School of Electrical, Electronic & Computer Engineering

Investigating Gate Grouping Algorithms for Mixed Radix Conversion

A. Rafiev, J. Murphy, D. Sokolov, A. Yakovlev

Technical Report Series

NCL-EECE-MSD-TR-2008-132

May 2008

Contact:
ashur.rafiev@ncl.ac.uk
j.p.murphy@ncl.ac.uk
danil.sokolov@ncl.ac.uk
alex.yakovlev@ncl.ac.uk

EPSRC supports this work via EP/F016786/1 (SURE)

NCL-EECE-MSD-TR-2008-132

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

`http://async.org.uk/`

Investigating Gate Grouping Algorithms for Mixed Radix Conversion

A. Rafiev, J. Murphy, D.Sokolov and A. Yakovlev

May 2008

Abstract

A conversion driven design approach is described. It takes the outputs of mature and time-proven EDA synthesis tools to generate mixed radix datapath circuits in an endeavour to investigate the added relative advantages or disadvantages. The application is found in a wide variety and overlapping areas of circuit design. Grouping of signals is defined as a general approach to higher radix conversion driven design. With respect to quaternary signals two basic types of grouping are presented, and algorithms underpinning the approaches are formally described and analysed.

1 Introduction

Single-rail circuits, traditionally used and adopted in conventional EDA flows, have a number of drawbacks with respect to security applications, asynchronous design and network-on-chip communication. These types of circuits have no power balancing, no completion detection and prone to hazards; *m-of-n codes* are known and often cited as a solution [1].

M-of-n codes are an encoding scheme in which data is represented using *n* wires and where *m* of them are set to an active level (usually high). A protocol is used to separate data using dummy signals (spacers) and called *return-to-zero* (RTZ) or *spacer protocol*. M-of-n codes with RTZ protocol are switching-balanced, i.e. have data independent switching of signals. Circuits based on m-of-n codes, typically 1-of-4 or 1-of-2, over the years have been used in a number of areas of electronics. Some specific examples but certainly not exhaustive include: network-on-chip [2], FPGA fabric [3], low power circuits [4], security based circuits [5] and clockless circuits.

1-of-2 (dual-rail) is widely used due to its simple theory and component implementation. However, we can observe 1-of-4 has a halved switching factor compared to that of 1-of-2, which makes it highly desirable if the goal is to minimise switching power, variability[1] (e.g. cross talk) and at the wire-level to have a constant power consumption. For security based circuits, this means the benefit of constant power consumption will be still present but lowered. The encoding of binary data in dual-rail and 1-of-4 is shown in Table 1.

Since the synthesis of 1-of-4 circuits is based on multi-valued logic (MVL) synthesis, it is a rather complex task with little tool support. A number of forward-thinking efforts dedicated to the MVL synthesis

[1] Submicron effects, without indepth silicon experimentation this added advantage cannot be validated, however it is an often cited benefit of radix based circuits [6] and this work has been conducted under this presumption.

Table 1: Dual-rail and 1-of-4 encodings

multi-valued	single-rail (binary)	dual-rail	1-of-4
0	00	01 01	0001
1	01	01 10	0010
2	10	10 01	0100
3	11	10 10	1000
NULL	spacer (NULL)	00 00	0000

have been made in recent years, in particular [7] and [8], but an effective methodology for MVL design is still an open challenge.

Prior to our work a review of the literature revealed a lack of a straightforward means or design flow to construct MVL circuits; on this premise this work was initiated. Our justification and reasoning for the research stems from the following facts:

- Moving away from the RTL design flow is frequently frowned upon by industry;
- Existing EDA tools are mature, known and time-proven;
- MVL synthesis methods employ computationally expensive algorithms instead of reusing the computational power of existing tools.

Having recognised a novel property of binary datapath circuits to facilitate conversion to a mixed radix circuit using a mixture of 1-of-4 and dual-rail, we now suggest a conversion driven design (CDD) approach. The goal of this approach is to achieve a tight integration with the conventional EDA flows with low algorithmic complexity.

As explained in Section 2, signal grouping problem is a key point of the CDD. This report presents a number of approaches to gate grouping together with their analysis and formal definitions of the algorithms. Section 3 introduces two basic types of gate grouping. The following sections 4 and 5 describe implementation and analysis of certain algorithms.

2 Conversion basics

The problem addressed in this report can be characterised as follows. The original single-rail datapath is given as a structural HDL netlist; where datapath is defined as logic gates without registers or combinational loops. The goal of the conversion is to produce an equivalent higher radix circuit.

Since dual-rail is based on binary computations, there is a transparent correspondence between initial specification and the resultant circuit. Conversion to dual-rail can be performed using direct mapping of single-rail gates to dual-rail counterparts [9]. The method was implemented earlier in a set of software tools which interface to conventional EDA tools and form a coherent design flow [10].

Conversion from binary to multi-valued logic can employ grouping of data signals. However, a grouping of all signals in the circuit is not globally efficient, because the original structure usually causes restructuring and splitting of higher radix data. This leads to the use of mixed radix encoding, which means that the circuit becomes partially binary and partially multi-valued (heterogeneous) [8].

In terms of CDD, quaternary logic is of more interest compared to other types of multi-valued logic due to the simplicity of signal grouping by two bits. As it was mentioned in Section 1, the motivation for

this work implies the use of dual-rail and 1-of-4 encodings. However, in general the CDD approach is not based on m-of-n codes and can use different representations of mixed radix data. In this report we use terms *binary* and *quaternary* in order to put aside particular encodings and to generalise conversion algorithms.

A result of the conversion is shown in Figure 1 and consists of binary and quaternary blocks connected through a row of splitters and mixers, which perform signal conversion according to the selected encodings. A splitter is an element which divides quaternary signal into two binary ones. A mixer is an element which merges two binary signals into one quaternary.

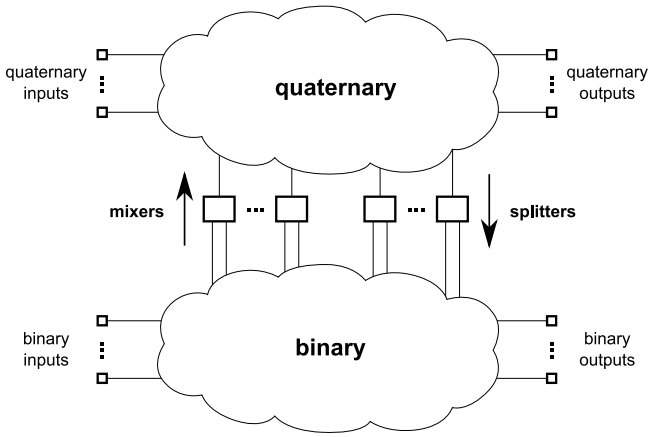


Figure 1: Mixed encoding in a converted combinational logic circuit.

A generic outline for the proposed conversion technology can be described as follows. The algorithm starts with "transferring" gates from binary block to quaternary block by grouping them into pairs. After all possible grouping is done the circuit can be mapped into a final netlist. A mapping is a replacement of technology independent (abstract) binary and quaternary components with real cells using specific encoding and library.

The way the gates are grouped determines the efficiency of the conversion, therefore the conversion problem corresponds directly to the gate grouping problem described in the following sections.

3 Types of gate grouping

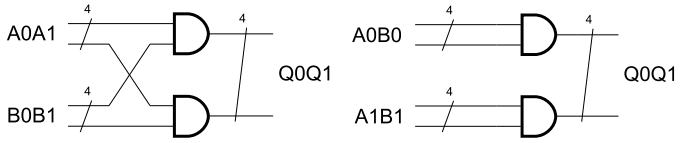
3.1 Bitwise grouping

For 2*n*-bit binary circuits there is an intuition to group higher and lower bits of each signal pair, as shown in Figure 2(a), to form a *n*-signal quaternary circuit. Certain gates which violate bitwise regularity of the original circuit will remain ungrouped forming a binary part of the resultant mixed radix circuit. An algorithm employing bitwise meaning of circuit signals is described in Section 4.

3.2 Operandwise grouping

Let's assume that the original circuit consists of two-input standard cells (AND, OR, XOR gates) and inverters. Usually EDA tools support decomposing of complex gates into standard cells, so this constraint does not complicate design flows. Considering this, it is possible to group inputs of any gate into one

quaternary signal. Outputs of a given pair of gates can also be grouped into a quaternary signal. This produces an *operandwise grouping* of gates illustrated in Figure 2(b).



(a) bitwise (b) operandwise

Figure 2: Types of gate grouping.

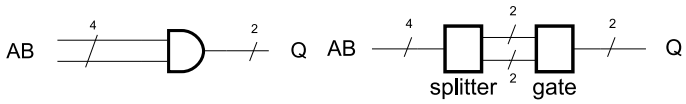
Due to the nature of operandwise grouping, any quaternary signal x can be rewritten as a pair of its original signals, $x = \langle s_1, s_0 \rangle_4$. For two quaternary signals $x = \langle s_1, s_0 \rangle_4$ and $y = \langle t_1, t_0 \rangle_4$, two binary functions A and B can form a quaternary operandwise operation $\langle \mathrm{A}, \mathrm{B} \rangle_4$ shown in (1) and clarified in Table 2.

$$Q_{\mathrm{AB}}(x,y) = \langle \mathrm{A}(s_1, s_0), \mathrm{B}(t_1, t_0) \rangle_4 \tag{1}$$

Table 2: Bitwise and operandwise quaternary operations example

x	y	bitwise AND	operandwise AND-AND
0	0	$0 \wedge 0 = 0$	$\langle 0 \wedge 0, 0 \wedge 0 \rangle_4 = \langle 0,0 \rangle_4 = 0$
0	1	$0 \wedge 1 = 0$	$\langle 0 \wedge 0, 0 \wedge 1 \rangle_4 = \langle 0,0 \rangle_4 = 0$
0	2	$0 \wedge 2 = 0$	$\langle 0 \wedge 0, 1 \wedge 0 \rangle_4 = \langle 0,0 \rangle_4 = 0$
0	3	$0 \wedge 3 = 0$	$\langle 0 \wedge 0, 1 \wedge 1 \rangle_4 = \langle 0,1 \rangle_4 = 1$
1	0	$1 \wedge 0 = 0$	$\langle 0 \wedge 1, 0 \wedge 0 \rangle_4 = \langle 0,0 \rangle_4 = 0$
1	1	$1 \wedge 1 = 1$	$\langle 0 \wedge 1, 0 \wedge 1 \rangle_4 = \langle 0,0 \rangle_4 = 0$
1	2	$1 \wedge 2 = 0$	$\langle 0 \wedge 1, 1 \wedge 0 \rangle_4 = \langle 0,0 \rangle_4 = 0$
1	3	$1 \wedge 3 = 1$	$\langle 0 \wedge 1, 1 \wedge 1 \rangle_4 = \langle 0,1 \rangle_4 = 1$
...	...	...	...
3	0	$3 \wedge 0 = 0$	$\langle 1 \wedge 1, 0 \wedge 0 \rangle_4 = \langle 1,0 \rangle_4 = 2$
3	1	$3 \wedge 1 = 1$	$\langle 1 \wedge 1, 0 \wedge 1 \rangle_4 = \langle 1,0 \rangle_4 = 2$
3	2	$3 \wedge 2 = 2$	$\langle 1 \wedge 1, 1 \wedge 0 \rangle_4 = \langle 1,0 \rangle_4 = 2$
3	3	$3 \wedge 3 = 3$	$\langle 1 \wedge 1, 1 \wedge 1 \rangle_4 = \langle 1,1 \rangle_4 = 3$

In a case when the output of a quaternary gate has to be split again into binary signals an insertion of a splitter can be avoided using incomplete operandwise grouping, i.e. a grouping when gate inputs are grouped, but the output remains binary as shown in Figure 3(a). A *Q/B gate* is a gate with a quaternary input and a binary output. Due to their semantical meaning clarified in Figure 3(b), Q/B gates can eliminate unnecessary splitters during the conversion.



(a) Q/B gate as an incomplete operandwise group (b) structural decomposition

Figure 3: Understanding Q/B gates.

The operandwise grouping suggests a binary trees approach described in Section 5.

4 Grouping based on bitwise regularity

Let's consider a binary circuit, which perform calculations on 2-bit values. Assuming that separate calculations of both bits are relatively similar, one can determine for any gate in the lower bit part its equivalent in the higher bit part. A distinguished pair of such relative gates can form a quaternary gate. For n-bit binary circuits, if $n > 2$, gates can be grouped in similar way but merging even and odd bit parts of the circuit. Parts of the circuit that cannot be distributed between certain bit parts remain binary.

Although the intuition behind the bitwise approach is straight-forward, automatically distinguishing even and odd bit parts of the given netlist is computationally complex or, in certain cases, infeasible. For example, S-box circuits [11, 12] tend to reshuffle input data, thus input signals have no bitwise meaning. However, circuits displaying bitwise regularity can be converted using this approach if the information on the bitwise meaning of the input and output signals of the datapath is supplied, i.e. input and output ports are initially grouped into pairs. It is possible to automate port grouping using a naming convention.

The algorithm shown in Algorithm 1 implies bitwise gate grouping using given netlist and port grouping information. Here G is defined as a set of binary gates; initially it contains all gates of the original circuit. Each gate $g \in G$ has a preset $I(g) = \{i_0(g), \ldots, i_{n-1}(g)\}$, i.e. other gates or circuit ports connected to the inputs of g; n is the number of inputs of g. P is a set of grouped gates or circuit ports (pairs). Since port grouping specification is given, P initially contains paired circuit ports. Please note that the order of items in a bitwise pair is important.

Algorithm 1 Grouping based on bitwise regularity

```
all ports are assumed to be already grouped
repeat:
    υmax = 0
    N = size of G
    for i = 0 to N − 2:
        for j = i + 1 to N − 1:
            if gi ∈ I(gj) or gj ∈ I(gi): skip this pair
            υ = regularity ratio for group {gi, gj}
            if υ > υmax:
                υmax = υ
                pmax = {gi, gj}
            end if
        end for
    end for
    if υmax > 0:
        add pmax to P
        remove gates in pmax from G
    end if
until there are no more pairs with u > 0
```

Bitwise regularity ratio (BRR) υ for the given group of gates is a characteristic showing how many quaternary links the group can form w.r.t. current state of P. In other words, for a bitwise group $p = \{g_1, g_2\}$ BRR can be calculated as follows:

$$\upsilon(p) = \frac{\upsilon_{out} + \sum_{j=1}^{n} \upsilon_j}{1+n}$$

where n is a number of gate inputs, $n = n(g_1) = n(g_2)$, and

$$\upsilon_{out} = \begin{cases} 1, & \text{if there exist such } k \text{ and } \{e_1, e_2\} \in P \text{ that } g_1 = i_k(e_1), g_2 = i_k(e_2) \\ 0, & \text{otherwise} \end{cases}$$

$$\upsilon_j = \begin{cases} 1, & \text{if there exist such } \{e_1, e_2\} \in P \text{ that } e_1 = i_j(g_1), e_2 = i_j(g_2) \\ 0, & \text{otherwise} \end{cases}$$

In these equations we assume that gates g_1 and g_2 are similar, i.e. they represent the same function and have equal number of inputs n. Considering similarity of bitwise circuit parts the gates corresponding to the same operation are also supposed to be similar. However, this constraint is not essential if we have a methodology of grouping non-similar gates.

BRR is used as estimation criterion in breadth-first search. As each iteration of outer loop in Algorithm 1 adds new pair in P, BRRs of gate pairs change over time. If the search reveals several maximum regular pairs, the algorithm uses the first encountered one instead of searching through all possible varieties; this feature is treated as disadvantage which in certain cases can lead to inefficient results.

The described algorithm has polynomial complexity $O(N) = \frac{1}{2}N^2 \log_2 N$ not considering calculations of BRR, where N is the number of binary gates in the initial circuit. BRR calculation for one pair has a complexity $O(n,M) = nM + nM$, where n is an average number of gate inputs and M is a size of P at the moment. For $n = 2$ and linearly growing P we have total algorithmic complexity $O(N) = 2N^2 \log_2^2 N$. In terms of CDD this computational cost is rather expensive; VLSI design presumably requires conversion of datapath blocks with $N > 500$.

Consider a 2-bit full adder shown in Figure 4. Initially P consists of pairs {A0, A1}, {B0, B1}, {Q0, Q1} due to the bitwise meaning of port signals. Ports C and CC have no pairs and remain dual-rail. Conversion using Algorithm 1 can be done in the following steps:

Iteration 0 maximum regular pair is {g00, g01} – all inputs can form quaternary links with input ports.

Iteration 1 maximum regular pair is {g10, g11} – the same reason.

Iteration 2 {g20, g21} – it can form a quaternary connections with {g00, g01} and with output port {Q0, Q1}.

Iteration 3 {g30, g31} – it can form a quaternary connection with {g00, g01}.

Iteration 4 {g40, g41} – it can form a quaternary connection with {g10, g11}.

There are no unpaired gates remaining. Please note that each iteration here is the iteration of outer loop, which in its turn searches through $4N \log_2^2 N$ iterations. Resultant circuit after insertion of signal converters is shown in Figure 5.

The described example demonstrates another drawback of the bitwise approach: it can form combinational loops in the resultant circuit. Without special consideration and additional completion detection RTZ-aware components [13] cause a spacer deadlock in these loops. Consider the mixer in Figure 5. Assume that the whole circuit is initially reset to spacer value. Incoming data from the port C cannot pass through the mixer because a spacer from the looped wire (an internal carry) blocks it. On the other hand,

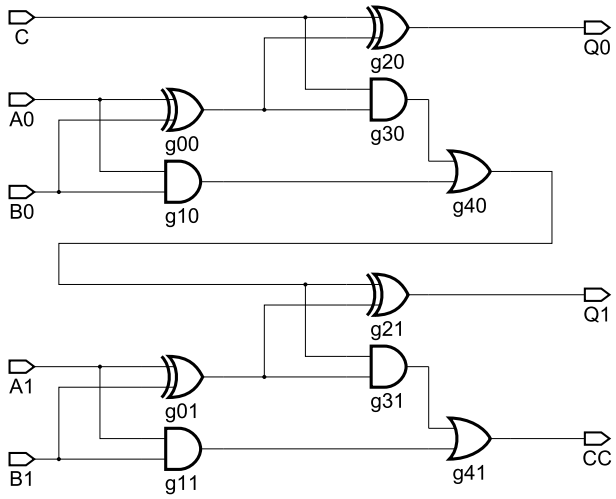


Figure 4: Example original single-rail circuit: 2-bit adder.

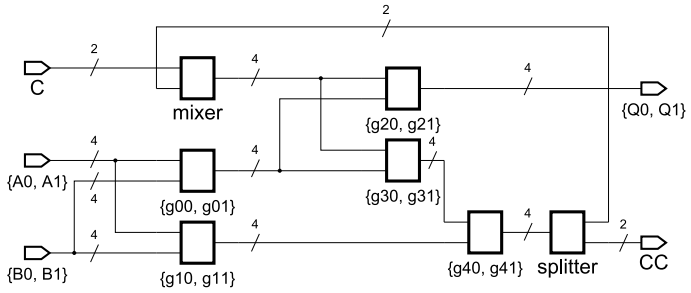


Figure 5: 2-bit adder converted using bitwise regularity approach; gates are shown as “black boxes”.

if the mixer uses a protocol allowing the data to propagate regardless to spacers, the circuit will produce invalid data output because a valid result is available only after the second iteration.

Although a number disadvantages revealed in the approach, the grouping based on bitwise regularity shows several positive features:

- Port bits are not reshuffled with respect to bitwise meaning of input and output signals, which is useful from the large scale design view.
- Resultant circuits have a structure similar to the original, which allows using placement suggestions from the single-rail equivalent.

5 Grouping based on binary trees

The operandwise grouping suggests a binary trees approach considering gates of the circuit to be tree-nodes and their inputs to be child branches. As it was mentioned before, the given datapath circuit contains no loops. However, a pure tree-like structure can be blocked by gates with multiple fanout. The tree size can be reduced by recursive operandwise grouping of child nodes for each gate in binary trees within the circuit. This grouping causes all signals in tree-like structures to become 1-of-4 encoded, but “blocked” parts of the circuit remain binary and go to dual-rail.

Formally, a circuit is considered to be a set of entities E; each entity has a type of *input*, *output*, or *gate*. Input and output entities are circuit ports. Each non-input entity $e \in E$ has a preset $I(e) = \{i_0(e), \ldots, i_{n-1}(e)\}$, i.e. entities connected to the inputs of e. Due to declared constraints $n = 2$ for gates, and $n = 1$ for outputs. Set P is a set of gate groups (pairs). It represents 1-of-4 encoded part of the circuit. In addition there is a parameter θ_e that stands for encoding of entity $e \in E$ and can be either dual-rail or 1-of-4. It is used for automated insertion of signal converters (splitters and mixers) into the final circuit.

The proposed algorithm contains three phases as shown in Algorithm 2. To avoid recursion the grouping is done in two search passes through E (phase 1 and 2). The first phase ignores gate fanouts and groups all signals considering the whole circuit as a binary tree. This leads to duplication of certain gates. The second phase analyses the duplicates and discards the groups which lead to duplication. The last phase reconstructs correctness of links between gates by inserting signal converters where appropriate. Splitters are reduced to Q/B gates.

There is no simple solution to estimate optimal grouping of outputs but to search through all possibilities. However, due to the nature of binary trees approach the grouping of outputs has minor influence to the structure comparing with the whole circuit. Therefore a suggestion to group any pair of outputs is accepted.

Algorithm 2 Conversion based on binary trees

Phase 1: group all gate inputs.

```
for each gate g in E:
    if i0(g) and i1(g) are of the same type:
        group {i0(g), i1(g)} and add to P
end for
```

Phase 2: discard groups containing gates with fanout > 1.

```
for each non-output entity e in E:
    u = how many groups share e
    if u > 1: remove groups containing e
    else if u = 1: set θe to 1-of-4
    else set θe to dual-rail
    end if
    if e is gate and group {i0(g), i1(g)} ∉ P:
        remove all groups containing i0(g) or i1(g)
end for
```

Phase 3: insert mixers and Q/B gates.

```
for each gate g in E:
    if θi(g) is 1-of-4 and θg is dual-rail:
        set type of g to Q/B gate
    else if θi(g) is dual-rail and θg is 1-of-4:
        insert mixer before g
end for
```

The computational complexity of the algorithm is linear, $O(N) = 3N$, where N is the size of E. The algorithm is highly modular; one can add more passes to the algorithm to increase efficiency of the conversion. However it can produce significant "fractioning" of dual-rail and 1-of-4 parts of the circuit increasing

the number of signal converters required.

Consider a 2-bit adder shown in Figure 4. Preset gates of the gate g40 can form the group {g30, g10}; similarly ports A0 and B0 are grouped as inputs to g00 or g10. Considering all gates we can make the following grouping: {A0, B0}, {A1, B1}, {g30, g10}, {g31, g11}, {g40, g01}. Preset of gates g20 and g30 cannot be grouped because of the different entity types (input C cannot be grouped with the gate g00). The second phase should cancel signal duplicating gate groups, but in this case nothing is to be cancelled. Indeed, in spite of the fact that some gates and ports have fanout > 1, they are not shared between different groups and do not lead to signal duplication. Finally, randomly selected output grouping {Q0, Q1} leads to the grouping {g20, g21}. Resultant circuit is shown in Figure 6. Gates g00 and g41 are Q/B gates.

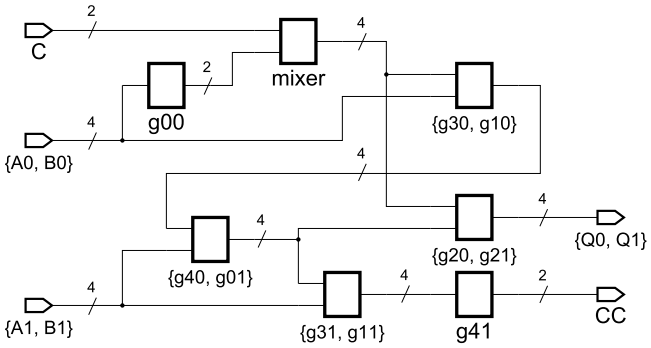


Figure 6: 2-bit adder converted using binary trees approach; gates are shown as “black boxes”.

6 Conclusions

Two algorithms supporting the CDD approach were described, but it is not an exhaustive number of solutions. Possible alternatives can be developed modifying weighted graph search algorithms.

Among the investigated approaches the operandwise grouping algorithm looks more attractive due to its simplicity, lower computational cost and relatively increased efficiency for security based examples. Bitwise grouping, in its turn, revealed a number of bottlenecks in the conversion algorithm and was disregarded for security applied CDD.

Gate grouping algorithms work with technology independent (abstract) mixed radix components implying the component level of abstraction in addition to design level, gate level, and transistor level. Optimal component implementation is the next important challenge in the conversion driven design approach.

References

[1] T. Verhoeff, “Delay-insensitive codes – an overview,” *Distributed Computing*, no. 3, pp. 1–8, 1988.

[2] W. Bainbridge, W. Toms, D. Edwards, and S. Furber, “Delay-insensitive, point-to-point interconnect using m-of-n codes,” in *Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC’03)*, 2003.

[3] J. Teifel and R. Manohar, “An asynchronous dataflow FPGA architecture,” *IEEE Trans. Comput.*, vol. 53, no. 11, pp. 1376–1392, 2004.

[4] T. Takahashi and T. Hanyu, "Multiple-valued multiple-rail encoding scheme for low-power asynchronous communication," in *ISMVL '04: Proceedings of the 34th International Symposium on Multiple-Valued Logic*, pp. 20–25, 2004.

[5] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving smart card security using self-timed circuits," *Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on*, pp. 211–218, 8-11 April 2002.

[6] US Patent 6202194, "Method and apparatus for routing 1 of N signals."

[7] M. Gao, J.-H. Jiang, Y. Jiang, Y. Li, S. Sinha, and R. Brayton, "MVSIS," in *Notes of the International Workshop on Logic Synthesis*, June 2001.

[8] W. B. Toms, D. A. Edwards, and A. Bardsley, "Synthesising heterogeneously encoded systems," in *Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC '06)*, 2006.

[9] A. Kondratyev and K. Lwin, "Design of asynchronous circuits using synchronous CAD tools," *IEEE Des. Test*, vol. 19, no. 4, pp. 107–117, 2002.

[10] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Design and analysis of dual-rail circuits for security applications," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 449–460, 2005.

[11] *Specification for the Advanced Encryption Standard (AES)*, Nov 26, 2001. Federal Information Processing Standards Publication 197.

[12] *3GPP Technical Specification 35.202*, 2001. v3.1.1.

[13] Y. Zhou, D. Sokolov, and A. Yakovlev, "Cost-aware synthesis of asynchronous circuits based on partial acknowledgement," in *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pp. 158–163, 2006.