School of Electrical, Electronic and Computer Engineering

Design and security evaluation of balanced 1-of-n circuits

F. Burns, A. Bystrov, A. Koelmans and A. Yakovlev

Technical Report Series

NCL-EECE-MSD-TR-2010-152

Mar 2010

Contact:
f.p.burns@ncl.ac.uk
a.bystrov@ncl.ac.uk
albert.koelmans@newcastle.ac.uk
Alex.Yakovlev@ncl.ac.uk

Supported by: EPSRC grants GR/S81421 and EP/F016786/1

NCL-EECE-MSD-TR-2010-152

School of Electrical, Electronic and Computer Engineering,
Merz Court,
University of Newcastle Upon Tyne,
Newcastle Upon Tyne, NE1 7RU, UK

http://async.org.uk/

Design and security evaluation of balanced 1-of-n circuits

F. Burns, A. Bystrov, A. Koelmans and A. Yakovlev

Abstract

A new balanced library is presented which consists of novel mixed 1-of-2 and 1-of-4 components based on N-nary logic. Cryptographic circuit specifications are refined and passed to optimization and mapping tools for mapping to a library of power-balanced components. Logic optimization tools are then applied to generate secure synchronous circuits for layout generation. This paper presents a new technique for evaluating the security of such circuits in particular those which offer a higher level of protection. A security metric is introduced which is based on the common selection function that is widely used in DPA attacks and a correlation measure similar to the one used in CPA attacks. This is used to compare the security level for these kinds of balanced circuits that are more difficult to attack. The paper shows that the circuits generated are more efficient and can offer a higher level of security than those generated by alternative techniques.

1 Introduction

Cryptographic devices are becoming increasingly ubiquitous and complex, and in order to satisfy the high throughput requirements of many applications, they are often implemented by means of VLSI devices (crypto-accelerators) [1]. The high complexity of such implementations raises concerns regarding their reliability. Attacks against such cryptographic devices include those based on side-channel [2] and those that are fault-based [3]. The side-channel attacks exploit easily accessible information like power consumption, running time, input-output behaviour under malfunctions [4], and can be mounted by anyone using low-cost equipment.

Attacks that are side-channel based have been known to exhibit a high degree of success. One of the most widely used attacks is Differential Power Analysis (DPA) which was introduced by Kocher [2]. Here he desribes a successful attack on the DES cipher. DPA attacks are successfully based on the correlation between a set of power measurments and a selection function related to the key. The number of ciphertexts used in the attack varies and depends on the nature of the attack. More recent attacks based on DPA have been carried out against the AES standard cipher [5]. Another similar type of attack that is widely used is Correlation Power Analysis (CPA) which was introduced by Brier [6]. Here the attack model used is based on a direct correlation with the power trace.

Some attempts have been made to counter side-channel attacks using balanced circuits. Recent work towards creating a VLSI design flow for side-channel attack resistant circuits was carried

out in [7]. This was primarily applied at the lower level using a library of differential balanced cells [8] and was targetted towards power-balanced synchronous circuits. In [9] they investigated side-channel attacks at the lower level and concluded the best solution to power analysis is to embed countermeasures into logic cells [10] to reduce leakage information. Here a novel logic style is proposed which relies on the use of signals with three different possible states operating with a power consumption independent of both the logic values and the sequence of data.

An alternative technique uses dual-rail [11] where the logic of dual-rail provides the security because of the 1-of-2 encoding used. Dual-rail provides in addition to security against side-channel attacks [12] a level of protection at the fault-level [13] as well. Unfortunately it suffers from significant overheads in area and power. A demonstrator chip based on an alternating spacer protocol attempts to overcome this problem by utilizing a low-overhead dual-rail logic style [14]. Attempts at using dual-rail for asynchronous solutions has so far proved to be useful but unfortunately they tend to exhibit overheads which lead to inefficiencies.

An efficient security design flow (partially automatic) is presented here centered around Dynamic logic. The inefficiency problem of using dual-rail for either synchonous or asynchronous power-balanced implementations can be resolved by steering towards alternative 1-of-n circuits [15] which use dynamic logic [16]. The aim here is to steer away from standard dual-rail circuits and move towards general N-nary 1-of-n circuits which use in addition to 1-of-2 circuits, direct mapping to 1-of-4 circuits [17]. Using dynamic logic it is possible to attain significant improvements in area and speed [18][19]. Another advantage of using 1-of-n as opposed to dual-rail encoding is that more complex codes offer the possibility of better energy efficiency [20].

The design flow that is presented here is focussed on cryptographic circuit generation and in particular Galois field implementation. A security specification is first entered using SystemC which undergoes various levels of refinement before sub-modules are generated. A novel logic library of specially designed power-balanced N-nary 1-of-n gates is provided. The dedicated dynamic gates from the new library, i.e. implicit-exor, exorhalf-implicit, etc., have been carefully designed to help reduce the area, delay and security of the implementation. The process is partially automated using a mapping algorithm to generate an optimal solution.

Encoded balanced circuits, such as the above, are less prone to attack as there is less side-channel leakage. Correspondingly this makes security evaluation of these types of protected circuits less easy to evaluate. To overcome this a metric is provided which provides a measure of security based on the measure of degree by which the circuit is attackable. This is achieved using a measure which is derived from a combination of DPA and CPA measurements. This provides a measure of correlation which corresponds to the most likely points of attack. The higher the value provided by the metric the more likely a successful attack will be. The lower the value the less likely an attack will be successful. The metric enables us to evaluate the security of the generated circuits.

The remainder of this paper is organised as follows: in section 2 we introduce our security metric; in section 3 we present the new library of 1-of-n encoded circuits; in section 4 we present our security design flow in section 5 we provide results and in section 6 conclusions are provided.

2 Power Analysis and Security metrics

DPA is a side-channel attack which involves statistically analyzing power consumption measurements from a cryptosystem. In digital circuits there are effects correlated to data values being

manipulated. These variations tend to be smaller and are sometimes overshadowed by measurement errors and other noise. In such cases, it is still often possible to break the system using statistical functions tailored to the target algorithm.

The DPA attack exploits biases in varying power consumption of a circuit while performing operations using secret keys. The attack may be based on a single bit or multiple bits such as when using the Hamming weight. Using DPA, an adversary can obtain secret keys by analyzing power consumption measurements from multiple cryptographic operations performed by a vulnerable smart card or other device.

In DPA normally a selection function is chosen $D(C, b, K_i)$ which computes the value of bit $0 \leq b \leq n$ for some target value of n-bits for ciphertext C, where the i bits of key K_i entering the function corresponding to bit b are represented by $0 \leq K_i \leq 2^n$. It is important to note that if K_i is incorrect, evaluating $D(C, b, K_i)$ will yield the correct value for bit b with probability $P \approx 1/2$ for each ciphertext. If K_i is correct, however, the computed value for $D(C, b, K_i)$ will equal the actual value of the target bit b with probability 1.

To implement a DPA attack, an attacker first observes m encryption operations and captures power traces $T_{1..m}[1..k]$ containing k samples each. In addition, the attacker records the ciphertexts $C_{1..m}$. No knowledge of the plaintext is required.

DPA analysis uses power consumption measurements to determine whether a key block guess K_i is correct. The attacker computes a k-sample differential trace $\triangle_D[1..k]$ by finding the difference between the average of the traces for which $D(C, b, K_i)$ is one and the average of the traces for which $D(C, b, K_i)$ is zero. Thus $\triangle_D[1..k]$ is the average over $C_{1..m}$ of the effect due to the value represented by the selection function D on the power consumption measurements at point j as defined in [2].

Power attacks are often unwieldy and very time consuming. In addition it may not be possible using an attack to find the key. A gauge of how vulnerable the circuit is would be often useful rather than executing a complete attack. To do this we can construct an evaluation metric based on the selection function. This is then used to construct a basic trace and from the differential trace we derive an equation which gives an indication of the level of security.

Traces tend to differ for different types of circuits and are in general more difficult to analyse the more balanced the circuit is. Consider the traces for two different implementations of a specific function; firstly we consider traces for one that is not balanced and then for one that is balanced. For this we will use a Galois inverse function [21] that is used in security circuits. The table for the Galois inverse of irreducible polynomial $x^4 + x^3 + 1$ is shown in Table 1. Table 1 uses the following binary encoding $0 = 00$, $1 = 01$, $\alpha = 10$, $\beta = 11$.

Table 1: Polynomial Inverse Table for $x^4 + x^3 + 1$.

$x \setminus y$	0	1	α	β
0	00	01	β0	α0
1	1α	$\beta\beta$	10	$\beta\alpha$
α	0β	β1	$\alpha\beta$	$\alpha\alpha$
β	0α	α1	1β	11

Two different types of circuit are to be analysed here for the inverse function; the first one using standard CMOS gates and the second one using dual-rail gates. The first implemention analysed is in terms of standard CMOS gates. For our measurements the key is first XOR-d with a range of input bits and the output is used directly as input to the circuit. The D selection function calculated for this was based on a choice of bit. Four key bits are involved in the circuit. CMOS circuits were described in Verilog, synthesized with Synopsys Design Compiler, and converted into SPICE format. The simulations were then executed using SPICE. For each experiment 1000 measurements was used as a standard measure i.e. samples were taken for 1000 ciphertexts. Power outputs were correlated using selection functions for each bit in the circuit. The largest value was extracted in each case. A diagram showing the differential power traces are shown in Fig. 1.

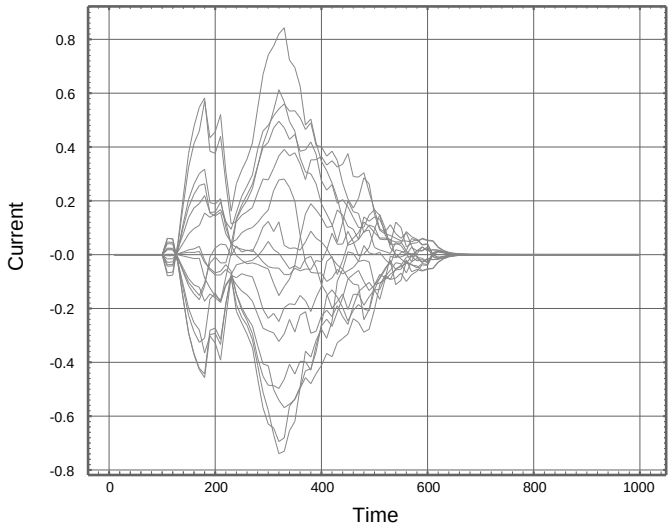

Figure 1: Differential trace - normal inverse circuit.

The second differential power analysis is for a standard dual-rail gate implementation. The dual-rail plot for the differential traces is as shown in Fig. 2. Here it can be seen that although the traces exhibit some similarity they exhibit more compactness.

We now present the CPA correlation metric. This is based on the Point Biserial Correlation coefficient(PBC). The point biserial correlation coefficient PBC is a correlation coefficient used when one variable is dichotomous, in this case 0 or 1. To calculate PBC, assume that the dichotomous variable Y has the two values 0 and 1. If we divide the data into two groups, group 1 which received the value "1" on Y and group 2 which received the value "0" on Y, then the point-biserial correlation coefficient is calculated as follows:

$$PBC = \frac{M_1 - M_0}{S_n}\sqrt{\frac{n_1 n_0}{n^2}} \tag{1}$$

Here M_1 is the mean value on the continuous variable X for all data points in group 1, M_0 is the mean value on the continuous variable X for all data points in group 2. Further, n_1 is the number of data points in group 1, n_0 is the number of data points in group 2 and n is the total sample size.

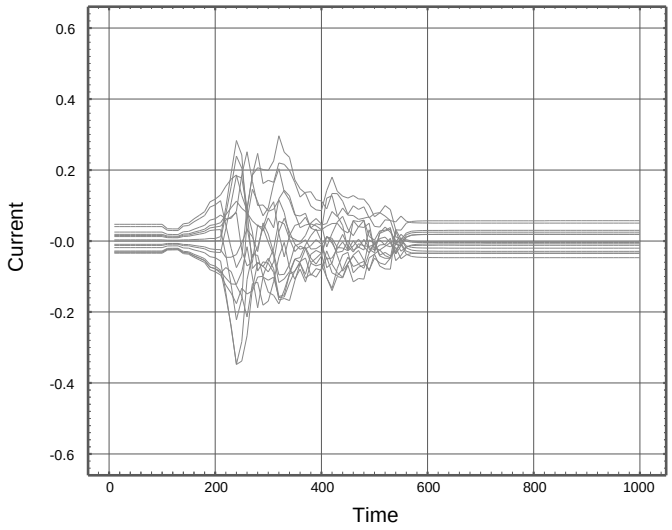

Figure 2: Differential trace - dual-rail inverse circuit.

S_n is the standard deviation used when you have data for every member of the population:

$$S_n = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{2}$$

It is easy to show algebraically that there is an equivalent formaula that uses S_{n-1}:

$$PBC = \frac{M_1 - M_0}{S_{n-1}}\sqrt{\frac{n_1 n_0}{n(n-1)}} \tag{3}$$

where S_{n-1} is the standard deviation used when you only have data for a sample of the population:

$$S_{n-1} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{4}$$

The PBC equation can be applied to find an appropriate measure of security in the following way. Equations 1 and 3 can be used to calculate the correlation for the different differential traces sampled. This can be done by taking sample points corresponding to the peaks in the graphs of the differential traces. In the equations n corresponds to the values for the D selection function. The correlation cofficient ranges between the values 0 and 1. The higher the PBC correlation value is, the less secure the circuit is; the lower the value is, the more secure it is.

In DPA the attack may centre on any of a number of specific bits. In this case if a range of bits is analysed we define the PBCM value as the PBC value corresponding to the highest or maximum

PBC value per bit measured. For the inverse example, for a complete set of traces, the PBCM results are shown in the following table.

Table 2: Example PBCM values.

Single-rail	0.49
1-of-2	0.31

Here it can be seen that the 1-of-2 circuit has a lower PCMB value than the single-rail circuit. This implies a lower correlation therefore implying a longer attack. The relationship between the metric and the number of measurements appears quadratic. Attacks above a PCMB of 0.5 are very easy to conduct with a few hundred ciphertexts. Successful attacks have been carried out at or just below a PCMB of 0.3, however, It becomes much more difficult to execute successful attacks significantly below this.

3 Secure Gate-level library

Our secure gate-level library is implemented using dynamic logic. In CMOS the use of N-nary encoding is well known where evaluations are performed in N-channel logic. Dynamic logic requires two phases. The first phase is called the precharge phase and the second phase the evaluation phase. There are three important benefits to N-channel only evaluation gates relative to traditional static gates [15].

(1) The first is the elimination of P-channel devices on input signals which reduces the input load significantly. In static gates the P-channel device tends to be significantly larger than the N-channel device which adds to the load. Because N-channel only evaluation gates do not require a P-channel device, their input load is reduced to one third that of a similar static gate. As a result, dynamic logic reduces circuit area by implementing compact 'NMOS style' gates without the overhead of static power dissipation.

(2) The second is the elimination of the need to build the complementary function in P-channel devices. With N-channel evaluation gates there is no need to implement the complementary function, either in N-channel or P-channel devices. This means that the more efficient faster N-channel gates are possible.

(3) The third advantage of N-channel only evaluation is the ability to share portions of the evaluate 'stack' among multiple outputs, which is not possible with static CMOS gates because it is not possible to obtain each output's function and complement from shared devices in both the P and N-channel stacks.

3.1 Basic N-nary gates

Here we show that by applying the N-nary gate principles it is possible to generate 1-of-n gates. As an example consider EXOR addition over GF(4). The addition table for GF(4) is shown in Table 3 where $\alpha = 10$ and $\beta = 11$.

GF4 has 4 states which can be represented by an encoding scheme with a balanced hamming weight. Table 4 shows the GF4 operations encoded in 1-of-4. It assumes the value zero=0001.

Table 3: GF4 Addition Table

+	0	1	α	β
0	0	1	α	β
1	1	0	β	α
α	α	β	0	1
β	β	α	1	0

Table 4: GF4 Addition encoded in 1-of-4

+	0001	0010	0100	1000
0001	0001	0010	0100	1000
0010	0010	0001	1000	0100
0100	0100	1000	0001	0010
1000	1000	0100	0010	0001

Using Table 4 it is possible to derive an N-nary 1-of-4 gate-level implementation which is shown in Fig. 3. The 1-of-4 Adder $\oplus_d$ gate has eight inputs, however, because only one of the α inputs and only one of the β inputs can be asserted at a time, it is convenient to treat these two sets of signals as individual inputs, each of which can represent one of four values.

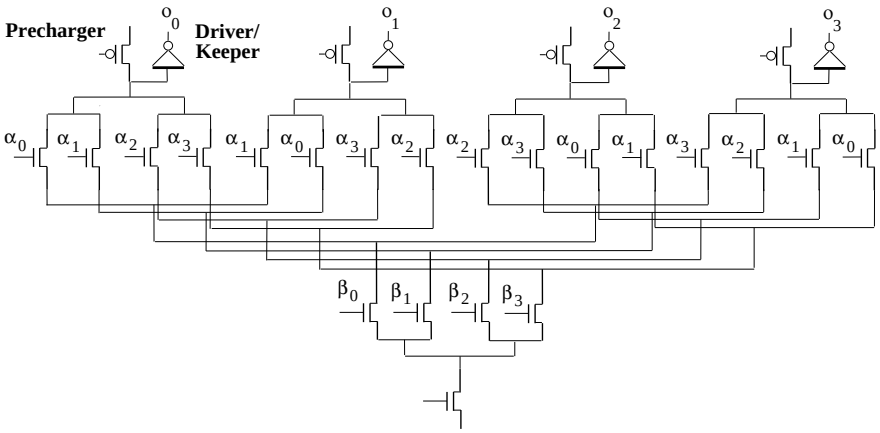

Figure 3: 1-of-4 Adder $\oplus_d$ gate.

Thus, the $\oplus_d$ gate has an α input signal and a β input signal, the α and β input signals can be any integer between zero and three inclusive. The function of the gate is to add the two inputs (GF addition) together and produce a 1-of-4 output. For the gate precharge devices are required for each output's evaluate node and each output must have its own driver/keeper cell as depicted in Fig. 3. Dynamic cells are constructed with a logical function built of NMOS devices coupled via

a precharged node. Precharge devices are required for each output's evaluate node since a single precharge device would short each output's evaluate node to the others. Keepers are sized to provide a compensation current to offset any and all sources of degrading current. For designs which have large numbers of NMOS transistors charge sharing becomes more of a problem. Charge sharing is mitigated by requiring internal precharge devices on any node which connects more than two transistors within the NMOS transistor trees. Any node which is connected beyond such a simple source/drain connection pair may have a significant amount of parasitic capacitance. This larger capacitance could share charge with the precharged node. To prevent this, the non-source/drain connected node is preset to a voltage level via an internal precharge device that prohibits charge sharing.

Power-balancing makes use of the 1-of-4 representation as only one signal is active at a time. For power-balancing it is important that the wires are balanced in such a gate so that capacitances are as level as possible. This is attained by balancing wire lengths from inputs to transistor and from transistor to outputs. This ensures that power signals are as balanced as possible. For this reason it is important to use regular gate structures where balancing is easier.

In the 1-of-4 representation, each block of 1-of-4 data is equivalent to two binary bits. If a binary function relies on odd data bits then bits from different data groups have to be merged together. For example suppose in binary that $\alpha = 10$ and $\beta = 01$ and we wish to merge the msb of α to the lsb of β to get 11. In 1-of-4 this is equivalent to merging $\alpha' = 0100$ and $\beta' = 0010$ to get 1000. Special merging functions are needed to implcitly merge the most significant or least significant values from two different 1-of-4 values to create a new 1-of-4 value. A table showing the combinations of msb and lsb merge operations is shown in Table 5. The table shows the operations in terms of their binary and 1-of-4 equivalents. The 1-of-4 values are represented in terms of bit-level equations.

Table 5: Merge operations

Operation	*Binary*	`1-of-4`	
`mergemsbmsb`	$\alpha_1\beta_1$	o_3	$(\alpha_2+\alpha_3)\cdot(\beta_2+\beta_3)$
		o_2	$(\alpha_2+\alpha_3)\cdot(\beta_0+\beta_1)$
		o_1	$(\alpha_0+\alpha_1)\cdot(\beta_2+\beta_3)$
		o_0	$(\alpha_0+\alpha_1)\cdot(\beta_0+\beta_1)$
`mergemsblsb`	$\alpha_1\beta_0$	o_3	$(\alpha_2+\alpha_3)\cdot(\beta_1+\beta_3)$
		o_2	$(\alpha_2+\alpha_3)\cdot(\beta_0+\beta_2)$
		o_1	$(\alpha_0+\alpha_1)\cdot(\beta_1+\beta_3)$
		o_0	$(\alpha_0+\alpha_1)\cdot(\beta_0+\beta_2)$
`mergelsbmsb`	$\alpha_0\beta_1$	o_3	$(\alpha_1+\alpha_3)\cdot(\beta_2+\beta_3)$
		o_2	$(\alpha_1+\alpha_3)\cdot(\beta_0+\beta_1)$
		o_1	$(\alpha_0+\alpha_2)\cdot(\beta_2+\beta_3)$
		o_0	$(\alpha_0+\alpha_2)\cdot(\beta_0+\beta_1)$
`mergelsblsb`	$\alpha_0\beta_0$	o_3	$(\alpha_1+\alpha_3)\cdot(\beta_1+\beta_3)$
		o_2	$(\alpha_1+\alpha_3)\cdot(\beta_0+\beta_2)$
		o_1	$(\alpha_0+\alpha_2)\cdot(\beta_1+\beta_3)$
		o_0	$(\alpha_0+\alpha_2)\cdot(\beta_0+\beta_2)$

A special gate which allows for the construction of merging functions is provided in the following subsection.

3.2 Optimized 1-of-n gates

It is more difficult to design 1-of-4 dynamic cells than dual-rail. The intention here is to construct a small scale standard cell library only. Therefore, the approach is restricted to designing only a limited number of cells which are highly generic and reusable. For power-balancing regular gate structures are preferable in order that transistors and wire lengths are matched. Optimisation is required to map to efficient gates which incorporate the above features. Such gates are designed to better balance and reduce the size of existing gates. This section introduces a set of optimized gates with good symmetrical properties. We start with showing how logic tree XOR functions in 1-of-4 can be optimized using a specialized type of symmetric gate called the exor-implicit gate. The structure of this gate is generic and can be used for implementing a variety of functions including the merge functions in 1-of-4. Further derivatives of this are then shown. The exor-implicit gate is shown in Fig. 4.

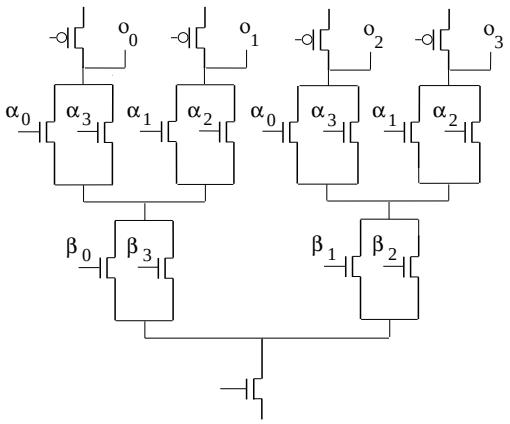

Figure 4: 1-of-4 Exor-implicit $\oplus_i$ gate.

The exor-implicit gate implemements the following binary equations.

$$o_1 = (\alpha_1 \oplus \alpha_0) \tag{5a}$$
$$o_0 = (\beta_1 \oplus \beta_0) \tag{5b}$$

As an example of its use consider the 2-bit binary signals $a = \{a_1, a_0\}$, $b = \{b_1, b_0\}$, $c = \{c_1, c_0\}$ and $o = \{o_1, o_0\}$. Assume these are related by the following binary equations:

$$o_1 = (a_1 \oplus a_0) \tag{6a}$$
$$o_0 = (b_1 \oplus b_0) \oplus (c_1 \oplus c_0) \tag{6b}$$

Assume also that a', b', c', o' are the corresponding 1-of-4 equivalent values. If the above equations had to be implemented in 1-of-4 the equivalent equation using the $\oplus_d$ operation as defined in Table 4 would be as follows:

$$\begin{aligned} o' &= (mergemsblsb(a',\, b' \oplus_d c')) \oplus_d \\ &\quad (mergelsbmsb(a',\, b' \oplus_d c')). \end{aligned} \tag{7}$$

Using the $\oplus_i$ operation this can be reduced to the following more simple expression:

$$o' = (a' \oplus_i (b' \oplus_i c'). \tag{8}$$

It is much more efficient, therefore, to represent equations (6a, 6b) using 1-of-4 $\oplus_i$ gates. It is also much easier to route wires into more simpler regular 1-of-4 gates such as this.

Merge gates have the same structure as the $\oplus_i$ gate but they use different combinations of inputs. For example, to implement *mergemsbmsb* using the $\oplus_i$ gate in Fig. 4 the input α_3 needs to be switched with α_1 and input α_1 switched with α_3 (similarly β_1 and β_3 are switched). The corresponding *mergemsbmsb* gate is shown in Fig. 5.

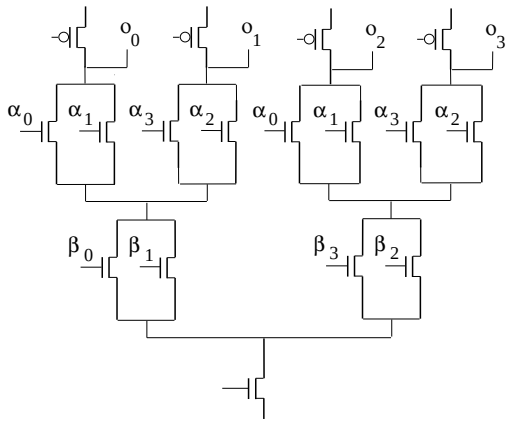

Figure 5: 1-of-4 mergemsbmsb gate.

Various combinations of merge gates implemented using $\oplus_i$ are shown in Table 6 with the corresponding input changes.

Another gate in the library is the 1-of-4 exorhalf-implicit $\oplus_{hni}$ gate. In binary it is used to add 2 bits from one binary pair and merge the result to 1 bit from another binary pair e.g. $\oplus_{h1i}$ implements the following binary equations

$$o_1 = a_1 \oplus a_0 \tag{9a}$$

$$o_0 = b_1 \tag{9b}$$

A set of $\oplus_{hni}$ gates exist which are similar to $\oplus_{h1i}$. Table 7 shows the operations of the $\oplus_{hni}$ gates in terms of their binary and 1-of-4 equivalents.

Table 6: Converting $\oplus_i$ to Merge gates

$\oplus_i$	$\alpha_0\alpha_3$	$\alpha_1\alpha_2$	$\beta_0\beta_3$	$\beta_1\beta_2$
$mergemsbmsb$	$\alpha_0\alpha_1$	$\alpha_3\alpha_2$	$\beta_0\beta_1$	$\beta_3\beta_2$
$mergemsblsb$	$\alpha_0\alpha_1$	$\alpha_2\alpha_3$	$\beta_0\beta_2$	$\beta_1\beta_3$
$mergelsbmsb$	$\alpha_0\alpha_2$	$\alpha_1\alpha_3$	$\beta_0\beta_1$	$\beta_2\beta_3$
$mergelsblsb$	$\alpha_0\alpha_2$	$\alpha_1\alpha_3$	$\beta_0\beta_2$	$\beta_1\beta_3$

Table 7: $\oplus_{hni}$ operations

Operation	*Binary*		`1-of-4`	
$\oplus_{h1i}$	o_1	$\alpha_1 \oplus \alpha_0$	o_3	$(\alpha_1+\alpha_2)\cdot(\beta_3+\beta_2)$
	o_0	β_1	o_2	$(\alpha_1+\alpha_2)\cdot(\beta_0+\beta_1)$
			o_1	$(\alpha_0+\alpha_3)\cdot(\beta_3+\beta_2)$
			o_0	$(\alpha_0+\alpha_3)\cdot(\beta_0+\beta_1)$
$\oplus_{h2i}$	o_1	$\beta_1 \oplus \beta_0$	o_3	$(\alpha_2+\alpha_3)\cdot(\beta_1+\beta_2)$
	o_0	α_1	o_2	$(\alpha_0+\alpha_1)\cdot(\beta_1+\beta_2)$
			o_1	$(\alpha_2+\alpha_3)\cdot(\beta_0+\beta_3)$
			o_0	$(\alpha_0+\alpha_1)\cdot(\beta_0+\beta_3)$
$\oplus_{h3i}$	o_1	$\alpha_1 \oplus \alpha_0$	o_3	$(\alpha_1+\alpha_2)\cdot(\beta_1+\beta_3)$
	o_0	β_0	o_2	$(\alpha_1+\alpha_2)\cdot(\beta_0+\beta_2)$
			o_1	$(\alpha_0+\alpha_3)\cdot(\beta_1+\beta_3)$
			o_0	$(\alpha_0+\alpha_3)\cdot(\beta_0+\beta_2)$
$\oplus_{h4i}$	o_1	$\beta_1 \oplus \beta_0$	o_3	$(\alpha_1+\alpha_3)\cdot(\beta_1+\beta_2)$
	o_0	α_0	o_2	$(\alpha_0+\alpha_2)\cdot(\beta_1+\beta_2)$
			o_1	$(\alpha_1+\alpha_3)\cdot(\beta_0+\beta_3)$
			o_0	$(\alpha_0+\alpha_2)\cdot(\beta_0+\beta_3)$

$\oplus_{hni}$ gates also have the same structure as the $\oplus_i$ gate but they use different combinations of inputs. Various combinations of $\oplus_{hni}$ gates implemented using $\oplus_i$ are shown in Table 8 with the corresponding input changes.

Another variation on the above gate is one where (6a) and (6b) undergo modification using $\oplus 1$ to the following.

$$o_1 = a_1 \oplus a_0 \tag{10a}$$

$$o_0 = b_0 \oplus 1 \tag{10b}$$

In this case Table 7 can be modified to generate the following $\oplus_{ni+1}$ gates shown in Table 9. These gates also share the same structure as the $\oplus_i$ gate.

Various mixed 1-of-n gates exist in the library for example gates with 1-of-2 inputs and 1-of-4 outputs or vica versa. An example of one of these is depicted in Fig. 6. This gate mimics the $\oplus_d$

Table 8: Converting $\oplus_i$ to $\oplus_{hni}$ gates

$\oplus_i$	$\alpha_0\alpha_3$	$\alpha_1\alpha_2$	$\beta_0\beta_3$	$\beta_1\beta_2$
$\oplus_{h1i}$	$\alpha_0\alpha_3$	$\alpha_1\alpha_2$	$\beta_0\beta_1$	$\beta_3\beta_2$
$\oplus_{h2i}$	$\beta_0\beta_3$	$\beta_1\beta_2$	$\alpha_0\alpha_1$	$\alpha_2\alpha_3$
$\oplus_{h3i}$	$\alpha_0\alpha_3$	$\alpha_1\alpha_2$	$\beta_0\beta_2$	$\beta_3\beta_1$
$\oplus_{h4i}$	$\beta_0\beta_3$	$\beta_1\beta_2$	$\alpha_0\alpha_2$	$\alpha_1\alpha_3$

Table 9: $\oplus_{hni+1}$ operations

Operation	*Binary*		`1-of-4`	
$\oplus_{h1i+1}$	o_1	$\alpha_1 \oplus \alpha_0$	o_3	$(\alpha_1 + \alpha_2) \cdot (\beta_0 + \beta_1)$
	o_0	$\beta_1 \oplus 1$	o_2	$(\alpha_1 + \alpha_2) \cdot (\beta_3 + \beta_2)$
			o_1	$(\alpha_0 + \alpha_3) \cdot (\beta_0 + \beta_1)$
			o_0	$(\alpha_0 + \alpha_3) \cdot (\beta_3 + \beta_2)$
$\oplus_{h2i+1}$	o_1	$\beta_1 \oplus \beta_0$	o_3	$(\alpha_0 + \alpha_1) \cdot (\beta_1 + \beta_2)$
	o_0	$\alpha_1 \oplus 1$	o_2	$(\alpha_2 + \alpha_3) \cdot (\beta_1 + \beta_2)$
			o_1	$(\alpha_0 + \alpha_1) \cdot (\beta_0 + \beta_3)$
			o_0	$(\alpha_2 + \alpha_3) \cdot (\beta_0 + \beta_3)$
$\oplus_{h3i+1}$	o_1	$\alpha_1 \oplus \alpha_0$	o_3	$(\alpha_1 + \alpha_2) \cdot (\beta_0 + \beta_2)$
	o_0	$\beta_0 \oplus 1$	o_2	$(\alpha_1 + \alpha_2) \cdot (\beta_1 + \beta_3)$
			o_1	$(\alpha_0 + \alpha_3) \cdot (\beta_0 + \beta_2)$
			o_0	$(\alpha_0 + \alpha_3) \cdot (\beta_1 + \beta_3)$
$\oplus_{h4i+1}$	o_1	$\beta_1 \oplus \beta_0$	o_3	$(\alpha_0 + \alpha_2) \cdot (\beta_1 + \beta_2)$
	o_0	$\alpha_0 \oplus 1$	o_2	$(\alpha_1 + \alpha_3) \cdot (\beta_1 + \beta_2)$
			o_1	$(\alpha_0 + \alpha_2) \cdot (\beta_0 + \beta_3)$
			o_0	$(\alpha_1 + \alpha_3) \cdot (\beta_0 + \beta_3)$

gate shown in Fig. 3 but it takes in two inputs in dual-rail format and produces a 1-of-4 output.

Efficient mixed 1-of-n gates for the AND function can be found in conjunction with EXOR which produce dual-rail output. For example, consider the following binary equation where o represents a single bit.

$$o = (a_1 \cdot b_1) \oplus (a_0 \cdot b_0) \tag{11}$$

This equation gives an inefficient CMOS implementation. It is better to represent such an equation in SOP form.

$$o = (a_1 \cdot b_1 \cdot \bar{a}_0) + (\bar{a}_1 \cdot a_0 \cdot b_0) + (a_1 \cdot \bar{b}_1 \cdot a_0 \cdot b_0) + (a_1 \cdot b_1 \cdot a_0 \cdot \bar{b}_0) \tag{12}$$

This equation can be converted to its corresponding 1-of-4 input, 1-of-2 output representation to give the following pair of dual-rail equations.

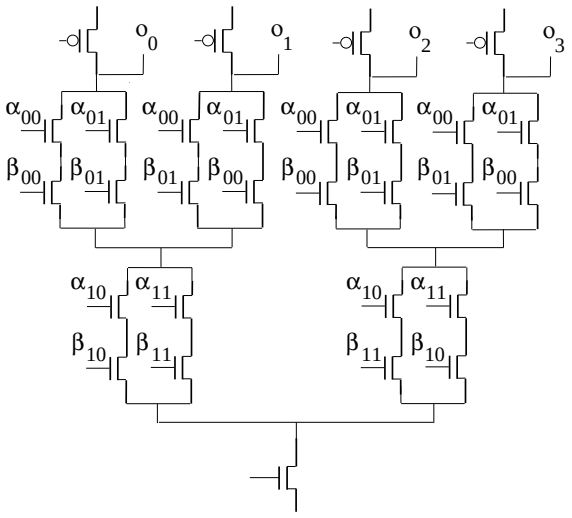

Figure 6: 1-of-2 in, 1-of-4 out Exor $\oplus_{24}$ gate.

$$o_1 = a_1b_1 + a_1b_3 + a_2b_2 + a_2b_3 + a_3b_1 + a_3b_2 \tag{13a}$$

$$o_0 = a_0b_0 + a_0b_1 + a_0b_2 + a_0b_3 + a_1b_0 + a_1b_2 + a_2b_0 + a_2b_1 + a_3b_0 + a_3b_3 \tag{13b}$$

These equations can be mapped to a 1-of-4 input, 1-of-2 output gate. The gate is implemented in two halves. The half gate for o_0 is shown in Fig. 7.

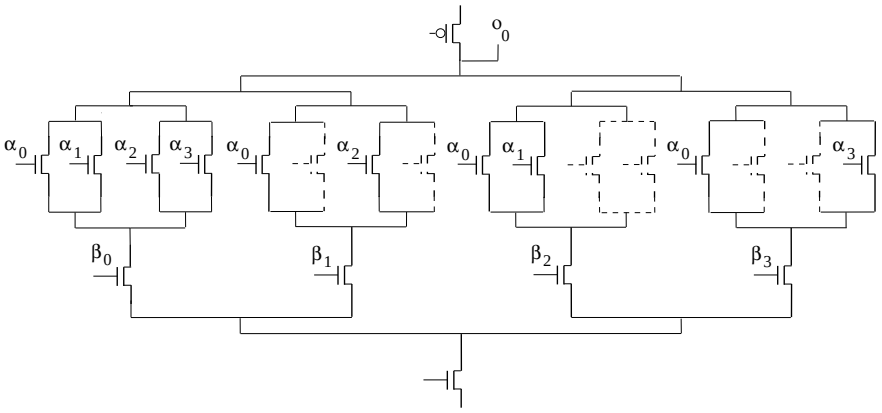

Figure 7: AND-EXOR-half &$\oplus_{42}$ gate 1.

A similar gate exists for o_1 which is shown in Fig. 8.

Here it can be seen that for each single positive term we have a transistor pair that is active in terms of a and b. Where transistors are shown dotted they are removed. The paths are balanced

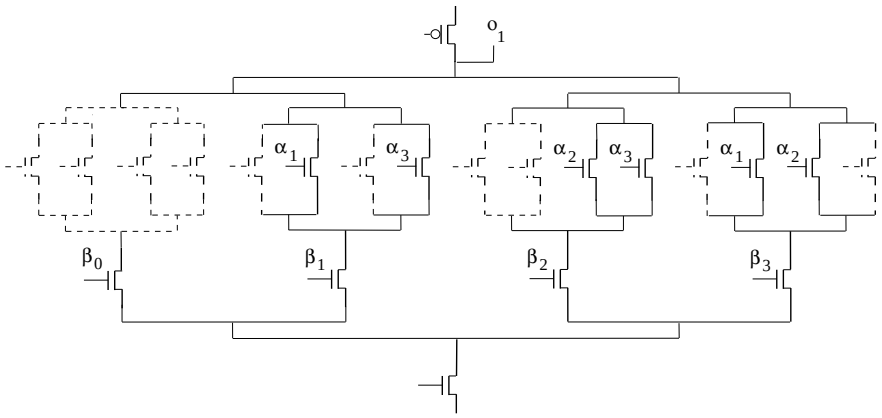

Figure 8: AND-EXOR-half $\&\oplus_{42}$ gate 2.

by length as depicted. In the actual cell the wire lengths are balanced as appropriate by length.

4 Design Flow

Our Secure Design flow (partially automatic) is now presented. Fig. 9 gives a block diagram which depicts the design-flow.

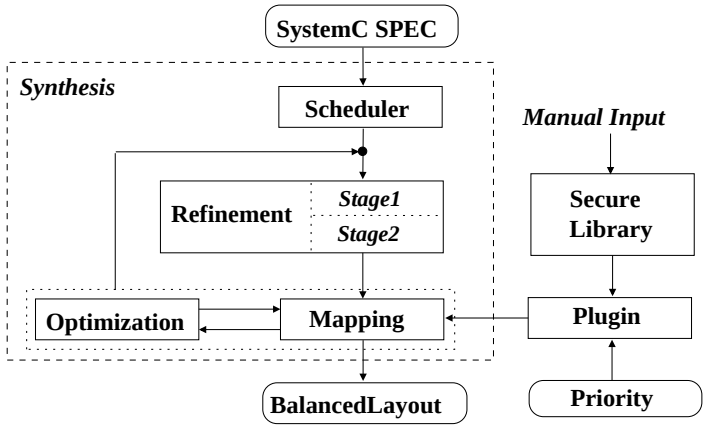

Figure 9: Diagram of design flow.

The security design flow inputs a SystemC description of the cryptographic specification. After behavioural compilation an extraction package is used to access relevant information about the modules. The design-flow then proceeds along various refinement stages, including Galois subfield breakdown, to generate small regular blocks. A secure library of components, based on section III, is accessible as a plugin. The components are mapped using partially automatic mapping to the sub-modules to generate a secure circuit.

4.1 Refinement

The design flow proceeds by refining the specification into smaller sub-modules. This passes through various stages of refinement starting with stage one which creates a sub-level representation making use of sub-field generation [22]. It makes use of the fact that a field over $GF(2^n)$ can be formed from a composite of fields $GF(2^n/2)$ and $GF(2)$. The subfield mapping makes use of the notion that any arbitrary polynomial can be represented as $ax + b$, given an irreducible polynomial of $x^2 + ax + b$. Thus, an element in $GF(2^8)$ may be represented as $ax + b$ where a and b are elements from the field $GF(2^4)$. The S-box of the AES is broken down using subfields into smaller components. Fig. 10 shows a diagram for the subfield generation.

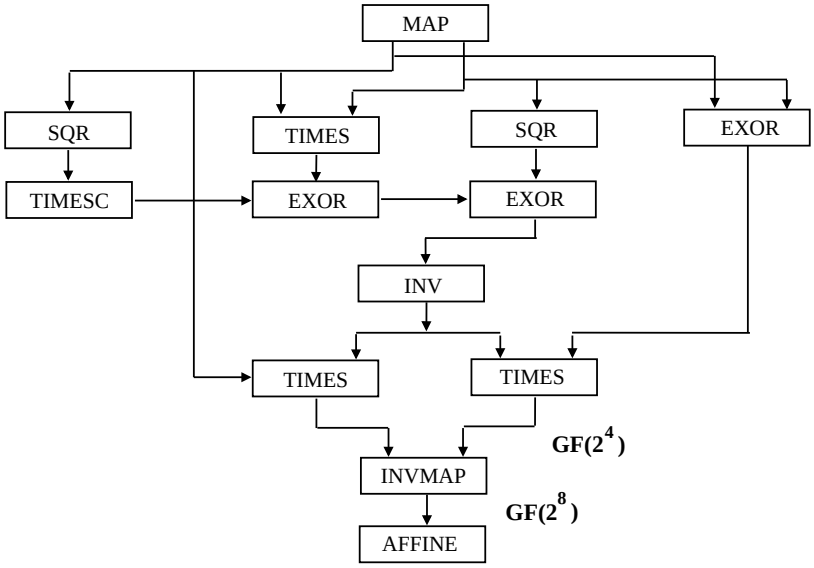

Figure 10: Subfield breakdown.

A next level of refinement is used to break the design down further. This employs either (i) the use of further subfield refinement or (ii) transformation using bases.

(i) Further use of subfield refinement makes use of the fact that the field $GF(2^4)$ is isomorphic to the composite field $GF((2^2)^2)$. This means that the field $GF(2^8)$ can now be transformed into the field $GF(((2^2)^2)^2)$. Here $GF(((2^2)^2)^2)$ is a field extension of degree 2 over $GF((2^2)^2)$ constructed using the irreducible polynomial $x^2 + ax + b$ where $a, b \in GF((2^2)^2)$.

(ii) For transformation using bases use is made of the dual-basis.

As an example of transformation of basis we consider a dual-basis transformation which can be applied to generate a 4-bit subfield multiplier. A regular polynomial-dual basis multiplier for $GF(2^m)$ can be automatically constructed out of a number of inner products and a number of non-symmetric additions. In [23] they describe a transformation using dual-basis where underlying equations can be generated for the inner products and addition trees which exhibit more regularity.

As a result of the dual-basis transformation the subfield multiplier can be implemented more efficiently using regular blocks as shown in Fig. 11. In Fig. 11 BLOCK1 implements the higher $\oplus$ terms and BLOCK2 implements the more regular inner product addition trees. As a result of

the more regular structure it is now more efficient to translate to a 1-of-4 representation and the balancing of the subsequent layout becomes easier.

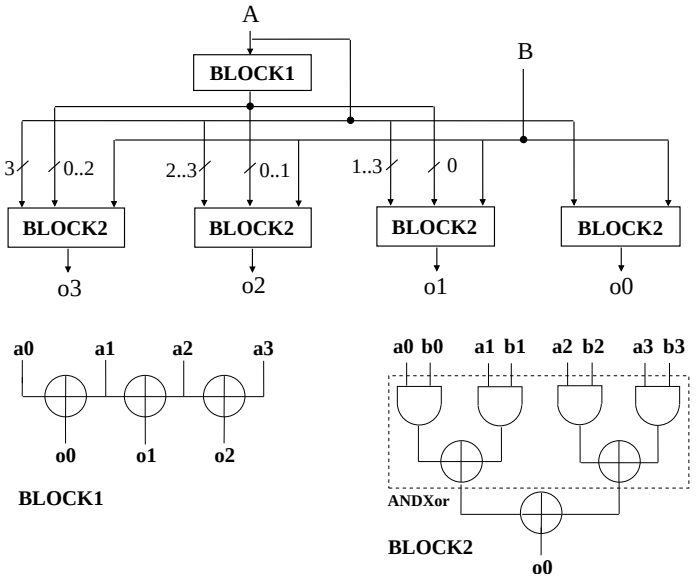

Figure 11: Multiplier Regular Implementation.

4.2 Mapping

The 1-of-n library in section III is used for mapping to the refined circuit. The library is split into two divisons: one with basic cells and one with optimized cells as shown in section III. Sub-modules undergo logic optimisation. Davio decomposition is first applied to generate decision graphs. These are subsequently subject to transformations including variable ordering and reduction. Also specific transformations are applied at this point to find the optimal alignment of inputs for 1-of-4 representation. During optimization this aims to target the minimum conversion which requires the least number of merge functions.

The mapping algorithm employs a greedy algorithm which inputs each sub-module and processes each of the modules 1-of-4 outputs. A priority list of gate types is used. The gates at the head of the list are given the highest priority. The priority list is ordered with the most complex gates processed first. These are subsequently replaced by smaller gates during mapping to see if a more efficient solution can be found.

It is easy to find an efficient mapping to XOR gates using 1-of-4. Thus during mapping the XOR plane is predominantly targetted using 1-of-4 gates. For the AND plane targetting to 1-of-4 gates is not so efficient. Therefore, on switching between planes, a manual choice is made to convert from 1-of-4 to dual-rail and back again. This is done by providing an option in the tool. Here mapping is made to mixed 1-of-4, 1-of-2 XOR-AND and AND-XOR gates where the planes adjoin.

Blocks are tested to see if they can be merged together to provide for a more efficient solution. If this is feasible they are merged and optimized prior to mapping. Simple sub-modules e.g. those which use a single layer of functionality are merged automatically prior to mapping.

Where transformation between bases is used the implementation must take into account differences in bases between blocks which is accounted for during mapping. To convert from polynomial to dual-basis input normally requires conversion. However, the hardware required for this transformation is often trivial. For irreducible trinomials no extra hardware is required to carry out the basis conversion only a reordering of coefficients.

For synchronous implementation a spacer protocol is used which uses a return to zero. The all-zeros state is used to indicate the absence of data, which separates one code word from another. This makes use of special power-balanced registers [12] which ensures power-balancing is achieved.

It is essential in order to achieve constant power consumption that a fixed amount of charge is used per transition. This means that the load capacitances at the outputs should be matched. This means designing the cells with internal paths that are of exactly the same length. For routing, in order for the logic gate not to have a different power signature for each output event that is possible during this transition, the output capacitances must also be matched and routing differences may not exist between the nets. Routing strategy external to cells can be carried out using techniques similar to those used for dual-rail techniques by routing four signals with parallel routes that are, at all times, in adjacent tracks of the routing grid, on the same layers, and of the same length. In this case, each set of four wires must be abstracted as a single fat wire. Crosstalk can be eliminated by shielding each set of four wires with power lines.

4.3 Example

The design flow example centers on the AES S-box [24]. Here we assume the specification has been entered and compiled and is ready for refinement and mapping. The design is first refined, using one level of subfield refinement only, converted to 1-of-n and then logic mapped to a synchonous implementation.

The design is initially broken down to the first level L1 using subfield generation. The S-box is initially refined to the circuit shown in Fig. 10. Each of the subblocks appearing in Fig. 10 is subsequently mapped to the 1-of-n library.

The multipliers are converted using the dual basis outlined in section 3 and mapped. The example that follows is a detailed breakdown of the multiplier outlined in sub-section 4.1. The multiplier is refined to the dual-basis multiplier blocks of Fig. 11, comprising BLOCKS 1 & 2, and these are mapped to 1-of-n components and a 1-of-n circuit generated. Use here is made of the cross over between 1-of-4 and 1-of-2 over the AND-XOR plane. The diagram showing the complete 1-of-n mapping is shown in Fig. 12.

The following description details the mapping from the 4-bit multiplier in Fig. 11 to the 1-of-n implementation shown in Fig. 12. The binary circuit for BLOCK1, shown at the bottom left of Fig. 11, takes a 4-bit input A {a3,a2,a1,a0} and produces 3 single-bit outputs.

After mapping to 1-of-4 cells, BLOCK1 is shown implemented in the upper part of Fig. 12 using a combination of Merge, XOR-implicit and XOR1/2-implicit gates. The inputs to these gates i.e. a3..2, a1..0 etc., relate to the corresponding pairs of bits in Fig. 11 and are assumed in Fig. 12 to be in their equivalent 1-of-4 format. The inputs to the AND-XOR blocks of BLOCK2 as depicted in the top diagram of Fig. 11 are formed from a mixed selection of the bits of the input A and various bits of the output of BLOCK1. The outputs from the Merge, XOR-implicit and XOR1/2-implicit gates in Fig. 12 provide the equivalent 1-of-4 data representation to the 1-of-n AND-XOR cells of BLOCK2.

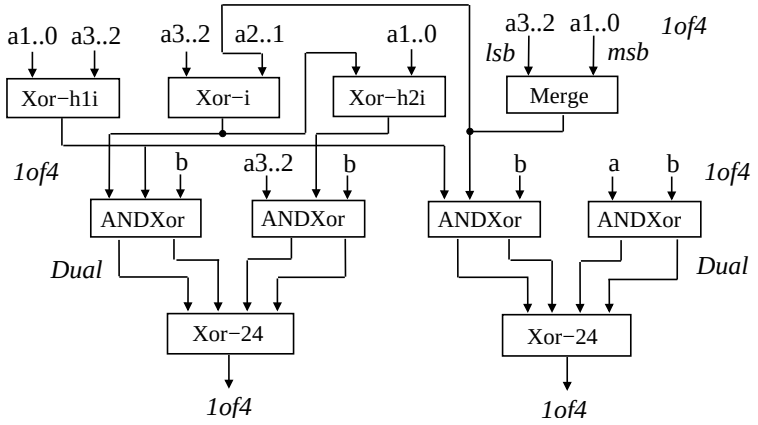

Figure 12: Multiplier 1-of-n implementation.

The AND-XOR cells of BLOCK2 are optimally mapped to the 1-of-4 input, 1-of-2 ouput AND-XOR $\&\oplus_{42}$ circuits depicted in Fig. 7 and Fig. 8. Each AND-XOR block in Fig. 12 is mapped to two 1-of-4 input, 1-of-2 ouput $\&\oplus_{42}$ circuits. Finally the outputs of the AND-XOR which are in 1-of-2 format are combined in pairs to generate 4-bit values in dual-rail format. These are input to the remaining XOR operations of BLOCK2 which are combined in pairs and mapped to the 1-of-2 input, 1-of-4 output adder $\oplus_{24}$ circuits shown in Fig. 6.

Many parts of the S-box implementation rely on XOR gates exclusively and these are mapped exclusively to 1-of-4 cells. For XOR blocks such as the map function or affine function the implicit-XOR cell together with its variants in section 2 is used for mapping.

As an example consider the mapping for the affine sub-module. The binary equations for the affine transformation are as follows:

$$\begin{aligned}
o_7 &= a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \\
o_6 &= a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus 1 \\
o_5 &= a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus 1 \\
o_4 &= a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\
o_3 &= a_7 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\
o_2 &= a_7 \oplus a_6 \oplus a_2 \oplus a_1 \oplus a_0 \\
o_1 &= a_7 \oplus a_6 \oplus a_5 \oplus a_1 \oplus a_0 \oplus 1 \\
o_0 &= a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_0 \oplus 1
\end{aligned} \tag{14}$$

These are optimized and mapped using the 1-of-4 mapping technique. After applying the mapping algorithm the affine sub-module is implemented using 1-of-4 gates as shown in Fig. 13.

Here all gates shown are derivatives of $\oplus_i$. It is implemented using 8 $\oplus_i$ and 8 $\oplus_h$ variants of $\oplus_i$ as depicted in Table 7. Some of the equations for the affine transformation end in $\oplus 1$. These parts of the equations can be mapped to $\oplus_{hni+1}$ gates as depicted in Table 9. In Fig. 13 it can be seen the affine transformation is generated using a highly regular layout.

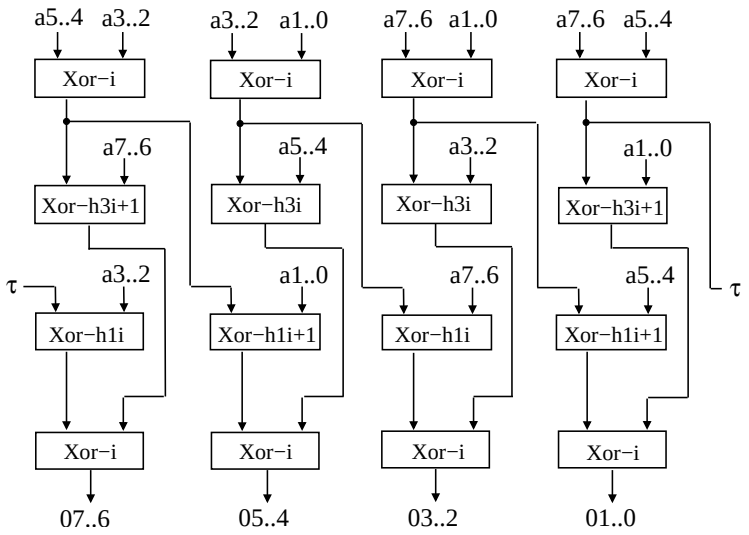

Figure 13: Affine implementation.

During mapping the small modules are tested for merging and are merged if this is found to be more optimal. As an example consider the merge between the SQUARE sub-module and the TIMESC sub-module shown on the left of Fig. 10.

The binary equations for the SQUARE sub-module are:

$$\begin{aligned} o_3 &= a_3 \\ o_2 &= a_1 \oplus a_3 \\ o_1 &= a_2 \\ o_0 &= a_0 \oplus a_2 \end{aligned} \tag{15}$$

The binary equations for the CTIMES sub-module are:

$$\begin{aligned} o_3 &= a_0 \oplus a_1 \oplus a_2 \oplus a_3 \\ o_2 &= a_0 \oplus a_1 \oplus a_2 \\ o_1 &= a_0 \oplus a_1 \\ o_0 &= a_1 \oplus a_2 \oplus a_3 \end{aligned} \tag{16}$$

After merging these two sub-modules together they are optimized and mapped to the circuit shown in Fig. 14. The mapping requires derivatives of the $\oplus_i$ only.

The remaining blocks in Fig. 10 are mapped using $\oplus_i$ and its derivatives apart from the inverse which is mapped to gates which are similar to the multiplier.

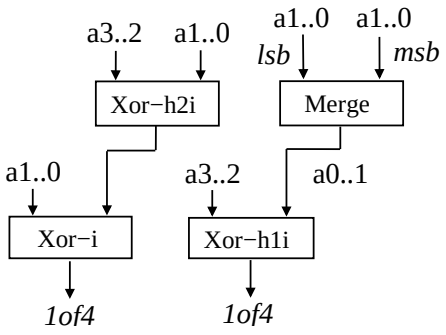

Figure 14: Mapping for merged SQUARE and CTIMES blocks.

Table 10: Multiplier Comparison

Level	Type	Gates	Transistors	Area	Delay	Power	PBCM
L1	Dual-rail	31	610	574	$0.69ns$	0.44	0.40
	Basic 1-of-n	49	995	968	$0.34ns$	0.30	0.23
	Optimized 1-of-n	22	520	488	$0.17ns$	0.15	0.21
L2.1	Dual-rail	35	950	812	$0.74ns$	0.59	0.55
	Basic 1-of-n	41	897	766	$0.33ns$	0.26	0.55
	Optimized 1-of-n	25	545	465	$0.19ns$	0.23	0.40
L2.2	Dual-rail	31	610	574	$0.69ns$	0.44	0.40
	Basic 1-of-n	30	590	555	$0.18ns$	0.28	0.16
	Optimized 1-of-n	14	350	329	$0.15ns$	0.96	0.19

5 Results

Experiments were conducted over a range of multipliers and S-boxes which were generated by varying the refinement level. A detailed set of security results including gate-count, transistor-count, time and power comparisons have been made for multipliers corresponding to the different refinement levels. A similar set of results has been taken for the corresponding S-boxes for similar refinement levels. The security metric (PBCM) was measured in each case using bit-level DPA: the correlation metric was measured for the differential trace for each bit and the highest value extracted in each case. CMOS circuits were described in Verilog, synthesized with Synopsys Design Compiler, and converted into SPICE format. The dynamic circuits have been generated by our design flow into SPICE fromat. All simulations were then executed using SPICE. For each experiment 1000 measurements was used as a standard measure i.e. samples were taken for 1000 ciphertexts. Tables 10 and 11 show comparions for our synchronous synthesized results over power-balanced results generated for standard dual-rail cells using a Synopsys 90 **nm** cell library.

Table 10 shows comparisons for the multipliers at different refinement levels. The results in Table 10 are shown in 8 columns. The first depicts the level i.e. L1 represents the 1st stage of refinement using subfield breakdown, L2.1 shows subsubfield breakdown whereas L2.2 uses dual-

Table 11: S-box Comparison

Level	Type	Gates	Transistors	Area	Delay	Power	PBCM
L1	Dual-rail	188	4360	3934	$3.66ns$	3.22	0.29
	Basic 1-of-n	351	6910	6234	$1.22ns$	2.11	0.25
	Optimized 1-of-n	155	3145	2837	$0.93ns$	1.17	0.22
L2.1	Dual-rail	216	6010	5230	$4.44ns$	4.13	0.17
	Basic 1-of-n	313	6468	5628	$1.45ns$	2.24	0.19
	Optimized 1-of-n	183	3577	3112	$1.18ns$	1.29	0.17
L2.2	Dual-rail	188	4360	3934	$3.66ns$	3.22	0.29
	Basic 1-of-n	294	5593	5046	$1.04ns$	1.98	0.21
	Optimized 1-of-n	131	2635	2377	$0.82ns$	1.12	0.17

basis. The second column depicts the type of implementation technology in terms of dual-rail, basic 1-of-n or optimized 1-of-n. The third and fourth columns show the number of gates and transistors respectively. The fifth and sixth column shows the area and delay. Finally the seventh and eighth columns shows the power and the security measure (PBCM). In Table 10 it can be seen that at levels L2.1 and L2.2 a small percentage gain in area in terms of transistors is made for the basic 1-of-n gates over dual-rail. For level L1 there is a jump in the gate and transistor number for the basic 1-of-n gates which is due to the large number of 1-of-4 merge functions required. These additional gates are reduced significantly when applying the mapping algorithm to generate optimized 1-of-4 gates resulting in a corresponding smaller gate count. There is therefore a large saving in gates made for a switch from basic gates to 1-of-n optimized gates. At level L2.2 a large percentage saving in area in terms of transistors is made for the optimized 1-of-n gates over dual-rail. This shows the best results for gate and transistor count were achieved using dual-basis which is due to the optimal gate mapping to larger gates. A significant saving in delay is apparent for 1-of-n over dual-rail because of the faster N-nary technology used. Power readings show that a saving in switching is apparent over dual-rail of a fair percentage.

The security measure PBCM (most correlated bit) on average shows a difference between dual-rail, basic 1-of-n gates and optimized 1-of-n gates. There is a 32 percent average difference between dual-rail and basic 1-of-n gates and a 43 percent average difference between dual-rail and optimized 1-of-n gates. The design using subsubfield refinement L2.1 shows less of a difference than the average. Because of the significant difference in results on an individual basis it is assumed that for the multiplier circuits the security measure is more dependent on design style.

Table 11 shows similar comparisons for the S-box for the different levels L1, L2.1 and L2.2. At each level there is a jump in the gate and transistor number for the basic 1-of-n gates this time which is due to the large number of 1-of-4 merge functions required. As before these additional gates are reduced significantly when generating optimized 1-of-n gates resulting in a corresponding smaller gate count. In Table 11 the best results are for the dual-basis as was the case for the multiplier which again is due to the optimal mapping to larger gates. The reduction also results in a corresponding smaller delay for optimized 1-of-n gates over basic 1-of-n.

The PBCM values for the S-boxes are in general lower than those for the multipliers as expected.

However, there is not as much variation between the PBCM values for the different S-boxes. This time there is a 12 percent average difference between dual-rail and basic 1-of-n gates and a 24 percent average difference between dual-rail and optimized 1-of-n gates. This shows that for the S-boxes the security measure is also dependent on the design but less so than for the multipliers.

6 Conclusions

This paper has presented a novel design-flow to generate efficient secure balanced circuits. A new design flow using 1-of-n encoding, which is partially automatic, is presented as a means to provide more efficient power-balanced circuits than can be provided by dual-rail alone. Preliminary breakdown or refinement at the subfield level to generate small regular blocks creates a suitable platform for efficient mapping to the 1-of-n library. We have presented a new library of optimized power-balanced cells using N-nary 1-of-n logic which represent an improvement over dual-rail. Preliminary results indicate improvements in area, time and power over dual-rail for optimized 1-of-n gates.

A security metric has been introduced based on DPA and CPA measurements that enables the evaluation of security of balanced circuits that are more difficult to measure. The security measure presented here enables comparison of the generated circuits on an individual basis highlighting which circuits are more secure. This enables us to construct a component library with better security. On average the results presented suggest our 1-of-n circuits provide similar or better security than standard dual-rail, however, this is dependent on the design.

For future work it is intended to automate the more complex tasks and adapt the design flow to a broader range of more general circuits.

Acknowledgment

The authors would like to thank everyone who has contributed to this work. This work was supported by EPSRC grants GR/S81421 (project SCREEN) and EP/F016786/1 (project SURE).

References

[1] A. Menezes, P. Oorschot and S. Vanstone, "Handbook of Applied Cryptography," *CRC Press,* 2004.

[2] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," *Proc. Advances in Cryptography - CRYPTO 1999,* pp. 388-397, 1999.

[3] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," *Proc. Advances in Cryptography - CRYPTO 1997,* pp. 513-525, 1997.

[4] T. Messerges, E. Dabbish and R. Sloan, "Examining Smart Card Security under the Threat of Power Analysis Attacks," *IEEE Trans. Comp.,* Vol 51, No. 5, May. 2002, pp. 541-552.

[5] N. Kamoun, L. Bossuet and A. Ghazel, "Experimental Implementation of DPA attacks on AES design with Flash-based FPGA technology," *Int. Conf. Systems, Signals and Devices - SSD 2009,* pp. 1-4, 2009.

[6] E. Brier, C. Clavier and F. Olivier, "Correlation Power Analysis with a Leakage Model," *Proc. Cryptographic Hardware and Embedded Systems - CHES 2004,* LCNS 3156, pp. 16-29, Springer-Verlag, 2004.

[7] K. Tiri and I. Verbauwhede, "A VLSI Design Flow for Secure Side-Channel Attack Resistant IC's," *Proc. Design Automation and Test in Europe - DATE 2005,* pp. 58-63, 2005.

[8] K. Tiri and I. Verbauwhede, "Design Method for Constant Power Consumption of Differential Logic Circuits," *Proc. Design Automation and Test in Europe - DATE 2005,* pp. 628-633, 2005.

[9] M. Aigner, M. Mangard, F. Menichelli, R. Menicocci, M. Olivieri, T. Popp, G. Scotti and A. Trifiletti, "Side Channel Analysis Resistant Design Flow," *In Proc. ISCAS 2006,* pp. 2909-2912, 2006.

[10] M. Aigner, M. Mangard, R. Menicocci, M. Olivieri, G. Scotti and A. Trifiletti, "A Novel CMOS Logic Style with Data Independent Power Consumption," *Proc. Int'l Symp. Circuits and Systems - ISCAS 2005,* pp. 1066-1069, 2005.

[11] S. Moore, R Anderson, P Cunningham, Robert Mullins and G. Taylor, "Improving Smart Card Security using Self-timed Circuits," *In Proc. ASYNC 2002,* IEEE, 2002.

[12] D. Sokolov, J. Murphy, A. Bystrov and A. Yakovlev, "Design and analysis of dual-rail circuits for security applications," *IEEE Trans. Comp.,* Vol 54, No. 4, Apr. 2005, pp. 449-460.

[13] J. Waddle and D. Wagner, "Fault Attacks on Dual-Rail Encoded Systems," *In Proc. ACSAC 2005,* pp. 483-494, 2005.

[14] J. Murphy and A. Yakovlev, "An alternating Spacer AES Cryptoprocessor," *In Proc. ESSIRC 2006,* pp. 126-129, 2006.

[15] http://www.intrinsity.com.

[16] G. Yee and C. Sechen, "Dynamic Logic Synthesis," *In Proc. CICC 1997,* IEEE, 1997.

[17] UK Patent Application No. 0719455.8 'Cryptographic processing and processors'.

[18] M. Morimoto, Y Tanaka, M Nagata and K. Taki, "Logic Synthesis Technique for High Speed Differential Dynamic Logic with Asymmetric Slope Transition," *IEICE Trans. Electron. 2005,* Vol E88-A No. 12, Dec. 2005, pp. 3324-3331.

[19] K. Kim, C Liu and S. Kang, "Implication Graph Based Domino Logic Synthesis," *In Proc. ICCAD 1999,* Nov. 1999, pp. 111-114.

[20] W. Toms, D Edwards and A. Bardsley, "Synthesizing Heterogeneously Encoded Systems," *In Proc. ASYNC 2006,* IEEE, 2006.

[21] C. Paar, "Efficient VLSI Architecture for Bit-Parallel Computation in Galois Fields ," *PhD Thesis,*, Institute for Experimental Mathematics, Unversity of Essen, Germany, 1994.

[22] S. Chantarawong, P. Noo-intara and S. Choomchuay, "An Architecture for S-box Computation in the AES," *Proc. ICEP 2004,* pp. 157-162, 2004.

[23] S. Fenn, M. Benaissa and D. Taylor, "GF(2ˆm) Multiplication and Division Over the Dual Basis," *IEEE Trans. Comp.,* Vol 45, No. 3, Mar. 1996, pp. 319-327.

[24] http://csrc.nist.gov/publications/fips/fips197-197.pdf.