grain, Mickey and Trivium). Loop pipelining and loop unrolling are two methods that can improve the hardware performance by introducing parallelism in loop iterations. In loop pipelining, the concept of pipelining is introduced to allow the operations to be implemented concurrently. However, pipelining increases the complexity and cost of the hardware. On the other hand, loop unrolling introduces multiple copies of the loop body to adjust the loop iteration counter. For stream ciphers, unrolling can be a promising lightweight solution to boost the performance. Therefore, we show the effect of unrolling on the efficiency of estream hardware ciphers. The optimal unrolling for designing high-performance stream ciphers are discussed. The results can be used by hardware designers to identify design principles to achieve better performance in stream ciphers.

3. Trivium, Grain and MICKEY Specifications

In this section, we present the parameters of Grain-80, Grain-128, MICKEY 2.0, MICKEY-128 2.0 and Trivium stream ciphers. These ciphers have two phases for initialization namely, key/IV setup, randomization, followed by the key-stream generation [23]. The specifications are summarized in Table 1 and are discussed in the following subsections.

Table 1. Cipher Specifications.

Cipher	Key Size (Bits)	IV Size (Bits)	Key/IV Setup (Cycles)	Randomization (Cycles)	Initialization (Cycles)	Internal State (Bits)
Trivium	80	80	288	1152	1440	288
Grain-80	80	64	160	160	320	160
Grain-128	128	96	256	256	512	256
MICKEY	80	<=80	160	100	260	211
MICKEY-128	128	<=128	256	160	414	332

3.1. Grain

The Grain [24] family of stream ciphers are designed for hardware environments where resources such as power and memory are limited, for instance, radio-frequency identification (RFID). The Grain family has 80-bit and 128-bit variants having linear feedback shift register (LFSR), nonlinear feedback shift register (NFSR) and a nonlinear output function (NFSR). The LFSR introduces the minimum period for the key-stream whereas NFSR produces non-linearity to the cipher. The sate of NFSR is balanced by masking the output of LFSR with the input to the NFSR. The LFSR is represented as $S_t = s_t + s_{t+1}, ..., s_{t+79}$ whereas NFSR is represented by $B_t = b_t, b_{t+1}, ..., b_{t+79}$. The function $H(B_t, S_t)$ is the output key-stream bit represented by z_t.

3.1.1. Grain-80 Design Parameters

The keysize of Grain-80 [7] is 80 bits whereas the initialization vector (IV) is of 64 bits. The key and IV is loaded to shift registers $b_i = k_i, 0 \geq i \leq 79$, $s_i = IV_i, 0 \geq i \leq 63$ where the remaining bits are set as 1. The cipher is then clocked 160 times to produce the key-stream bits. The updated function of the LFSR and NFSR are defined in Equations (1) and (2), respectively.

$$s_{t+80} = s_{t+62} \oplus s_{t+51} \oplus s_{t+38} \oplus s_{t+23} \oplus s_{t+13} \oplus s_t. \tag{1}$$

$$\begin{aligned} b_{t+80} = {} & s_t \oplus b_{t+62} \oplus b_{t+60} \oplus b_{t+52} \oplus b_{t+45} \oplus b_{t+37} \oplus b_{t+33} \oplus b_{t+28} \oplus b_{t+21} \oplus b_{t+14} \oplus b_{t+9} \oplus b_t \oplus b_{t+63}b_{t+60} \\ & \oplus b_{t+37}b_{t+33} \oplus b_{t+15}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+45} \oplus b_{t+33}b_{t+28}b_{t+21} \oplus b_{t+63}b_{t+45}b_{t+28}b_{t+9} \oplus b_{t+60}b_{t+52}b_{t+37}b_{t+33} \oplus \\ & b_{t+63}b_{t+60}b_{t+21}b_{t+15} \oplus b_{t+63}b_{t+60}b_{t+52}b_{t+45}b_{t+37} \oplus b_{t+33}b_{t+28}b_{t+21}b_{t+15}b_{t+9} \oplus b_{t+52}b_{t+45}b_{t+37}b_{t+33}b_{t+28}b_{t+21}. \end{aligned} \tag{2}$$

The Boolean function $h(x)$ is defined as:

$$h(x) = h(x_0, x_1, ..., x_4) = x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \oplus x_0x_2x_3 \oplus x_0x_2x_4 \oplus x_1x_2x_4 \oplus x_2x_3x_4 \tag{3}$$

where the variables x_0, x_1, x_2, x_3 and x_4 correspond to $s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}$ and b_{t+63}, respectively. The output function $H(B_t, S_t)$ is given by

$$z_t = H(B_t, S_t) = \bigoplus_{j \in A} b_{t+j} \oplus h(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63}). \tag{4}$$

where $A = \{1, 2, 4, 10, 31, 43, 56\}$.

3.1.2. Grain-128 Design Parameters

Grain-128 [25] has a keysize of 128 bits, whereas size of the IV is specified to be 96 bits. Th initialization is done using $b_i = k_i, 0 \geq i \leq 127, s_i = IV_i, 0 \geq i \leq 95$ and loaded to NFSR and LFSR. The updated functions of LFSR and NFSR are provided by Equations (5) and (6), respectively.

$$s_{t+128} = s_t \oplus s_{t+7} \oplus s_{t+38} \oplus s_{t+70} \oplus s_{t+81} \oplus s_{t+96}. \tag{5}$$

$$b_{t+128} = s_t \oplus b_t \oplus b_{t+26} \oplus b_{t+56} \oplus b_{t+91} \oplus b_{t+96} \oplus b_{t+3}b_{t+67} \oplus b_{t+11}b_{t+13} \oplus b_{t+17}b_{t+18} \oplus b_{t+27}b_{t+59} \oplus b_{t+40}b_{t+48} \oplus b_{t+61}b_{t+65} \oplus b_{t+68}b_{t+84}. \quad (6)$$

The Boolean function $h(x)$ is defined as:

$$h(x) = h(x_0, x_1, ..., x_8) = x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8. \quad (7)$$

where the variables x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7 and x_8 correspond to b_{t+12}, s_{t+8}, s_{t+13}, s_{t+20}, b_{t+95}, s_{t+42}, s_{t+60}, s_{t+79} and s_{t+95}, respectively. The output function $H(B_t, S_t)$ is defined as

$$z_t = H(B_t, S_t) = \bigoplus_{j \in A} b_{t+j} \oplus h(x) \oplus s_{t+93}. \quad (8)$$

where $A = \{2, 15, 36, 45, 64, 73, 89\}$.

3.2. Mickey 2.0

Mickey [26] stands for Mutual Irregular Clocking KEY-stream generator is a family of stream ciphers that are implemented on resource constrained hardware. The aim of Mickey is to provide high level of security while having low complexity over hardware implementation. There are two variants of Mickey 2.0, MICKEY-80 and and MICKEY-128 having 80 bit and 128 bit key, respectively. There are two registers R and S represented by $(r_0, \ldots, r_{99})$ and $(r_0, \ldots, r_{99})$, respectively. The registers have two modes of clocking $CLOCK_R$ and $CLOCK_S$. Clocking of shift registers introduces pseudorandomness. Four control variables are used to update the contents of R and S in a non-linear manner. The hardware-oriented form of the two clock modes are presented as follows:

$CLOCK_R$:

$$r_0 \Leftarrow (r_0 \cdot CR) \oplus r_{99} \oplus IR \quad (9)$$

$$r_{i \in [1....99]} \Leftarrow \begin{cases} r_{i-1} \oplus (r_i \cdot CR) \oplus r_{99} \oplus IR & \text{if} \quad i \in RTAPS \\ r_{i-1} \oplus (r_i \cdot CR) & \text{if} \quad i \notin RATPS \end{cases} \quad (10)$$

$CLOCK_S$:

$$s_0 \Leftarrow s_{99} \oplus IS \quad (11)$$

$$s_{i \in [i.....98]} \Leftarrow s_{i-1} \oplus ((s_i \oplus COMP0_i) \cdot (s_{i+1} \oplus COMP1_i)) \oplus FB_i \cdot (s_{99} \oplus IS) \quad (12)$$

$$s_{99} \Leftarrow s_{98} \oplus ((s_{99} \oplus IS) \cdot \overline{CS} \quad (13)$$

3.3. Trivium

Trivium [27] is designed to have high speed, low gate count and reasonable implementation efficiency. Trivium consists of three shift registers of different lengths. In each round a bit is shifted to three shift registers creating internal state denoted by $S = (s_1...., s_{93}, s_{94}, ..., s_{177}, s_{178}, ..., s_{288}$. The structure of the internal state of Trivium is presented in [9]. Trivium generate 264 bits of key stream from 80 bit key and 80 bit IV. The key and IV are uploaded to the shift registers and updates 1152 times to generate key streams. The pseudo-code of the key stream is presented as follows:

for i = 1 to N **do**
$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$
$z_i \leftarrow t_1 + t_2 + t_3$
$t_1 \leftarrow t_1 + s_{91}.s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175}.s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286}.s_{287} + s_{69}$
$(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, ..., s_{92})$
$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, ..., s_{176})$
$(s_{178}, s_{279}, ..., s_{288}) \leftarrow (t_2, s_{178}, ..., s_{287})$
end for

4. Implementation

4.1. Unrolled Hardware Implementations

This section discusses the unrolling methodology for the three stream ciphers at hand. Since MICKEY is based on Jump Registers, its unrolling is different from that of Trivium and Grain which are based on feedback shift registers (FSRs). We take MICKEY 2.0 and Grain-80 as a test example. To study the effects of unrolling on implementation, we started from a basic design implementation of the cipher using VHDL and attempted unrolling. For all the designs, the natural interface resulting out of unrolling is used to avoid performance imbalance. For example, the basic Trivium design consists of two 1-bit pins for clock and reset, two 80-bit pins for Key and IV, one 1-bit output pin for a valid signal (indicating beginning of key-stream generation) and one 1-bit output pin carrying the key. For Trivium, we purposefully unrolled beyond the state update function (till 128) to study the effect on throughput and area performance.

4.2. Design Synthesis Setup

Synthesis is carried out using Synopsys Design Compiler Version H-2013.03-SP1, with *compile_ultra* option. The synthesis is driven by throughput maximization with the *max_area* constraint set to 0. The frequency is scaled up until there is a failure to meet the clock constraint. The area is reported using equivalent NAND gates. All CMOS synthesis reported has been performed targeting either

- Faraday UMC 65 nm SP/RVT Low-K process technology library. Best case condition with 1.1 V, −40 °C parameters are assumed.
- Faraday UMC 130 nm high speed FSG process technology library. Typical case condition with 1.2 V, 25 °C parameters are assumed.

4.3. Unrolling MICKEY 2.0

Figure 1 shows the architecture of MICKEY 2.0, with no unrolling (generating 1 bit/cycle of key-stream). R and S are so called Jump Registers. The registers jump to new values using jump bit (cbr, cbs). Hence unlike the conventional LFSRs, the entire R and S may be updated after a clock cycle as each bit depends on its neighboring bits for update.

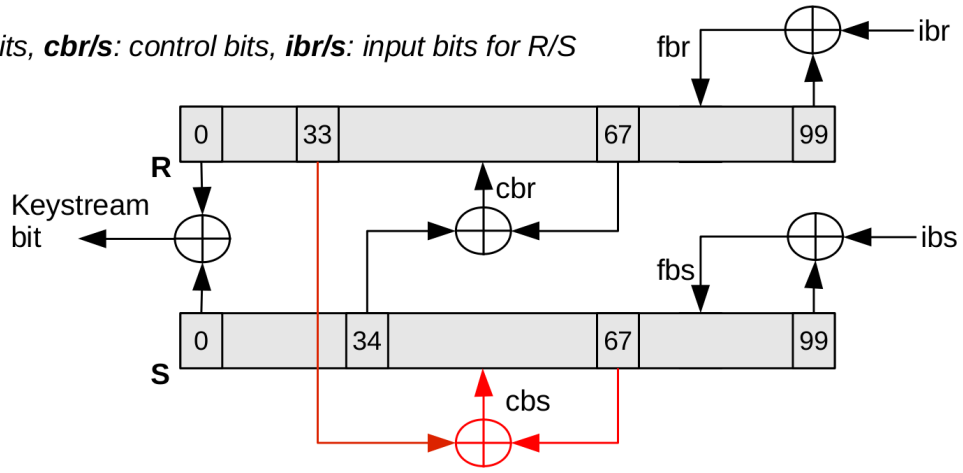

Figure 1. MICKEY 2.0 architecture with no unrolling.

For n-× unrolling we update the registers ahead n times. Since we wish to obtain n simultaneous output bits, we look-ahead for $n-1$ more rounds of feedback and control bits to generate $n-1$ more key-stream bits on the same clock edge. Theoretically, $n \geq 1$, we experimented for $n = 2, 3, 4, 8$. Figure 2 shows the 2× unrolled implementation hardware for MICKEY 2.0 with critical path highlighted. R1 and S1 are temporary buffers for holding the next iteration states of R and S, respectively.

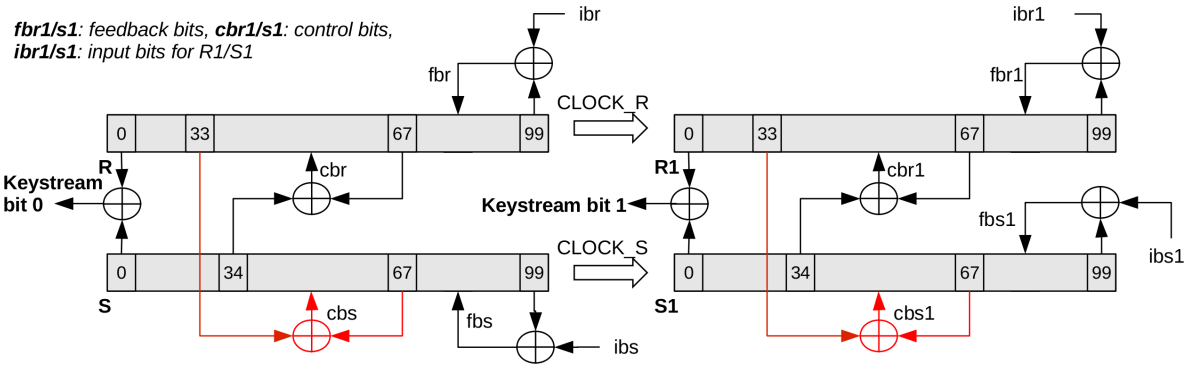

Figure 2. MICKEY 2.0 (2X) architecture (unrolled twice).

Algorithm 1 discusses a generic n-× unrolling methodology for MICKEY 2.0. For every increment in the unroll factor, future values of *cbr, cbs, fbr, fbs, ibr, ibs* are generated and the original R and S registers are updated. Every iteration of loop indexed with j in Algorithm 1, delivers a batch of n-bits of key-stream and executes till the key-stream generated bitlength l is exhausted. The inner loop indexed with i, produces one bit of key-stream. MICKEY 128 2.0 unrolling has been explained by [28], however beyond 2 times unrolling was not explored due to lowering of area efficiency. Moreover, the calculation of future control and feedback bits depended on the clocking of only a few particular bits of R and S (skipping the entire register update), which does not hold true for MICKEY 2.0.

Algorithm 1: MICKEY 2.0 n-× parallelization algorithm

Input: State R_0 and S_0 after Preclock stage
Input: Unrolling factor n, key-stream bits l
Output: n-key-stream bit per clock
Return first keybit as $R_0[0] \oplus S_0[0]$;
Set ibr_0, ibs_0 as 0.
for j=1 **to** $\lceil \frac{l}{n} \rceil$ **step** 1 **do**
  for i=0 **till** $\leq (n-1)$ **step** 1 **do**
    Calculate cbr_i, cbs_i, fbr_i, fbs_i for R_i, S_i as per CLOCK_KG.
    Call $CLOCK\_R(R_i, 0, cbr_i)$ and let R_{i+1} be the state of R_i after clocking.
    Call $CLOCK\_S(S_i, 0, cbr_i)$ and let S_{i+1} be the state of S_i after clocking.
    Return $R_{i+1}[0] \oplus S_{i+1}[0]$ as next keybit;

Unrolled Design Synthesis for MICKEY

Table 2 reports the synthesis results for synthesizing MICKEY 2.0 unrolled versions at the highest possible operating frequency for 65 nm CMOS. Deviating from the conventional wisdom suggesting unrolling always improves design efficiency, we see a contradiction. For 2-× unrolling, the Throughput Per Area Ratio (TPAR) is highest and drops for higher unrolling values. For each unrolled implementation, the critical path of the design varies. Consequently a different Boolean gate implementation is modeled based on the synthesis tool. Synthesis tools heuristics are hard to model, however we try to answer this in the next section.

Table 2. MICKEY 2.0 synthesis results for highest frequency (65 nm).

Unrolling	Area (Gates)	Frequency (GHz)	Bits/Cycle	Throughput (Gbps)	Throughput/Area (Kbps/Gates)
1	4069	1.0	1	1.0	246
2	5360	1.0	2	2.0	373
3	8498	1.0	3	3.0	353
4	13224	1.0	4	4.0	302
8	13375	0.5	8	4.0	299

MICKEY 128 2.0 has a larger key size and internal state in comparison. Table 3 gives the synthesis results for synthesizing MICKEY 128 2.0 unrolled versions at the highest possible operating frequency for 65 nm CMOS. Unlike MICKEY 2.0, the TPAR for MICKEY 128 2.0 shows are rising trend with the unroll factor. Hence when we go from rolled MICKEY to 2× unrolled MICKEY, we find that the throughput exactly doubles, whereas the area, though it increases, is less than double. Tables 4 and 5 gives the FPGA synthesis results for MICKEY 2.0 and MICKEY 128 2.0 respectively.

Table 3. MICKEY-128 2.0 synthesis results for highest frequency (65 nm).

Unrolling	Area (Gates)	Frequency (GHz)	Bits/Cycle	Throughput (Gbps)	Throughput/Area (Kbps/Gates)
1	93	1.0	1	1.0	258
2	97	1.0	2	2.0	334
3	102	1.0	3	3.0	375
4	116	1.0	4	4.0	399
8	119	1.0	8	8.0	442

Table 4. MICKEY 2.0 synthesis results (Xilinx Spartan6, device XC6SLX45).

Unrolling	Area (in Slices)			Frequency	Throughput	Throughput/Area
	Registers	LUTs	Total	(MHz)	(Mbps)	(Mbps/Slice)
1	225	672	225	370	370	1.64
2	226	805	226	370	740	3.27
3	227	1043	227	370	1110	4.89
4	228	1301	228	370	1480	6.49
8	232	2091	228	370	2960	12.98

Table 5. MICKEY-128 2.0 synthesis results (Xilinx Spartan6, device XC6SLX45).

Unrolling	Area (in Slices)			Frequency	Throughput	Throughput/Area
	Registers	LUTs	Total	(MHz)	(Mbps)	(Mbps/Slice)
1	360	410	317	370	370	1.17
2	531	461	368	370	740	2.01
3	700	629	536	370	1110	2.07
4	868	796	703	370	1480	2.11
8	1545	1469	1376	370	2960	2.15

4.4. Unrolling Grain-80/128

For Grain-80, parts of LFSR and NFSR (bit 65 til 79) do not contribute to the calculation of the next state of these registers as referred in Figure 3. Consequently, the combinational logic functions ($f(x)$, $g(x)$, $h(x)$) can be carefully replicated L times for an L times look ahead or unrolling; Grain 80 ×1, ×2, ×4, ×8 and ×16 can be implemented by calculating L functions set $f_L(x)$, $g_L(x)$, $h_L(x)$ instead of one function set $f(x)$, $g(x)$, $h(x)$ whereas $L = 2, 4, 8, 16$. For Grain-80, L can be taken as any factor of 160 up to 16 (where 160 is the state size for Grain-80, ref. Table 1). Figure 4 exhibits the doubling of combinational

logic for an unrolled ($\times 2$) version of Grain80. Unrolling beyond 16 needs duplication of LFSR and NFSR registers too other than the combinational logic functions. Consequently, no improvement is observed in area efficiency. For Grain-128 too, parts of LFSR and NFSR (bit 96 til 127) do not contribute to next state calculation. Consequently, unrolling up to Grain-128 $\times$ 32 can be calculated (L can be any factor of 256 up to 32). Beyond L = 32, no area efficiency gain is achieved, hence for Grain-80/128 we discuss the unrolling results till L = 16/32, respectively.

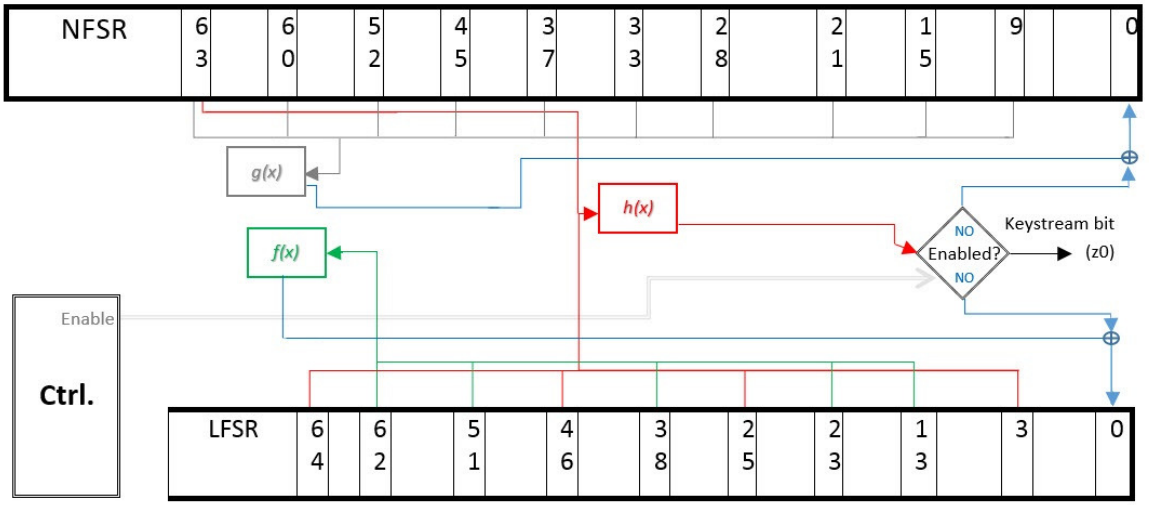

Figure 3. Grain-80 architecture with no unrolling.

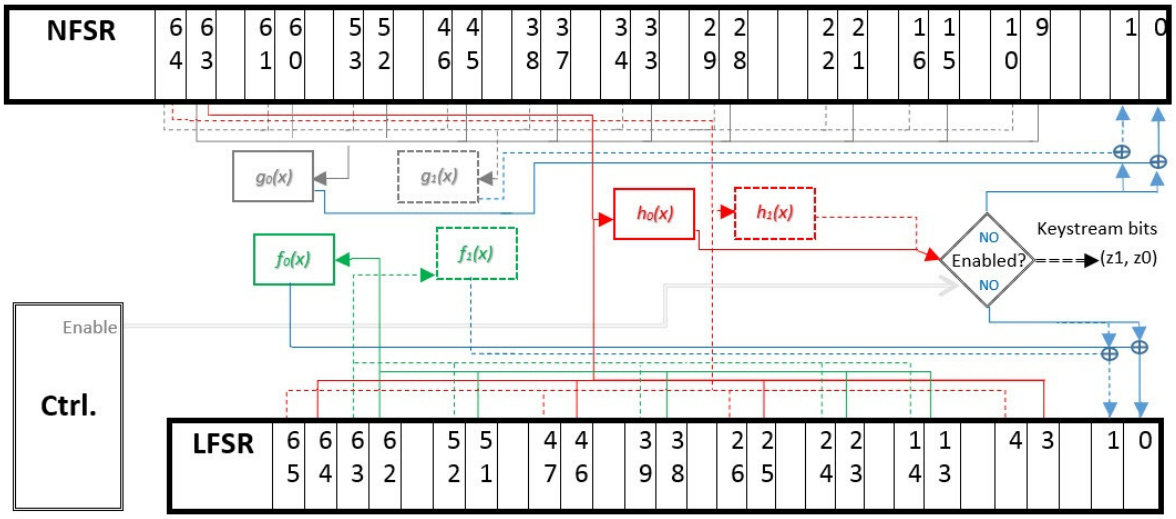

Figure 4. Grain (2X) architecture (unrolled twice).

4.4.1. Unrolled Design Synthesis for Grain-80/128

Tables 6 and 7 report the synthesis results for synthesizing Grain-80 and Grain-128 unrolled versions at the highest possible operating frequency for 65 nm CMOS. The results show an improvement in the TPAR design efficiency as the unrolling factor is increased. Consequently, Grain-80 $\times$ 16 and Grain-80 $\times$ 32 are the most efficient unrolled versions of Grain-80 and Grain-128, respectively. A similar trend of improvement in area efficiency can be seen for FPGA synthesis results as can be seen in Tables 8 and 9 for Grain-80 and Grain-128, respectively.

Table 6. Grain-80 synthesis results (65 nm).

Unrolling	Area (GE)	Frequency (GHz)	Bits/Cycle	Throughput (Gbps)	Throughput/Area (Mbps/GE)	Initialization Latency (Cycles)
1	1,175.25	1	1	1	0.85	161
2	1,281.50	1	2	2	1.56	81
4	1,497.50	1	4	4	2.67	41
8	1,955.25	1	8	8	4.09	21
16	3,047.75	1	16	16	5.25	11

Table 7. Grain-128 synthesis results (65 nm).

Unrolling	Area (Gates)	Frequency (GHz)	Bits/Cycle	Throughput (Gbps)	Throughput/Area (Mbps/Gates)	Initialization Latency (Cycles)
1	1,690.00	1	1	1	0.59	257
2	1,790.50	1	2	2	1.12	129
4	2,007.75	1	4	4	1.99	65
8	2,451.25	1	8	8	3.26	33
16	3,341.75	1	16	16	4.79	21
32	5,392.25	1	32	32	5.93	5

Table 8. Grain-80 synthesis results (Xilinx Virtex7, device XC7vx330t).

Unrolling	Freq (MHz)	Latency	Slice Reg.	Slice LUTs	Throughput (Mbps)	Throughput /Slice Reg	Throughput /Slice LUT
1	696.42	161	133	142	696.42	5.24	4.90
2	618.77	81	139	158	1392.85	10.02	8.81
4	618.77	41	151	190	2785.71	18.44	14.66
8	606.87	21	175	261	5571.42	31.84	21.35
16	595.09	11	223	403	11142.83	49.97	27.65
32	595.09	11	456	812	19043.07	41.76	23.45

Table 9. Grain-128 synthesis results (Xilinx Virtex7, device XC7vx330t).

Unrolling	Freq (MHz)	Latency	Slice Reg.	Slice LUTs	Throughput (Mbps)	Throughput /Slice Reg	Throughput /Slice LUT
1	768.76	257	198	206	768.76	3.88	3.73
2	768.76	129	204	219	1537.52	7.54	7.02
4	765.46	65	216	245	3061.85	14.17	12.49
8	760.17	33	240	297	6081.34	25.34	20.48
16	754.95	21	288	401	12079.12	41.94	30.12
32	751.94	5	384	609	24061.95	62.66	39.51
64	751.94	5	768	1218	48123.90	62.01	39.31

4.4.2. Unrolled Design Synthesis for Trivium

For trivium, we skip the unrolling methodology details since it follows similar idea as that of Grain and discuss the results directly. Tables 10 and 11 benchmark the highest possible throughput performance against area resource utilization for 65 nm and 130 nm technology libraries, respectively.

Table 12 reports the synthesis results achieved with FPGA technology. As the target board, Xilinx Spartan6, device XC6SLX45 is used. Xilinx ISE synthesis tool version 14.3 is used with "balanced" as the design goal. For all the designs, a conservative clock frequency of 100 MHz was selected, which could be achieved after placement and routing. For the unrolling factor of 64, the design could not be fit in the I/O bounds of the device.

Table 10. Trivium synthesis results (65 nm).

Unrolling	Area (Gates)	Frequency (GHz)	Bits/Cycle	Throughput (Gbps)	Throughput/Area (Gbps/KGates)	Initialization Latency (Cycles)
1	2426	6.69	1	6.69	2.76	1152
4	2433	6.28	4	25.12	10.32	288
8	2479	4.79	8	38.32	15.46	144
16	2676	2.96	16	47.36	17.69	72
32	3116	2.87	32	91.84	29.47	36
64	4023	2.89	64	184.96	45.98	18
72	4279	2.49	72	179.28	41.89	16
96	4879	1.78	96	170.88	35.02	12
128	5694	0.53	128	67.64	11.88	9

Table 11. Trivium synthesis results (130 nm).

Unrolling	Area (Gates)	Frequency (GHz)	Bits/Cycle	Throughput (Gbps)	Throughput/Area (Gbps/KGates)	Initialization Latency (Cycles)
1	2332	2.19	1	2.19	0.94	1152
4	2469	2.08	4	8.32	3.37	288
8	2652	2.0	8	16.0	6.03	144
16	2788	1.41	16	22.56	8.09	72
32	3339	1.23	32	39.36	11.79	36
64	4383	1.21	64	77.44	17.67	18
72	4711	1.09	72	78.48	16.66	16
96	5521	0.73	96	70.08	12.69	12
128	7116	0.59	128	75.52	10.61	9

Table 12. Trivium synthesis results (Xilinx Spartan6, device XC6SLX45).

Unrolling	Area (in Slices) (% Utilization)			Frequency	Throughput	Throughput/Area
	Registers	LUTs	Total	(MHz)	(Mbps)	(Mbps/Slice)
1	575 (1%)	370 (1%)	510(7%)	100	100	0.19
4	555 (1%)	379 (1%)	506(7%)	100	400	0.79
8	586 (1%)	399 (1%)	511(7%)	100	800	1.56
16	620 (1%)	436 (1%)	530(7%)	100	1600	3.02
32	621 (1%)	503 (1%)	533(7%)	100	3200	6.00

For predicting the efficiency improvement, corresponding to the unrolling factor, a popular approach is to count the number of Boolean functions required in the unrolled implementation. Thenceforth, the gate count is derived by approximating each Boolean function with an equivalent number of NAND gates. This is also presented in the Trivium specification [9] and reproduced here (Table 13) for convenience of reading. Evidently, this approach is naive and cannot capture the effects of technology reliably. This is reflected in the experimental results presented in the Tables 10–12. For each unrolling factor, we compute the predicted throughput/area by assuming a doubling of throughput.

Table 13. Theoretical Prediction of the Effect of Unrolling on Trivium [9].

Unrolling	1×	4×	8×	16×	32×	64×
Flip-Flops:	288	288	288	288	288	288
AND gates:	3	12	24	48	96	192
XOR gates:	11	44	88	176	352	704
NAND gate count:	3488	3584	3712	3968	4480	5504
Throughput/Area:	0.28	1.11	2.15	4.03	7.14	11.63

In order to compare, how the theoretical prediction of the area-efficiency matches with the practical results, we scaled the area-efficiency of the original model in each case to a value of 1 and computed the relative area-efficiency increase for all the following points using the formula $Eff(A)_{rel} = \frac{Eff(A)_{i+1}}{Eff(A)_i}$, where i indicates an element in the set of unrolling factors. The resulting graph is shown in the Figure 5. The dotted line indicates a relative increase of 1. As long as the unrolling provides a relative increase more than 1, it is justified for an area-efficient design.

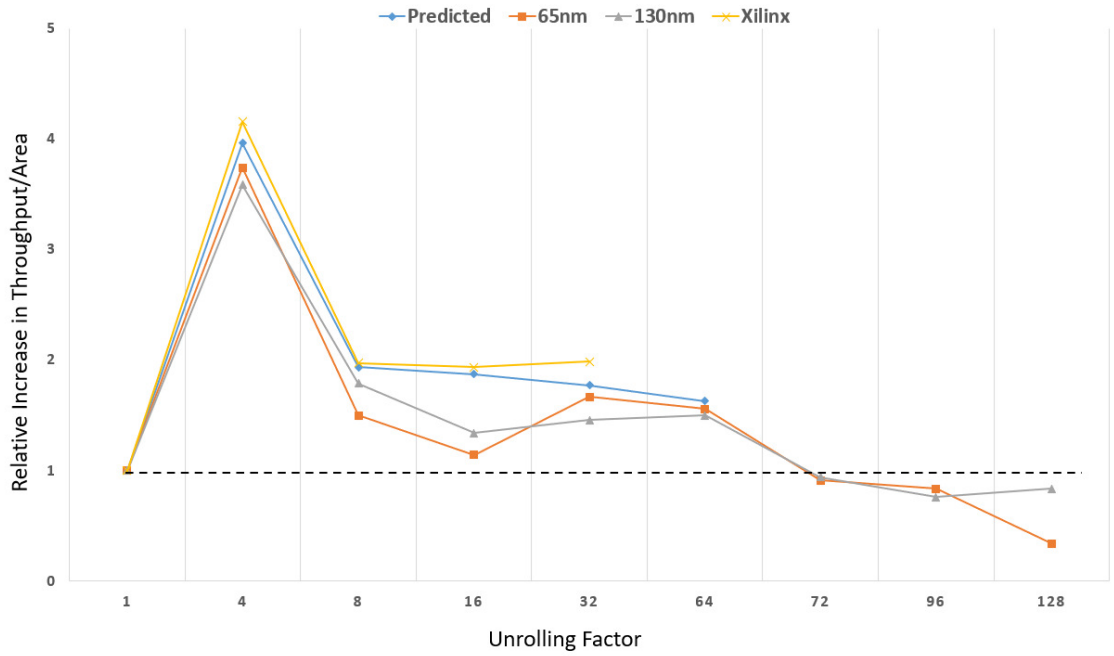

Figure 5. Relative increase in throughput/area vs. unrolling factor: Trivium.

It can be observed from the Figure 5 that, the predicted gain in area-efficiency matches well with the FPGA target technology. Whereas, for ASIC technology library, there are deviations from the prediction. On a finer study, it turns out that the theoretical prediction is too optimistic compared to the ASIC technology library and too pessimistic compared to the FPGA technology library. In the following, we attempt to explain (some of) the hidden factors that contribute to the mismatch between a theoretical model and practical results. It is worth noting that the unrolling factor of 64 provides the highest area-efficiency, it is not significantly different from the unrolling factor of 72 for ASIC technologies. It is, therefore, advisable to explore unrolling beyond the state update function.

5. Imprecise Modelling of the Effects of Unrolling

5.1. Effect of Cell Selection

Each technology library comes with a rich set of logic gates for allowing an efficient implementation. Depending on the user-driven synthesis constraints, a particular cell from the library is selected for implementing a Boolean logic. This choice is driven, internally, by synthesis heuristics, and therefore, hard to model. However, once the design is mapped to a complete gate-level implementation, it is possible to reason, why a particular selection is chosen. We provide a simple example from 130 nm technology library, when we move from the basic Trivium implementation to a quadruple-unrolled version.

For the basic Trivium implementation, only one read output from each of the state registers is needed. For unrolled versions, the number of outputs for several registers increases. This leads to a selection of different FlipFlop (F/F), as shown in the Figure 6. The upper part of the figure contains a

snapshot of the original Trivium implementation and the lower part shows the quadruple-unrolled version of the same. The corresponding F/F with one and two outputs are also shown. In this example, four storages, s_{89}, s_{90}, s_{91} and s_{92} require at least two outputs. The dual-output F/F incurs more delay and area. For the design with 64 unrolling factor, there still remains a few states with single output, e.g., s_{192}. On the other hand, there are storages with as many as five outputs, e.g., s_{225}.

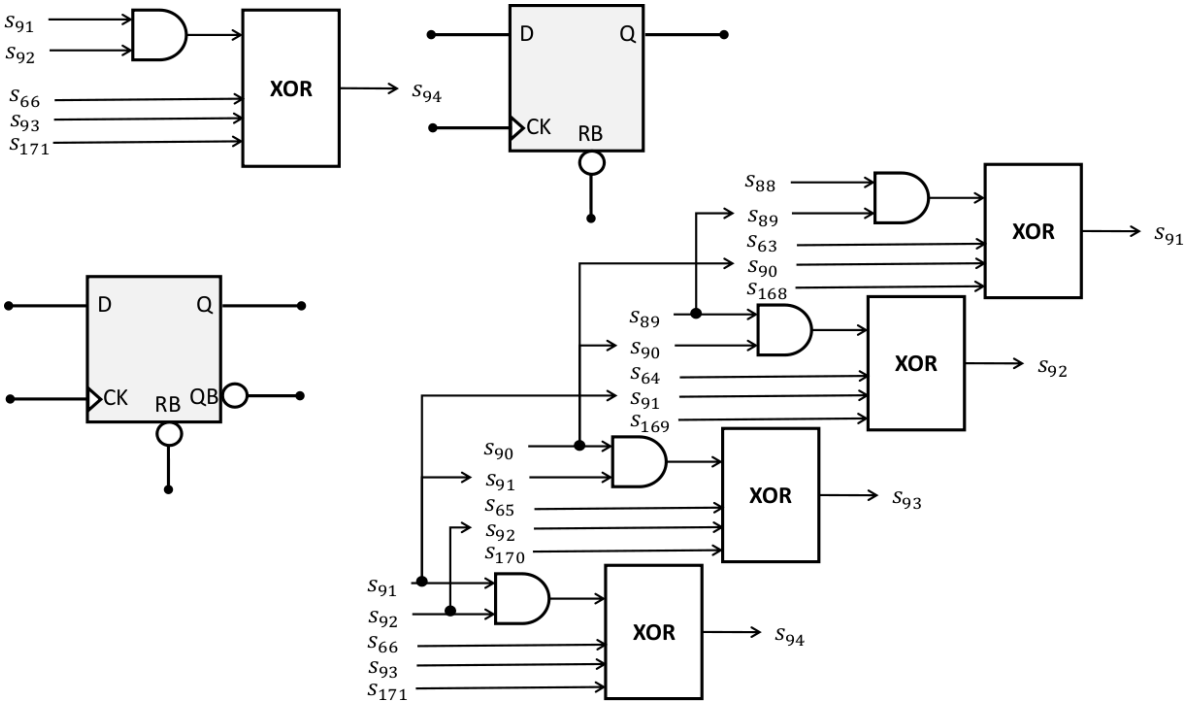

Figure 6. Effect of cell selection: multiple output.

5.2. Effect of Increased Driving Load

A logic circuit is characterized to have a drive strength of $n\times$ if it is able to drive an output capacitive load of $n \cdot C$ with the same rise and fall delays as compared to a reference inverter driving a load capacitance of C. The same logic implementation in a technology library often has many implementation variations depending on the output capacitive load. This allows the synthesis tool (or physical designer) to select a small-area implementation for non-critical path as well as to adopt multiple threshold voltages for cells with low drive strength when low-power design is targeted. The output load depends on load capacitance the following logic cell as well as on the number of fanouts. For higher output loads, bigger cells need to be selected from the technology library, which incurs higher delay overheads. This effect becomes prominent with increasing unrolling factors, due to sharing of combinational logic and increased fanouts of the storage cells.

5.3. Effect of Interconnect and Buffer Insertion

In submicron process technologies, the effect of interconnect delay is more dominant compared to the logic/transistor delay. This problem is aggravated in designs with long interconnect wires. To the first order, interconnect delay is proportional to the square of the interconnect length. To address this issue, buffers are inserted in a wire, effectively generating multiple wire segments and obtaining a linear delay increase with increasing wire length [29]. Buffers are also inserted to address increasing fanout load. While a pre-layout synthesis result does not accurately model the effect of buffer insertion, yet this is reflected in the experimental results since, the synthesis is run in topographical mode. In this mode, a virtual layout of the design is created for accurate net delay and capacitance prediction. The effect of buffer insertion is observed in the reported critical path for designs with more unrolling.

Other than these factors of imprecise modeling, the difference in ASIC vs. FPGA technology also contributes to it. Yet another factor is the design constraints and synthesis options, that can lead to hugely varying results even if the synthesis is targeted for the same device.

6. Growth and Saturation of Area-Efficiency: Lessons for Designer

The demand of physical constraints play an important role in designing ciphers today, as the security concerns become ubiquitous ranging from packet encryption in large-scale servers to security protocols in ultralight wireless sensor nodes. The prime intention behind design unrolling is to increase the area-efficiency. Consider a cryptographic accelerator with area A and a throughput performance of T. By simply deploying two independent accelerators, one doubles the area to $2A$ and hopes to get a throughput of $2T$. However, this throughput improvement is not unconditional. In particular, it is assumed that the two accelerators will be provided with independent message streams, essentially requiring more ports to the system. Even under the ideal condition, where both area and throughput doubles, the area-efficiency of the overall system remains $\frac{T}{A}$. A design, which can be unrolled, allows to increase area-efficiency beyond the basic design. This is a key optimization technique for area-constrained cipher implementations.

6.1. Growth of Area-Efficiency

The area-efficiency stops growing around the unrolling factor. This is observed to be between 64 and 72 for Trivium. However, and more importantly, this observation is valid only when we do a throughput-driven synthesis. On the other hand, if we set a low frequency constraint (e.g., for low-power applications) then, unrolling beyond the state update function is easily justifiable. An experiment is performed for the Trivium implementation with 130 nm technology. For both the factors of 64 and 128, the design was synthesized with a target clock of 500 MHz. The resulting area-efficiency, in terms of Gbps/KGates, are 7.41 and 10.05 for 64 and 128-unrolled designs, respectively. In other words, the area-efficiency keeps growing beyond the conventional highest unroll factor of 64 for some synthesis constraints and some target technologies. To allow unrolling beyond the state update function, the cipher needs to have a "simple" update function. For example, unrolling beyond the state update failed to increase the area-efficiency when studied in the case of SNOW 3G, ZUC and RC4 [30].

6.2. Saturation of Area-Efficiency

Even before the area-efficiency growth stops at a certain point, the value saturates, i.e., one obtains less return in throughput improvement for a given area increase. This can be quantified by considering a primitive model with combinational logic (C), storage area (S), total area ($A = C + S$) and throughput (T). For an unroll factor of n, the resulting area is $A_n = S + n \cdot C$ and the new throughput T_n is $n \cdot T$. Hence, the area-efficiency for an unroll factor of n is $\frac{T_n}{A_n} = \frac{T}{C+\frac{S}{n}}$. Therefore, the growth of area-efficiency compared to the basic model is

$$Eff(A)_{rel} = \frac{\frac{C}{S} + 1}{\frac{C}{S} + \frac{1}{n}} \tag{14}$$

Equation (14) clearly shows that unrolling leads to a growing area-efficiency proportional to n when $\frac{C}{S} \to 0$. So, the area proportion of combinational logic in the overall cipher should be as low as possible. This effect is illustrated in Figure 7, where different curves for absolute area-efficiency corresponds to different $\frac{C}{S}$ values, ranging from 10% to 50%. For a lower value of $\frac{C}{S}$, the growth is sharper and saturates late, at high unroll factor.

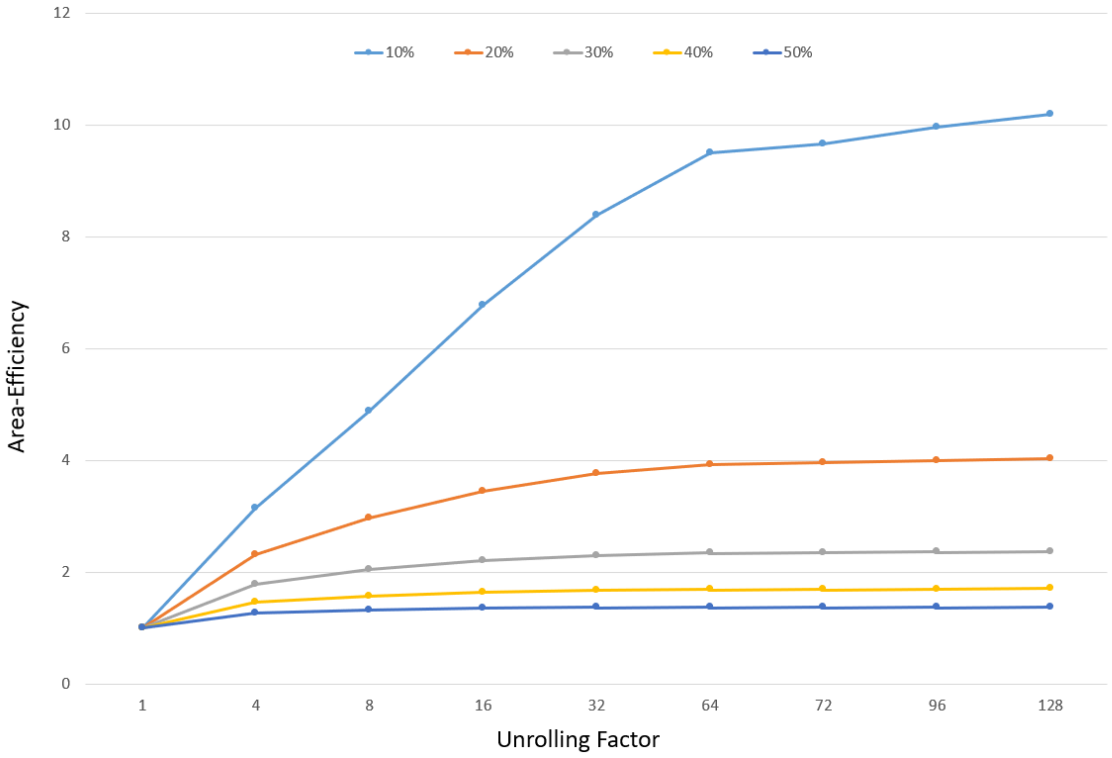

Figure 7. Area-efficiency vs. unrolling factor for different $\frac{C}{S}$.

7. Conclusions

In this paper, we benchmark Trivium, Grain and MICKEY VLSI performance with various unroll factors on 65 nm, 130 nm CMOS technology and Xilinx FPGA. The theory-vs-practical results for unrolling are presented. The unroll factors beyond conventional maximum limits is extended to get higher area efficiency. For Trivium, we compared the deviation of throughput efficiency against the theoretical prediction and explained some factors for the mismatch. The reason and cutoff points for the area-efficiency saturation due to unrolling is explored. We outlined how area-efficiency varies with unrolling for MICKEY, Grain and Trivium. A direct along with n-way parallelized designs is implemented to illustrate the feasibility of maintaining the high-throughput, low-resource and high-security qualities of the cipher. Efficiency of a design is estimated by calculating ratio of throughput to area. The results presented in this paper can be used by cryptographers to choose the most efficient design. As our future work, we intend to extend this work in two major directions: First, extend the unrolling design methodology to support other prominent stream ciphers and benchmark its effect in terms of exploring critical design points in the area vs. throughput design space. Secondly, support for the countermeasures against the side channel analysis attacks. Moreover, we also intend to explore the affect of loop pipelining on stream ciphers performance.

Author Contributions: Individual contributions of authors are as follows: methodology, M.K.H., A.C. (Anusha Chowdhury) and A.C. (Anupam Chattopadhyay); validation, M.K.H. and A.K.; resources, A.C. (Anupam Chattopadhyay) and F.A.; writing—review and editing, I.T.J., F.A. and A.K.; supervision, F.A. and I.T.J.; project administration, F.A.; funding acquisition; F.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Shaqra University, Shaqra 15526, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Philip, M.A.; Vaithiyanathan. A survey on lightweight ciphers for IoT devices. In Proceedings of the 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), Kollam, India, 21–23 December 2017; pp. 1–4.
2. Jiao, L.; Hao, Y.; Feng, D. Stream cipher designs: A review. *Sci. China Inf. Sci.* **2020**, *63*, 1–25.
3. Robshaw, M.; Billet, O. *New sTream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4986.
4. Hell, M.; Johansson, T.; Meier, W.; Sönnerup, J.; Yoshida, H. An AEAD variant of the grain stream cipher. In *International Conference on Codes, Cryptology, and Information Security*; Springer: Cham, Switzerland, 2019; pp. 55–71.
5. Agren, M.; Hell, M.; Johansson, T.; Meier, W. Grain-128a: A new version of Grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput.* **2011**, *5*, 48–59.
6. Hell, M.; Johansson, T.; Maximov, A.; Meier, W. A stream cipher proposal: Grain-128. In Proceedings of the 2006 IEEE International Symposium on Information Theory, Seattle, WA, USA, 9–14 July 2006; pp. 1614–1618.
7. Hell, M.; Johansson, T.; Meier, W. Grain: A stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.* **2007**, *2*, 86–93.
8. Babbage, S.; Dodd, M. The MICKEY stream ciphers. In *New Stream Cipher Designs*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 191–209.
9. De Canniere, C.; Preneel, B. Trivium specifications. In *eSTREAM, ECRYPT Stream Cipher Project*; Citeseer: New York, NY, USA, 2005.
10. Hwang, D.; Chaney, M.; Karanam, S.; Ton, N.; Gaj, K. Comparison of FPGA-targeted hardware implementations of eSTREAM stream cipher candidates. In Proceedings of the ECRYPT State of the Art of Stream Ciphers, Lausanne, Switzerland, 13–14 February 2008; pp. 151–162.
11. Good, T.; Chelton, W.; Benaissa, M. Review of stream cipher candidates from a low resource hardware perspective. In Proceedings of the SASC 2006 Stream Ciphers Revisit, Leuven, Belgium, 2–3 February 2006; pp. 163–173.
12. Banik, S.; Mikhalev, V.; Armknecht, F.; Isobe, T.; Meier, W.; Bogdanov, A.; Watanabe, Y.; Regazzoni, F. Towards low energy stream ciphers. *IACR Trans. Symmetric Cryptol.* **2018**, *2018*, 1–19.
13. Potestad-Ordóñez, F.E.; Jiménez-Fernández, C.J.; Valencia-Barrero, M. Experimental and timing analysis comparison of FPGA trivium implementations and their vulnerability to clock fault injection. In Proceedings of the 2016 Conference on Design of Circuits and Integrated Systems (DCIS), Granada, Spain, 23–25 November 2016; pp. 1–6.
14. Knellwolf, S.; Meier, W.; Naya-Plasencia, M. Conditional differential cryptanalysis of trivium and KATAN. In *International Workshop on Selected Areas in Cryptography*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 200–212.
15. Ding, L.; Guan, J. Cryptanalysis of MICKEY family of stream ciphers. *Secur. Commun. Netw.* **2013**, *6*, 936–941.
16. Hridya, P.; Jose, J. Cryptanalysis of the Grain Family of Ciphers: A Review. In Proceedings of the 2019 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 4–6 April 2019; pp. 0892–0897.
17. Bhasin, S.; Guilley, S.; Sauvage, L.; Danger, J.L. Unrolling cryptographic circuits: A simple countermeasure against side-channel attacks. In *Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 195–207.
18. eSTREAM. ECRYPT Stream Cipher Project. 2009. Available online: https://www.ecrypt.eu.org/stream/ (accessed on 15 October 2020).
19. Kuznetsov, A.; Frolenko, V.; Eremin, E.; Zavgorodnia, O. Research of cross-platform stream symmetric ciphers implementation. In Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kiev, Ukraine, 24–27 May 2018; pp. 30.
20. Gaj, K.; Southern, G.; Bachimanchi, R. Comparison of hardware performance of selected Phase II eSTREAM candidates. In Proceedings of the State Art Stream Ciphers Workshop (SASC), Bochum, Germany, 31 January–1 February 2007.
21. Kitsos, P.; Sklavos, N.; Provelengios, G.; Skodras, A.N. FPGA-based performance analysis of stream ciphers ZUC, Snow3g, Grain V1, Mickey V2, Trivium and E0. *Microprocess. Microsyst.* **2013**, *37*, 235–245.

22. Li, B.; Liu, M.; Lin, D. FPGA implementations of Grain v1, Mickey 2.0, Trivium, Lizard and Plantlet. *Microprocess. Microsyst.* **2020**, *78*, 103210, doi:10.1016/j.micpro.2020.103210.
23. Galanis, M.D.; Kitsos, P.; Kostopoulos, G.; Sklavos, N.; Koufopavlou, O.; Goutis, C.E. Comparison of the hardware architectures and FPGA implementations of stream ciphers. In Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, Tel Aviv, Israel, 15 December 2004; pp. 571–574.
24. Hell, M.; Johansson, T.; Maximov, A.; Meier, W. The Grain Family of Stream Ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists*; Robshaw, M.; Billet, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 179–190, doi:10.1007/978-3-540-68351-3_14.
25. Hell, M. Grain-128 AEAD-A Lightweight AEAD sTream Cipher Cover Sheet. Available online:https://csrc.nist.gov/projects/lightweight-cryptography (accessed on 5 October 2020).
26. Babbage, S.; Dodd, M. The stream cipher MICKEY 2.0. In *ECRYPT Stream Cipher*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 191–209.
27. De Cannière, C.; Preneel, B. Trivium. In *New Stream Cipher Designs: The eSTREAM Finalists*; Robshaw, M., Billet, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 244–266, doi:10.1007/978-3-540-68351-3_18.
28. Stefan, D.; Mitchell, C. On the parallelization of the MICKEY-128 2.0 stream cipher. In Proceedings of the ECRYPT State of the Art of Stream Ciphers, Lausanne, Switzerland, 13–14 February 2008; pp. 175–185.
29. Cong, J.; He, L.; Koh, C.; Madden, P. Interconnect Optimization for High Performance VLSI Design. *Integr. J.* **1996**, *21*, 1–94.
30. Gupta, S.S.; Chattopadhyay, A.; Khalid, A. Designing integrated accelerator for stream ciphers with structural similarities. *Cryptogr. Commun.* **2013**, *5*, 19–47.