School of Electrical, Electronic and Computer Engineering

Secure Design Flow using 1-of-n Encoding

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

Technical Report Series

NCL-EECE-MSD-TR-2009-146

May 2009

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-2009-146
Copyright © 2009 University of Newcastle Upon Tyne

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

http://async.org.uk/

Secure Design Flow using 1-of-n Encoding

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

Abstract

This paper presents a new design flow for security using 1-of-n encoding. Initially high-level SystemC Galois descriptions are compiled into an intermediate format. The design flow passes though several stages of refinement, including subfield-breakdown and change of basis, to generate small, regular logic blocks. These are converted into 1-of-n represention and subsequently passed to optimization and mapping tools for mapping to a new library of power-balanced components. The new library consists of novel mixed 1-of-2 and 1-of-4 components based on N-nary logic. Finally logic optimization tools are applied to generate secure synchronous circuits for layout generation. The paper shows that the circuits generated are more efficient than those generated by alternative techniques.

Keywords: Security, Dynamic logic, 1-of-n, Power-balancing.

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 that are signature-based [2] and those that are fault-based [3]. These 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.

Targetting secure systems to chips normally requires significant manual design effort. One of the reasons for the lack of commercial tools is the level of complexity of generating secure circuits. Research is therefore needed to develop methodologies and techniques for synthesizing robust cryptographic systems efficiently. Recent work towards creating a VLSI design flow for side-channel attack resistant circuits was carried out in [5]. This was primarily applied at the lower level using a library of differential balanced cells [6] and was targetted towards power-balanced synchronous circuits. In [7] they investigated side-channel attacks at the lower level and concluded the best solution to power analysis is to embed countermeasures into logic cells [8] 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 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 [9]

a level of protection at the fault-level 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 [10]. The power analysis attack for this chip was demonstrated to yield a 40 fold increase in the number of power measurements required to crack the key. 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.

The inefficiency problem of using dual-rail for synchonous power-balanced implementations can be resolved by steering towards alternative 1-of-n circuits [11] which use dynamic logic [12]. Using dynamic logic it is possible to attain significant improvements in area and speed [13][14]. 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. The aim in this paper is to steer away from dual-rail circuits and move towards general N-nary 1-of-n circuits which combine the advantages of using, in addition to 1-of-2 circuits, direct mapping to 1-of-4 circuits [15]. In this paper we explore the Galois design space making use of an efficient 1-of-2, 1-of-4 library and novel 1-of-n mapping techniques to generate efficient encoded power-balanced synchronous security implementations which are more efficient than those generated from dual-rail.

Our new design flow inputs a high-level SystemC Galois specification, optimizes and schedules it, and translates it into an intermediate AND-XOR net representation. The specification then undergoes various stages of refinement including refinement at the subfield level and further refinement using subfield breakdown or basis conversion to generate small regular sub-modules. The sub-modules are then encoded using 1-of-n mapping and mapped to a novel logic library of specially designed power-balanced N-nary 1-of-n gates. The gates from the new library, i.e. implicit-exor, exorhalf-implicit, etc., have been carefully designed to help reduce the area and delay of the implementation. A mapping algorithm is employed which is used to explore the design space to generate an optimal solution. After mapping highly optimized secure synchronous power-balanced circuits are generated.

The remainder of this paper is organised as follows: in section 2 we present the new hardware library; in section 3 we present the design flow mapping methodology; in section 4 we present a synthesis example; in section 5 we provide results; in section 6 we present some conclusions.

2 Secure Gate-level library

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 [11].

(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 func-

tion, 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.

2.1 Basic N-nary gates

Here it is shown 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 1 binary encoded (left), where $\alpha = 10$ and $\beta = 11$, and 1-of-4 encoded (right).

Table 1: GF4 Addition Table

+	0	1	α	β	+	0001	0010	0100	1000
0	0	1	α	β	0001	0001	0010	0100	1000
1	1	0	β	α	0010	0010	0001	1000	0100
α	α	β	0	1	0100	0100	1000	0001	0010
β	β	α	1	0	1000	1000	0100	0010	0001

GF4 has 4 states which can be represented by an encoding scheme with a balanced hamming weight. For 1-of-4 it is assumed the value zero=0001.

Using Table 1 it is possible to derive an N-nary 1-of-4 gate-level implementation which is shown in Fig. 1. 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.

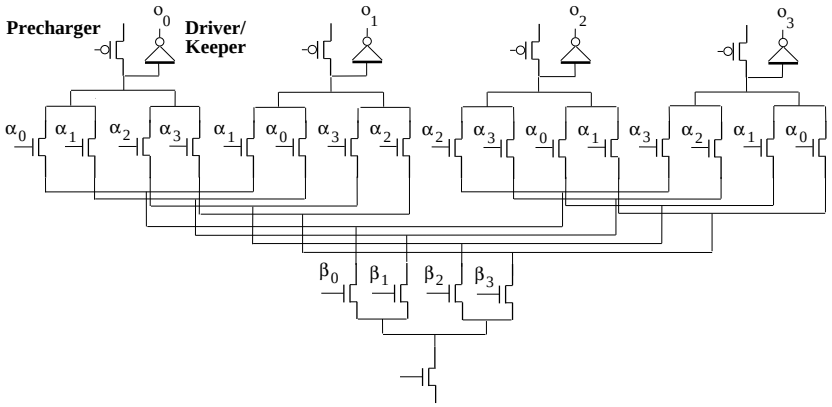

Figure 1: 1-of-4 Adder $\oplus_d$ gate.

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 has its own driver/keeper cell as depicted in Fig. 1.

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 combinations of msb and lsb merge operations is shown in Table 2. 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 2: 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	$(\alpha_0 + \alpha_1) \cdot (\beta_0 + \beta_2)$

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

2.2 Optimized 1-of-n gates

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.

One 1-of-4 gate that exhibits the above features which is presented here is the exor-implicit gate. The exor-implicit gate is shown in Appendix A Fig. 6.

The exor-implicit gate implemements the following binary equations.

$$o_1 = (\alpha_1 \oplus \alpha_0) \tag{1a}$$
$$o_0 = (\beta_1 \oplus \beta_0) \tag{1b}$$

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:

$$o_1 = (a_1 \oplus a_0) \tag{2a}$$
$$o_0 = (b_1 \oplus b_0) \oplus (c_1 \oplus c_0) \tag{2b}$$

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 1 would be as follows:

$$\begin{aligned} o' \quad = \quad & (mergemsblsb(a',\, b' \oplus_d c')) \oplus_d \\ & (mergelsbmsb(a',\, b' \oplus_d c')). \end{aligned} \tag{3}$$

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

$$o' = (a' \oplus_i (b' \oplus_i c'). \tag{4}$$

It is much more efficient, therefore, to represent equations (2a, 2b) using 1-of-4 $\oplus_i$ gates. It is also much easier to route wires into more simple 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 Appendix A Fig. 6 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 Appendix A Fig. 7.

Various combinations of merge gates implemented using $\oplus_i$ are shown in Table 3 with the corresponding input changes.

Table 3: 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	$\alpha_0\alpha_2$	$\alpha_1\alpha_3$	$\beta_0\beta_2$	$\beta_1\beta_3$

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

$$o_1 = a_1 \oplus a_0 \tag{5a}$$
$$o_0 = b_1 \tag{5b}$$

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

Table 4: $\oplus_{hi}$ 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_2	$(\alpha_1 + \alpha_2) \cdot (\beta_0 + \beta_1)$
	o_0	β_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_2	$(\alpha_0 + \alpha_1) \cdot (\beta_1 + \beta_2)$
	o_0	α_1	o_1	$(\alpha_2 + \alpha_3) \cdot (\beta_0 + \beta_3)$
			o_0	$(\alpha_0 + \alpha_1) \cdot (\beta_0 + \beta_3)$

Table 5: Converting $\oplus_i$ to $\oplus_{hi}$ gates

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

$\oplus_{hni}$ gates also have the same structure as the $\oplus_i$ gate but they use different combinations of inputs. Various combinations of $\oplus_{hi}$ gates implemented using $\oplus_i$ are shown in Table 5 with the corresponding input changes.

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 Appendix A Fig. 8. This gate mimics the $\oplus_d$ gate shown in Fig. 1 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.

$$o = (a_1 \cdot b_1) \oplus (a_0 \cdot b_0) \tag{6}$$

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

$$o = (a_1 \cdot b_1 \cdot \bar{a}_0) + (\bar{a}_1 \cdot a_0 \cdot b_0) + (a_1 \cdot \bar{b}_1 \cdot a_0 \cdot b_0) + (a_1 \cdot b_1 \cdot a_0 \cdot \bar{b}_0) \tag{7}$$

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.

$$o_1 = a_1b_1 + a_1b_3 + a_2b_2 + a_2b_3 + a_3b_1 + a_3b_2 \tag{8a}$$

$$o_0 = a_0b_0 + a_0b_1 + a_0b_2 + a_0b_3 + a_1b_0 + a_1b_2 + a_2b_0 + a_2b_1 + a_3b_0 + a_3b_3 \tag{8b}$$

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 gates for o_0 and o_1 are shown in Appendix A in Fig. 9 and Fig. 10.

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

3 Security Design Flow

Fig. 2 gives a block diagram which depicts the design-flow.

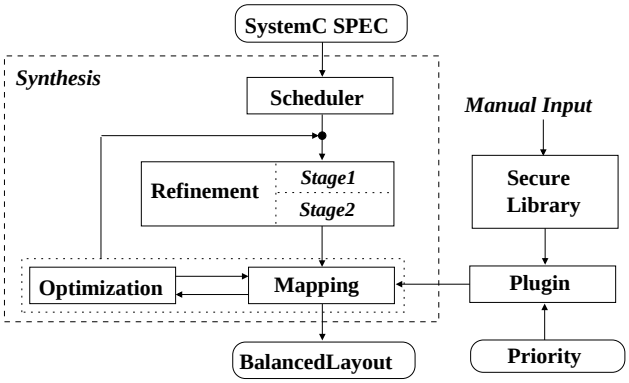

Figure 2: Diagram of design flow.

The security design flow inputs a SystemC Galois 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 to generate small regular blocks. A secure library of components, based on section 2, is accessible as a plugin. The components are mapped to the sub-modules to generate a secure circuit.

3.1 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 [15]. 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.

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 [16] 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. 3. In Fig. 3 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.

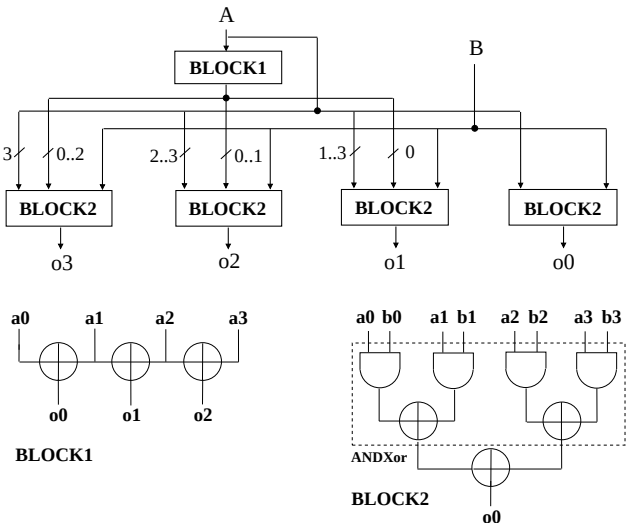

Figure 3: Multiplier Regular Implementation.

3.2 Mapping

The 1-of-n library in section 2 is used for mapping to the refined circuit. The library is split into two divisons: one with basic cells and one with the optimized cells shown in sub-section 2.2. 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-XOR plane targetting to 1-of-4 gates is not so efficient. Therefore, on switching between planes, a heuristic is adopted to convert from 1-of-4 to dual-rail and back again. Here, where possible, trees are reconstructed to find an efficient mapping to complex 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. 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.

4 Design Flow Example

The design flow example centers on the AES S-box [17]. 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 the library.

The design is initially broken down to the first level using subfield generation. The S-box is initially refined to subfield components such as Galois multipliers, affine transformation, etc. Each of the subfield circuit subblocks 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 3.1. The multiplier may be refined to the dual-basis multiplier blocks of Fig. 3, comprising BLOCKS 1 & 2, and these are mapped to 1-of-n components and a 1-of-n circuit generated. A cross-over is made here from 1-of-4 to 1-of-2 over the AND-XOR plane. The diagram showing the complete 1-of-n mapping is shown in Fig. 4.

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

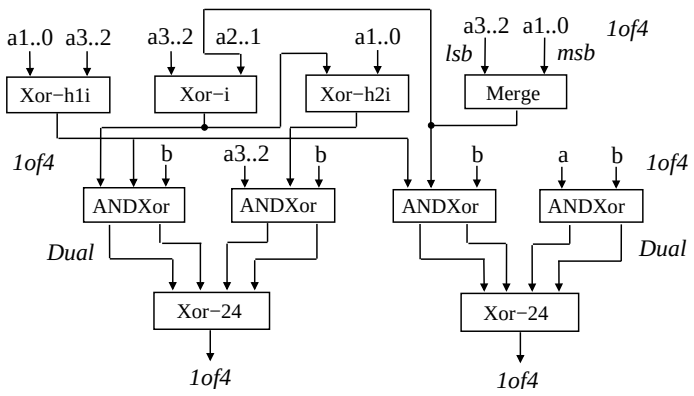

Figure 4: Multiplier 1-of-n implementation.

cells, BLOCK1 is shown implemented in the upper part of Fig. 4 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. 3 and are assumed in Fig. 4 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. 3 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. 4 provide the equivalent 1-of-4 data representation to the 1-of-n AND-XOR cells of BLOCK2.

Using a component priority search in the mapping algorithm 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 Appendix A Fig. 9 and Fig. 10. Each AND-XOR block in Fig. 3 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 Appendix A Fig. 8.

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 [17]. The binary equations for the affine transformation are as follows:

$$\begin{aligned}
o_7 &= a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \\
o_6 &= a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus 1 \\
o_5 &= a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus 1 \\
o_4 &= a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\
o_3 &= a_7 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\
o_2 &= a_7 \oplus a_6 \oplus a_2 \oplus a_1 \oplus a_0 \\
o_1 &= a_7 \oplus a_6 \oplus a_5 \oplus a_1 \oplus a_0 \oplus 1 \\
o_0 &= a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_0 \oplus 1
\end{aligned} \tag{9}$$

These are decomposed using Davio decomposition and 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. 5.

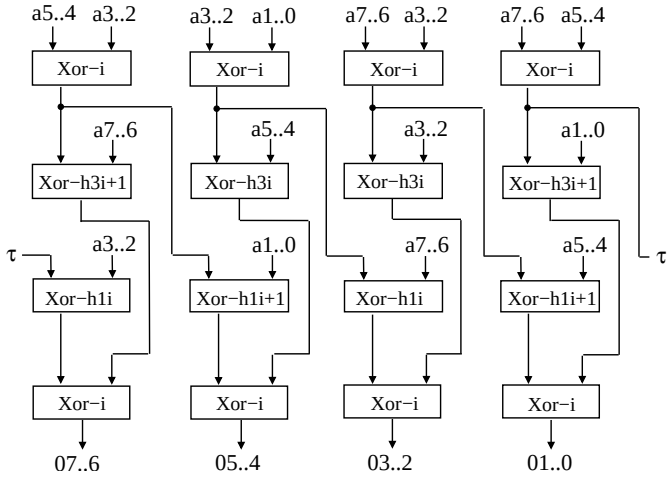

Figure 5: Affine implementation.

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 4. Some of the equations for the affine transformation end in $\oplus 1$. These parts of the equations can be mapped to $\oplus_{hi+1}$ gates which are derivatives of $\oplus_{hi}$ gates. The affine transformation in Fig. 5 is generated using a highly regular layout.

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

5 Results

A detailed set of results including gate-count and transistor-count time comparisons have been made for multipliers at different refinement levels. A similar set of results has been taken for the corresponding S-boxes using similar refinement levels. Tables 6 and 7 show comparions for our synchronous synthesized results over power-balanced results generated for dual-rail cells using a Synopsys 90 nm cell library.

Table 6 shows comparisons for the multipliers at different refinement levels. The results in Table 6 are shown in 6 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-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. Finally the fifth and sixth column shows the area and delay.

In Table 6 it can be seen that at levels L2.1 and L2.2 a small percentage improvement 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

Table 6: Multiplier Comparison

Level	Type	Gates	Transistors	Area	Delay
L1	Dual-rail	31	610	574	0.69ns
	Basic 1-of-n	49	995	968	0.34ns
	Optimized 1-of-n	22	520	488	0.17ns
L2.1	Dual-rail	35	950	812	0.74ns
	Basic 1-of-n	41	897	766	0.33ns
	Optimized 1-of-n	25	545	465	0.19ns
L2.2	Dual-rail	31	610	574	0.69ns
	Basic 1-of-n	30	590	555	0.18ns
	Optimized 1-of-n	14	350	329	0.15ns

Table 7: S-box Comparison

Level	Type	Gates	Transistors	Area	Delay
L1	Dual-rail	188	4360	3934	3.66ns
	Basic 1-of-n	351	6910	6234	1.22ns
	Optimized 1-of-n	155	3145	2837	0.93ns
L2.1	Dual-rail	216	6010	5230	4.44ns
	Basic 1-of-n	313	6468	5628	1.45ns
	Optimized 1-of-n	183	3577	3112	1.18ns
L2.2	Dual-rail	188	4360	3934	3.66ns
	Basic 1-of-n	294	5593	5046	1.04ns
	Optimized 1-of-n	131	2635	2377	0.82ns

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 better 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. A saving in switching is estimated over dual-rail of a fair percentage.

Table 7 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-4 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 7 better results are shown 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.

6 Conclusions

This report has presented a novel way of synthesizing efficient secure circuits. A new design flow using 1-of-n encoding is presented as a means to provide more efficient power-balanced circuits than can be provided by dual-rail alone.

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. The cells are efficient, regular and easy to power-balance and cover both 1-of-2 and 1-of-4. A novel synthesis design flow has been adopted. 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. Preliminary results indicate improvements in area, time and power over dual-rail for optimized 1-of-n gates for similar benefits in security at the synchronous level.

We are looking at ways of improving the layout generation of the circuits. We are also interested in generating more efficient layouts for our implementations.

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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

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

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

[16] 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.

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

A Example gates from library.

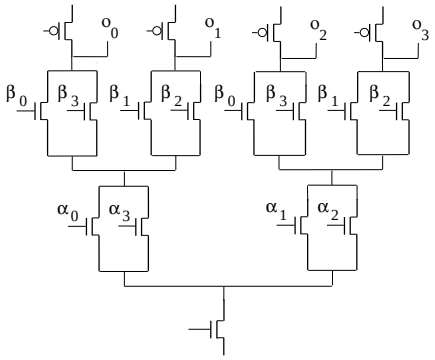

Figure 6: 1-of-4 Exor-implicit $\oplus_i$ gate.

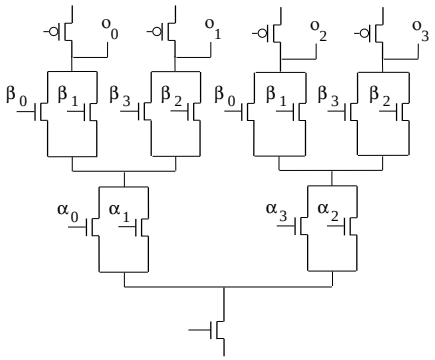

Figure 7: 1-of-4 mergemsbmsb gate.

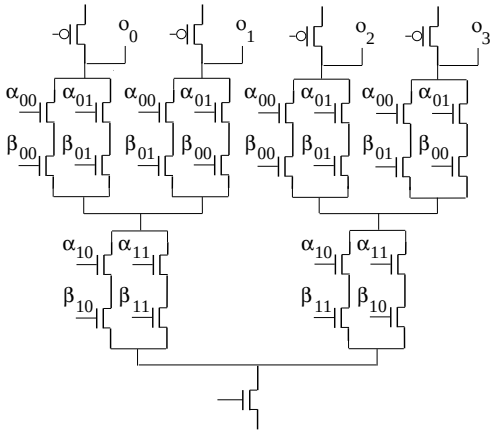

Figure 8: 1-of-2 in, 1-of-4 out Exor $\oplus_{24}$ gate.

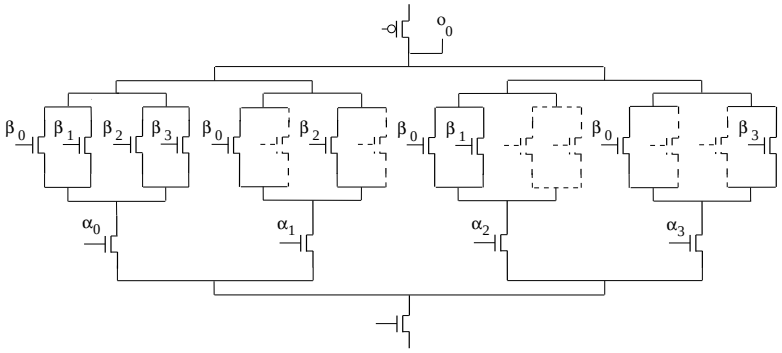

Figure 9: AND-EXOR-half &$\oplus_{42}$ gate 1.

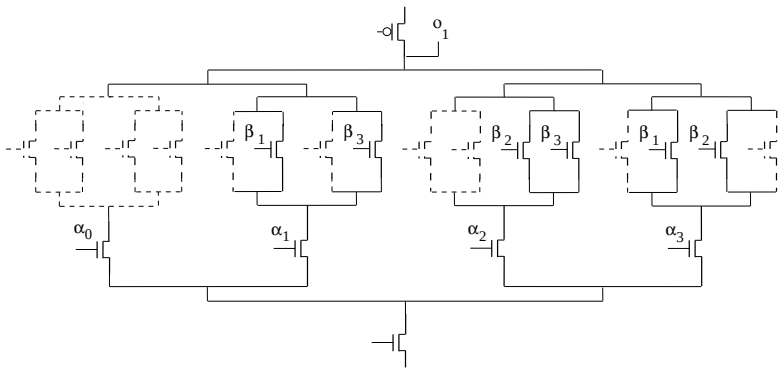

Figure 10: AND-EXOR-half &$\oplus_{42}$ gate 2.