A Robust Asynchronous Early Output Full Adder

PADMANABHAN BALASUBRAMANIAN
School of Computer Science
The University of Manchester
Oxford Road, Manchester M13 9PL
UNITED KINGDOM
padmanab@cs.man.ac.uk, spbalan04@gmail.com

Abstract: - A robust asynchronous full adder design corresponding to early output logic, synthesized using the elements of a standard cell library is presented in this paper. As the name suggests, the adder ensures gate orphan freedom and neatly fits into the self-timed system architecture. In comparison with many of the indicating full adder designs, which can be embedded in the self-timed system, it is found that the proposed full adder enables reduction in latency by 20.7%, occupies lesser area by 15.4% and features minimized average power dissipation by 8.6% against the best design metrics of its counterparts. These design estimates correspond to simulation results of the 32-bit carry-ripple adder circuit; derived by targeting a high-speed 130nm bulk CMOS process technology. Also, the proposed full adder facilitates a faster reset and the return-to-zero for the fundamental carry-propagate topology is achieved with only two full adder delays.

Key-Words: - Full adder, Asynchronous design, Early propagation, Indication, Standard cells, CMOS process

1 Introduction

Reliability has been labelled as one of the five crosscutting design challenges in the Semiconductor Industry Association's ITRS 2008 report on design [1], which drives home the point that 'design robustness' is becoming an increasing priority for digital logic design in ultra deep submicron technologies. In this scenario, self-timed design attracts attention due to its inherent ability to tolerate supply voltage, process parameter and temperature variations [2]. Due to the absence of a global clock reference, robust asynchronous circuits tend to have better noise and electro-magnetic compatibility properties compared to their synchronous counterparts [3]. In addition, they feature greater modularity permitting convenient design reuse [4], which is all the more important since design reuse as a percentage of overall logic is expected to account for 55% by 2020 [1].

This paper deals with a high-speed, low area and low power robust asynchronous realization of a basic arithmetic component, viz. the full adder. The rest of this paper is organized as follows. Section 2 briefly summarizes the various timing models adopted, discusses the attributes of a function block and explains a widely preferred robust asynchronous signaling convention: the 4-phase handshaking. Also, the full adder designs corresponding to various self-timed design methods are highlighted in this section, accompanied with a brief discussion about the same. Section 3 illustrates the problem of circuit orphans describing how they undermine the robustness of an asynchronous circuit. The robust asynchronous full adder design put forward in this work, based on the concept of early output logic, is then presented and its properties are described. It is shown how the adder is easily embedded into the standard self-timed system topology without affecting the latter's characteristics. The simulation results corresponding to various 32-bit self-timed carry-ripple adders are given in Section 4, and the conclusions are arrived at in Section 5.

2 Fundamentals of Input/Output Mode Circuits

2.1 Timing Models

The following circuit models adhere to input/output mode, with no timing assumptions on when the environment should respond to the circuit.

- Delay-Insensitive (DI)
- Quasi-Delay-Insensitive (QDI)
- Speed-Independent (SI)

A DI circuit guarantees correct normal operation irrespective of the delays of its gates and the delays encountered in the communicating signal wires, i.e. unbounded (arbitrary, but positive and finite) gate delay and wire delay models are considered. This is the most robust of all the unbounded delay models; such circuits are guaranteed to be *correct by construction*. It was shown in [5] that C-elements and inverters are the only DI elements and so, unfortunately, the class of pure DI circuits would be very limited when considering only these two logical operators.

DI circuits with isochronic fork assumptions [5] are referred to as QDI circuits; it is not necessary that every fork be an isochronic fork in a QDI circuit. The isochronic fork assumption has been defined in [5] as follows: "*In an isochronic fork, when a transition on one output is acknowledged, and thus completed, the transitions on all outputs are acknowledged, and thus completed*". A recent work by Martin et al. [6] shows that the main building blocks of QDI logic, including realization of the isochronicity assumption, can be successfully implemented even in nano-CMOS technologies where stricter design rules and larger parametric variations could be anticipated. This is an encouraging pointer towards the feasibility of this approach in the nano-CMOS era. Similar to the DI circuit, the QDI circuit conforms to the unbounded delay model for gates and wires, but with the exclusion of isochronic forks.

A SI circuit operates correctly regardless of gate delays; wires are assumed to have no or negligible delay – hence, unbounded gate delay and bounded wire delay. Every fork is assumed to be an isochronic fork in a SI circuit. Technically, wire delays are typically accounted for in the components (gates) according to this timing model, and subsequently wires are assumed to be ideal (i.e. zero delay).

Referring to the circuit fragment in figure 1(a), d_{g1}, d_{g2} and d_{g3} represent the propagation delay of gates $g1$, $g2$ and $g3$ respectively, while d_{w1}, d_{w2} and d_{w3} signify the delay values of the corresponding nets. For the DI delay model, d_{g1}, d_{g2}, d_{g3}, d_{w1}, d_{w2} and d_{w3} can be arbitrary, while in case of the QDI delay model; d_{w2} is assumed to be equal to d_{w3} with f being an isochronic fork junction. For the SI delay model, d_{w1} = d_{w2} = d_{w3} = 0, but the wire delays are accounted for in the delay of gate $g1$, whose output acts as input for gates $g2$ and $g3$. Hence, the delay of gate $g1$ is modeled as $d_{g1}+d_{w1}+d_{w2}$ or $d_{g1}+d_{w1}+d_{w3}$, as shown in figure 1(b).

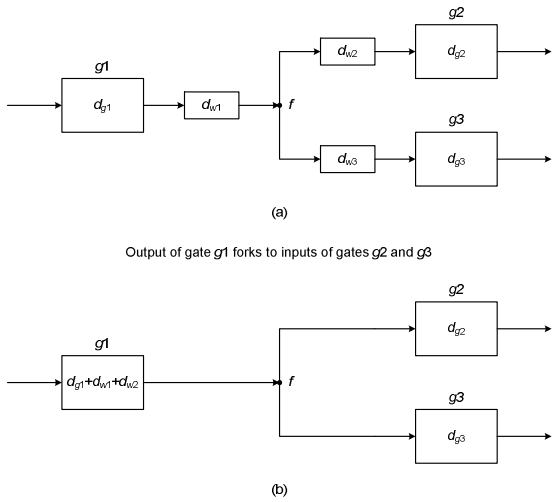

Fig. 1. Illustration of DI, QDI and SI delay models

2.2 Function Block – Characterization

Seitz classified the 'function block', which is the asynchronous equivalent of a synchronous combinational logic circuit, into two basic robust categories depending on their indicating mechanism as either *strongly indicating* or *weakly indicating* [7]. A strong-indication function block waits for all of its inputs (valid/spacer) to arrive before it starts to compute and produce any output (valid/spacer). A weak-indication function block starts to compute and produce outputs (valid/spacer) even with a subset of the inputs (valid/spacer). However, Seitz's weak timing specifications require that at least one output (valid/spacer) should not have been produced until after all inputs (valid/spacer) have arrived. The signaling scheme for strong and weak-indication timing regimes in terms of their input and output behavior is illustrated graphically in figure 2.

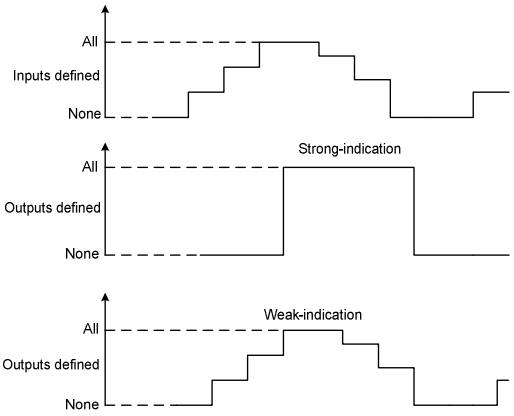

Fig. 2. Depicting strong and weak-indication phenomena

Many robust function block designs adhere to a 4-phase handshaking convention for simplicity of implementation and can employ any DI data-encoding scheme, with the dual-rail data-encoding scheme being widely preferred. Each data wire x is represented using two data wires, x^0 and x^1, with the request signal embedded within the data wires. A low-to-high transition on the x^0 wire indicates that a *zero* has been transmitted, while a low-to-high transition on the x^1 wire indicates that a *one* has been transmitted. Since the request is embedded within the data wires, a transition on either x^0 or x^1 informs the receiver about the validity of the data. The condition of both x^0 and x^1 being a zero at the same time is referred to as the *spacer* (empty data). Both x^0 and x^1 are not allowed to transition simultaneously as it is illegal and invalid, since the coding scheme is *unordered*, i.e. no code word should be a subset of another code word.

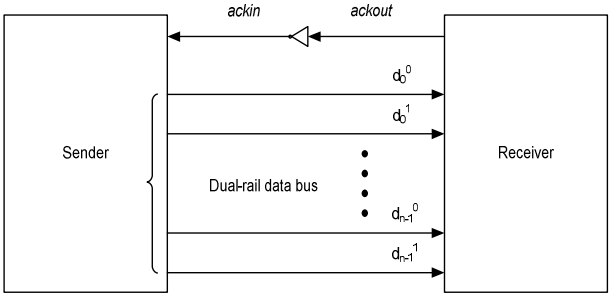

Fig. 3. Dual-rail data encoding and 4-phase handshaking convention

With reference to the above figure, the 4-phase handshake protocol is explained as follows[1]:

- The dual-rail data bus is initially in the spacer state. The sender transmits the code word (valid data). This results in 'low' to 'high' transitions on the bus wires, which correspond to non-zero bits of the codeword
- After the receiver receives the codeword, it drives the *ackout* (*ackin*) wire 'high' ('low')
- The sender waits for the *ackin* to go 'low' and then resets the data bus (i.e. spacer state)
- After an unbounded, but finite (positive) amount of time, the receiver drives the *ackout* (*ackin*) wire 'low' ('high'). A single transaction is now said to be complete and the system is ready to resume the next transaction

[1] The explanation remains valid for data representation using any DI data-encoding scheme.

2.3 Self-Timed Full Adder Designs

The strong and weak-indication realizations of a full adder based on Seitz's approach [7] are portrayed by figures 4 and 5 respectively. The basic equations governing a full adder with dual-rail inputs ($a1$, $a0$), ($b1$, $b0$), ($cin1$, $cin0$) and dual-rail outputs ($Sum1$, $Sum0$), ($Cout1$, $Cout0$) are given below.

$$Sum1 = a0b0cin1 + a0b1cin0 + a1b0cin0 + a1b1cin1 \quad (1)$$

$$Sum0 = a0b0cin0 + a0b1cin1 + a1b0cin1 + a1b1cin0 \quad (2)$$

$$Cout1 = a0b1cin1 + a1b0cin1 + a1b1cin0 + a1b1cin1 \quad (3)$$

$$Cout0 = a0b0cin0 + a0b0cin1 + a0b1cin0 + a1b0cin0 \quad (4)$$

The weak-indication adder takes advantage of the fact that the output carry of an adder module could become defined as soon as its input operands become defined. This depends on the occurrence of carry-kill ($a0$=$b0$=1) or carry-generate ($a1$=$b1$=1) conditions. Thus the carry output equations can be expressed as,

$$Cout1 = a0b1cin1 + a1b0cin1 + a1b1 \quad (5)$$

$$Cout0 = a0b1cin0 + a1b0cin0 + a0b0 \quad (6)$$

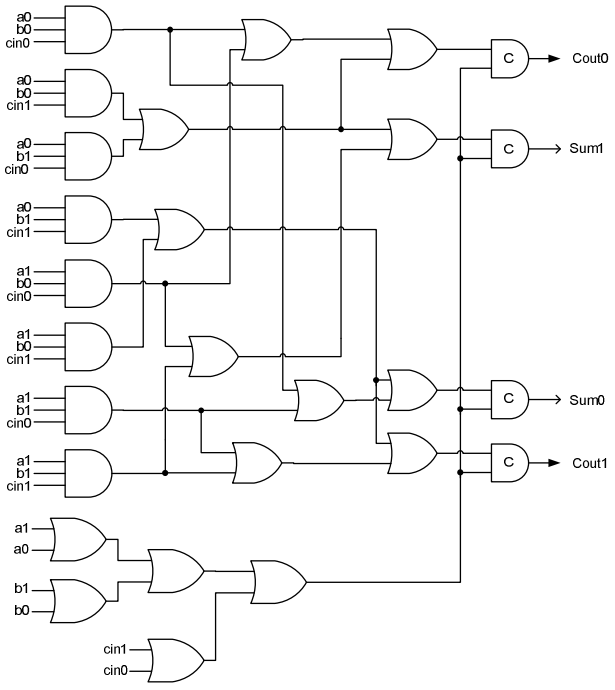

Fig. 4. Seitz's strong-indication full adder [7]

Comparing the adder circuitry depicted in figures 4 and 5, it can be noticed that the strong-indication adder increases the datapath delay, while the weak-indication adder incorporates a fast carry propagation path as only the sum output depends on all the inputs, while the carry output do not always depend, i.e. the indication is distributed between the adder outputs in case of the latter realization. It is to be noted that the completion detection logic, when not implemented as a single OR gate due to fan-in restrictions of the cell library, would necessitate timing assumptions.

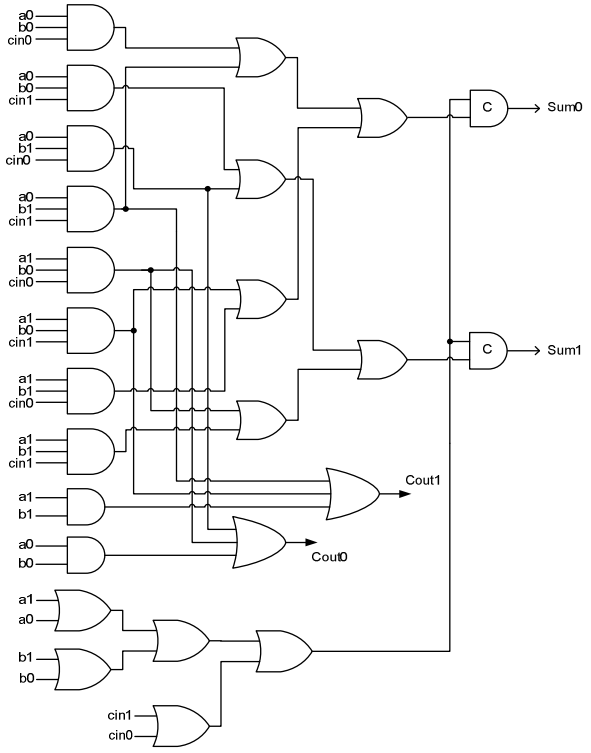

Fig. 5. Seitz's weak-indication full adder [7]

The strong-indication full adder realized according to the Delay-Insensitive Minterm Synthesis (DIMS) technique [9] is shown in figure 6. Isochronic fork assumptions are made with regard to the primary inputs as they feed many C-gates[2], while the forks that feed the OR-gates need not be isochronic. Hence, the DIMS method corresponds to the QDI timing model. Similar to the case of Seitz's weak-indication adder, the carry-kill and carry-generate conditions can be utilized to make the adder weakly indicating, as shown in figure 7.

[2] The C-gate is a *rendezvous* element. It output a 1(0), when all its inputs are 1(0); otherwise it retains its existing steady state. The C-element is represented by the AND gate symbol with the marking 'C' on its periphery.

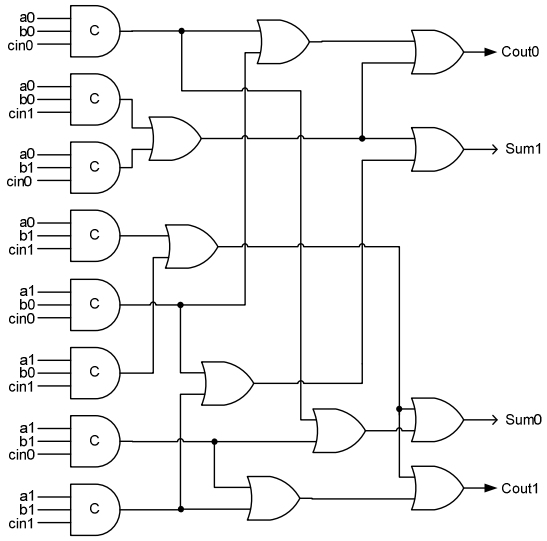

Fig. 6. Strong-indication full adder realization based on DIMS method [9]

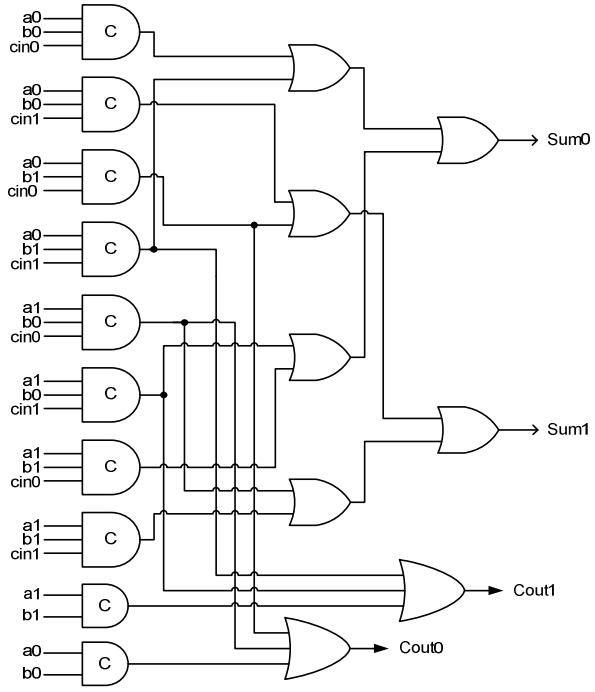

Fig. 7. Weak-indication full adder implementation on the basis of DIMS method [9]

The strong and weak-indication realizations of the full adder block, derived on the basis of the methods proposed in [10] and [13] are portrayed through figures 8 and 9 respectively. Strong-indication adders are governed by worst-case latency, whereas weak-indication and early output adders facilitate data-dependent computation [15]. The weak-indication and

early propagative adders generally outperform the strongly indicating adders in terms of latency and throughput on account of the carry-generate and carry-kill scenarios. However, this is not found to be the case with Toms et al.'s adders, since the critical path of the weak-indication adder [13] would be traversed through OR2, CE2 and OR3 elements as opposed to CE2 and two OR2 gates for the strong-indication version [10][3]. The weak-indication version has been constructed adopting the de-synchronization approach [16] [17] and takes a cue from the asynchronous circuit synthesis method based on partial acknowledgement [18].

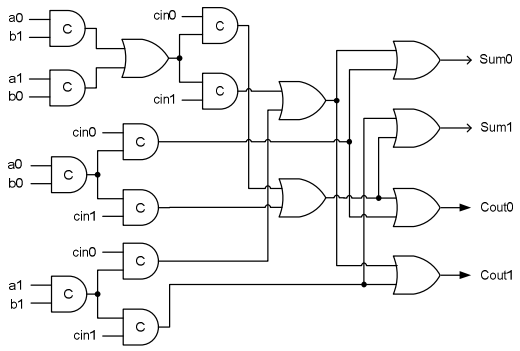

Fig. 8. Strong-indication full adder based on [10]

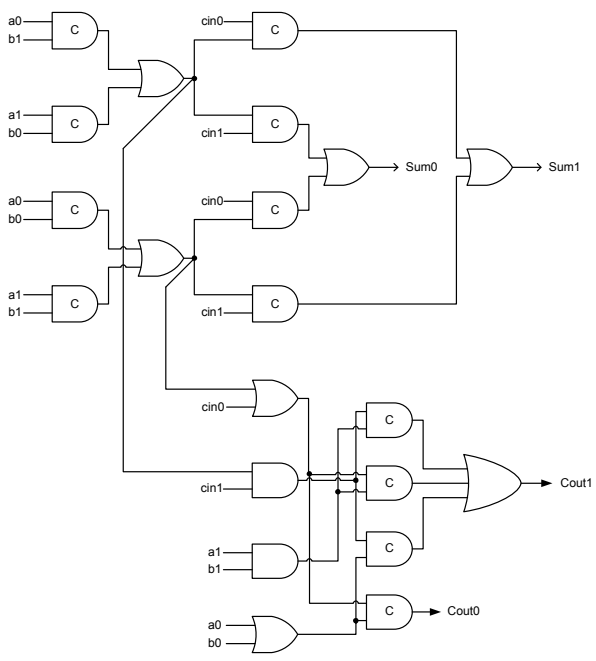

Fig. 9. Weak-indication full adder based on [13]

[3] The standard cell library consists of AND gates and OR gates with a maximum fan-in of 4 and 3 respectively.

Folco et al.'s approach [11] bears a similarity with the previous approach in the sense that the synthesis of combinational logic as QDI circuits is initially performed using 2-input C-elements and OR gates, with the resulting circuits satisfying either strong or weak-indication constraints [7]. This approach makes use of algorithms for constructing reduced ordered binary decision diagrams [19] as the basis for its synthesis strategy with the exception that logical conjunctions are perceived as achieved through C-gates rather than AND gates. The technology-mapping step is subsequently performed targeting a cell library [20] that includes custom asynchronous elements developed on the basis of the STMicroelectronics bulk CMOS process. The technology mapping process actually follows the structural pattern-matching algorithm originally proposed by Zhao et al. in [21]. The full adder circuit implemented using this approach taking into account the shared logic is shown in figure 10. It can be observed that while the sum outputs depend on all the inputs for evaluation, the carry outputs need not as they utilize the carry-kill and carry-generate conditions. Thus the full adder adheres to weak-indication constraints.

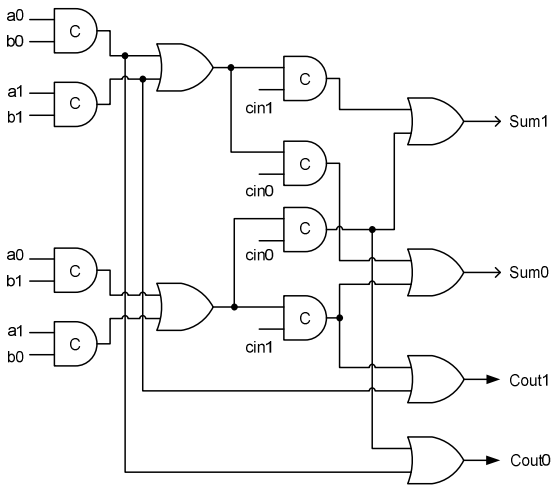

Fig. 10. Full adder synthesized using Folco et al.'s approach [11]

Our earlier full adder design (referred to as the SSSC_adder – single-sum, single-carry adder based on dual-rail encoding) is shown in figure 11. It was designed in a semi-custom style [12] to investigate two important issues: i) how the responsibility of indication can be confined to the sum output alone, thereby freeing the carry signal from indication constraints, and ii) how logic redundancy can be made implicit in a self-timed design to enable reduction in

latency. It can be noticed that the sum output is strongly indicating, while the output carry is eager and is realized by means of a complex gate, viz. AO222 cell. Even with the arrival of any subset of the inputs, the carry outputs could become defined/undefined, while the sum outputs would wait for the arrival of all the inputs to become defined/undefined. Thus the full adder satisfies Seitz's weak-indication timing constraints [7], but without resorting to distribution of inputs indication unlike the conventional approach. This style of implementation was later labelled as the biased approach in [22], targeting self-timed logic circuit synthesis at a block level. However, our method differs from this approach in that the latter relies upon dual-rail combinational logic (DRCL) style as the basis for realizing those outputs, which are selected as candidates for relaxation.

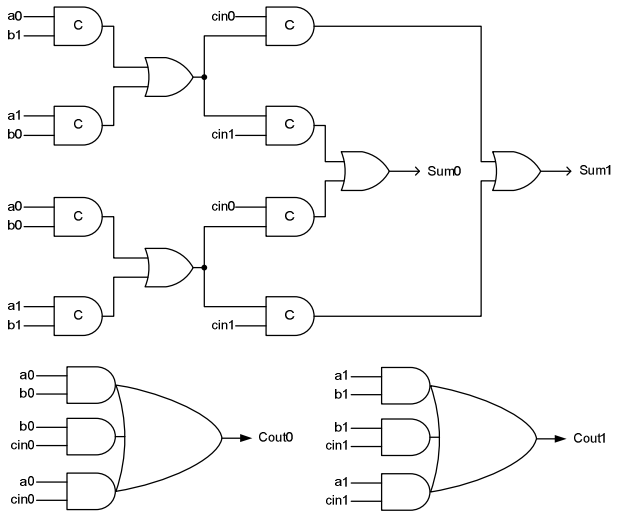

Fig. 11. Weak-indication full adder block [12]

Reference [8] deals with an approach to realize arbitrary combinational logic as asynchronous circuits, but this method is not robust since gate orphans could result. Also, the physical implementation of the circuit may not be feasible when the circuit has to be modified to achieve robustness. This is owing to the requirement of unconventional complex gates, which is clarified through an elaborate discussion in [14]. The full adder functionality can also be realized based on the Null Convention Logic (NCL) design paradigm [23], but this would require usage of the proprietary cell macros developed on the basis of threshold logic [24] [25]. In this context, it should be noted that the NCL utilizes the DRCL style as the underlying principle for realizing self-timed versions of combinatorial logic, which is discussed in the next section.

3 Robust Early Output Full Adder

The DRCL style utilizes De-Morgan's theorems of Boolean algebra to implement a combinational logic function in an asynchronous style by replacing each gate by its dual-rail equivalent (dual-rail pair). We consider two scenarios for the dual-rail combinational equivalent of a Boolean function, say F = *ab* + *cd* as shown in figure 12, to clarify the necessity for ensuring proper indication of signal events at the primary inputs as well as the intermediate output nodes and to describe how *wire* and *gate orphans* could possibly result [14].

Assuming all the data inputs to be currently spacers, when *a*0 and *c*0 become defined, intermediate signals *x*0 and *y*0 would become defined and eventually F0 would become defined. Assuming *b*0 and *d*0 would also become defined subsequently, they would not be acknowledged by the intermediate signals (*x*0 and *y*0) or by the corresponding output in the present evaluation phase resulting in 'wire orphans'.

Let us assume that *a*1 and *b*1 become defined after a return-to-zero phase. This would lead to defining of the intermediate signal *x*1. Assuming that *c*1 and *d*1 become subsequently defined during the current evaluation phase, F1 could have become defined as a result of *x*1 alone becoming defined, and hence a late transition on *y*1 would not be acknowledged by the primary output, giving rise to a 'gate orphan'.

From the above discussion, it should be clear that the DRCL realization is basically non-indicating and, as such it conforms to *eager evaluation* owing to the fact that even with a subset of the function block inputs becoming defined (undefined), all the outputs could become defined (undefined) regardless of the lately arriving inputs. Hence the DRCL style is not strongly or weakly indicating but corresponds to *early output logic*, i.e. early set and/or reset could happen.

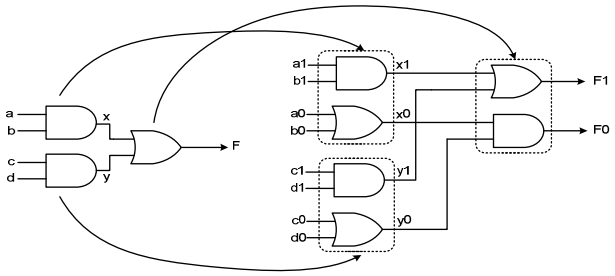

Fig. 12. Dual-rail combinational equivalent of the logic function, F = *ab* + *cd*

The proposed robust early output logic based full adder is portrayed by figure 13. (*a*0, *a*1) and (*b*0, *b*1) are the augend and addend inputs, with (*cin*0, *cin*1) representing the input carry. The sum and carry outputs of the adder are specified by the signals (*Sum*0, *Sum*1) and (*Cout*0, *Cout*1) respectively. The physical implementation of the adder requires eight complex gates (including four AO22 gates), with the 2-input Muller C-element realized using an AO222 gate with feedback, and two OR2 gates.

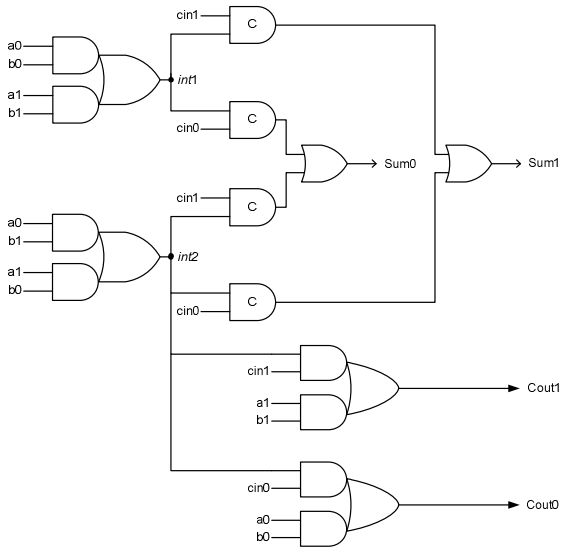

Fig. 13. Robust asynchronous full adder realization corresponding to early output logic

The synthesized full adder design, shown above, pertains to early propagation, but is robust as the primary outputs duly acknowledge the steady state of all the internal nodes thereby preventing the occurrence of gate orphans. This is feasible, with intermediate gate output nodes '*int*1' and '*int*2' of the above figure specified as isochronic fork junctions. For example, when valid data is applied, a low-to-high transition on *int*1 or *int*2 nodes would be followed by a low-to-high transition on *Sum*0/*Sum*1 output. Especially for the up-going transition on *int*2, multiple acknowledgements would result since an up-going transition would also eventually occur on *Sum*0/*Sum*1 and *Cout*0/*Cout*1 outputs. For a down-going transition on nodes *int*1/*int*2, such an event is duly acknowledged by the sum outputs with the dual-rail carry input also experiencing a high-to-low transition. Nevertheless, the carry outputs may not acknowledge the down-going transition on node *int*2 since input-incomplete gates have been used for realizing the output carry logic. But this does not affect the robustness of the adder realization since Martin's QDI timing assumption is satisfied, which was quoted in section 2.1. Also, the primary inputs are routed to different gates with the help of isochronic forks that are widely prevalent in QDI circuits [27].

When valid data is applied the adder circuit shown in figure 13, after a return-to-zero state, the sum and carry signals would demand the arrival of the necessary primary inputs to produce the desired output. However for the spacer data, with a reset of either the augend or the augend inputs and the input carry, the sum outputs could be reset. With respect to the output carry, the dual-rail carry signal could be reset even with the reset of a single dual-rail input regardless of whether the carry-kill, carry-generate or carry-propagate condition occurs. Hence, an early reset of the adder is possible since the return-to-zero condition does not require the reset of all the primary inputs thus making it early propagative (early reset). Also, an *n*-bit adder can return to the spacer state with just two full adder delays similar to that of Martin's adder [26]. When the augend and/or addend inputs of a k^{th} adder stage is reset, the corresponding dual-rail carry output is also reset, which serves as the input carry for the $(k+1)^{th}$ adder stage. With the augend/addend inputs of the $(k+1)^{th}$ adder stage also being simultaneously reset, the dual-rail sum outputs of that stage could be reset. Thus it becomes clear that with an approximate delay of only two full adder stages, an entire *n*-bit carry-ripple adder can be reset, which is beneficial for the speed of the adder.

With reference to the preceding discussions, the important properties of the proposed full adder are summarized as follows:

- Does not process valid data eagerly but could be reset in an early fashion, relaxing the weak-indication constraints, but without any compromise on robustness
- Exhibits actual case latency when adding valid data and constant latency during return-to-zero, since the intermediate carry outputs can be reset on the basis of the data inputs associated with each adder stage

Though the proposed early output adder guarantees gate orphan freedom, wire orphans could still manifest. To clarify this phenomenon, we consider an instance. With *a*1=*b*1=*cin*1=1 in figure 13, outputs *Sum*1 and *Cout*1 experience a transition. In the return-to-zero phase, assuming that *a*1 and *cin*1 have experienced a low-going transition before *b*1, *Sum*1 and *Cout*1 could be reset even though *b*1 has not become undefined (spacer), giving rise to a wire orphan. Nevertheless, the problem of wire orphans gets resolved when the adder is embedded within the system configuration as shown in figure 14.

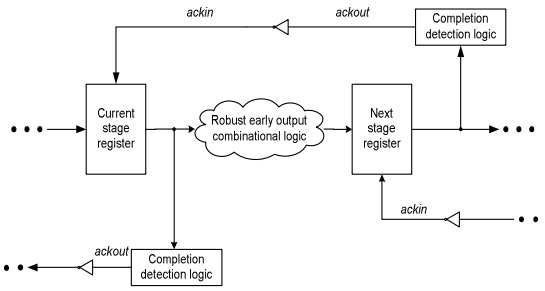

Fig. 14. Architecture of the self-timed system

Here, the robust early output combinational logic block can be identified as the full adder that is sandwiched between two register bank stages. When the adder is incorporated into the system topology, the reset of *b*1, although unacknowledged by the asynchronous logic block is however acknowledged by the completion detection (CD) logic associated with the current stage register. The CD circuit in this case consists of a bank of OR2 gates with an OR2 gate dedicated for each dual-rail input. The outputs of all the OR gates are synchronized by means of a C-element tree producing the *ackout* signal [15], which may be buffered to act as the input for the register bank of the preceding stage. Given the isochronicity assumption on all the primary inputs, wire orphans do not crop up. Thus in contrast to an indicating logic block, the responsibility of indicating the primary inputs is shared between the robust early output logic block and the CD circuit, thus preserving the QDI property of the system.

4 Results

A number of 32-bit self-timed ripple carry adders (RCAs), which incorporate different types of full adders were constructed and optimized for minimum delay and their design metrics were evaluated. These are given in Table 1. Table 2 shows the critical path elements encountered in the different self-timed RCAs.

The simulation results were obtained by targeting a high-speed 130nm UMC (Faraday) bulk CMOS process technology. The delay metric signifies the latency of the critical path. The area parameter refers to the total area of all the cells comprising the combinatorial adder logic, the register bank and the CD circuitry associated with it. The average power dissipation is estimated for 1000 random input vectors, supplied at intervals of 25ns to the self-timed system from the external environment (test bench). All the adder inputs were configured with the driving strength of the minimum sized inverter in the library while the outputs possess fanout-of-4 drive strength.

Table 1. Design metrics of various 32-bit self-timed carry-ripple adders

Full adder realization style	Delay (ns)	Area (μm^2)	Power (μW)
Seitz_adder (Strong) [7]	12.8	8329	507.9
Seitz_adder (Weak) [7]	6.5	7689	459.5
DIMS_adder (Strong) [9]	13.8	10089	427.4
DIMS_adder (Weak) [9]	12.8	10665	476.5
Toms et al._adder (Strong) [10]	10.6	7561	388.4
Folco et al._adder (Weak) [11]	8.0	6633	383.2
SSSC_adder (Weak) [12]	5.8	7081	420.4
Toms et al._adder (Weak) [13]	11.6	8713	545.3
Proposed_adder (Robust Early)	4.6	5609	350.1

Table 2. Critical path element(s) pertaining to the various self-timed RCAs

Full adder realization style	Elements in the critical path
Seitz_adder (Strong)	OR2 + OR2 + CE2
Seitz_adder (Weak)	AND3 + OR3
DIMS_adder (Strong)	CE3 + OR2 + OR2
DIMS_adder (Weak)	CE3 + OR3
Toms et al._adder (Strong)	CE2 + OR2 + OR2
Folco et al._adder (Weak)	CE2 + OR2
SSSC_adder (Weak)	AO222
Toms et al._adder (Weak)	OR2 + CE2 + OR3
Proposed_adder (Robust Early)	AO22

5 Conclusions

It is evident from the simulation results that the proposed robust full adder corresponding to early output logic leads to optimization of design metrics, outweighing its nearest competitors in terms of delay, area and power parameters by 20.7%, 15.4% and 8.6% respectively – thanks to the reduced latency in producing the intermediate carries and the compact circuit realization. Thus, this work highlights the merit and potential of robust asynchronous arithmetic circuit synthesis based on the notion of early propagation.

References:

[1] SIA's ITRS 2008 Design Report, Available: http://www.itrs.net

[2] A.J. Martin, S.M. Burns, T.K. Lee, D. Borkovic and P.J. Hazewindus, "The first asynchronous microprocessor: the test results," *ACM SIGARCH Computer Architecture News*, vol. 17, no. 4, June 1989, pp. 95-98.

[3] G.F. Bouesse, G. Sicard, A. Baixas and M. Renaudin, "Quasi delay insensitive asynchronous circuits for low EMI," *Proc. 4th International Workshop on Electro-Magnetic Compatibility of Integrated Circuits*, 2004, pp. 27-31.

[4] C.H. van Berkel, M.B. Josephs and S.M. Nowick, "Scanning the technology: applications of asynchronous circuits," *Proc. of the IEEE*, vol. 87, no. 2, February 1999, pp. 223-233.

[5] A.J. Martin, "The limitation to delay-insensitivity in asynchronous circuits," *Proc. 6th Conference on Advanced Research on VLSI*, MIT Press, 1990, pp. 263-278.

[6] A.J. Martin and P. Prakash, "Asynchronous nano-electronics: preliminary investigation," *Proc. 14th IEEE International Symposium on Asynchronous Circuits*, 2008, pp. 58-68.

[7] C.L. Seitz, "System Timing" in *Introduction to VLSI Systems*, C. Mead and L. Conway (Eds.), Addison-Wesley, MA, USA, 1980, pp. 218-262.

[8] I. David, R. Ginosar and M. Yoeli, "An efficient implementation of Boolean functions as self-timed circuits," *IEEE Trans. on Computers*, vol. 41, no. 1, January 1992, pp. 2-11.

[9] J. Sparso and J. Staunstrup, "Delay-insensitive multi-ring structures," *Integration, the VLSI Journal*, vol. 15, no. 3, pp. 313-340, Oct. 1993.

[10] W.B. Toms and D.A. Edwards, "Efficient synthesis of speed independent combinational logic circuits," *Proc. 10th Asia and South-Pacific Design Automation Conference*, 2005, pp. 1022-1026.

[11] B. Folco, V. Bregier, L. Fesquet and M. Renaudin, "Technology mapping for area optimized quasi delay insensitive circuits," *Proc. IFIP International Conference on VLSI/SoC*, 2005, pp. 146-151.

[12] P. Balasubramanian and D.A. Edwards, "A delay efficient robust self-timed full adder," *Proc. 3rd IEEE International Design and Test Workshop*, 2008, pp. 129-134.

[13] W.B. Toms and D.A. Edwards, "A complete synthesis method for block-level relaxation in self-timed datapaths," *Proc. 10th International Conference on Application of Concurrency to System Design*, 2010, pp. 24-34.

[14] P. Balasubramanian, K. Prasad and N.E. Mastorakis, "Robust asynchronous implementation of Boolean functions on the basis of duality," *Proc. 14th WSEAS International Conference on Circuits*, 2010, pp. 37-43.

[15] J. Sparso and S.B. Furber (Eds.), *Principles of Asynchronous Circuit Design – A Systems Perspective*, Kluwer Academic Publishers, Boston, 2001.

[16] A. Kondratyev and K. Lwin, "Design of asynchronous circuits by synchronous CAD tools," *IEEE Design & Test of Computers*, vol. 19, no. 4, July-August 2002, pp. 107-117.

[17] J. Cortadella, A. Kondratyev, L. Lavagno and C. Sotiriou, "Coping with the variability of combinational logic delays," *Proc. IEEE International Conference on Computer Design*, 2004, pp. 505-508.

[18] Y. Zhou, D. Sokolov and A. Yakovlev, "Cost-aware synthesis of asynchronous circuits based on partial acknowledgement," *Proc. IEEE/ACM International Conference on Computer Aided Design*, 2006, pp. 158-163.

[19] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. on Computers*, vol. C-35, no. 8, August 1986, pp. 677-691.

[20] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard and M. Renaudin, "Static implementation of QDI asynchronous primitives," *Proc. International Workshop on Power and Timing Modeling, Optimization and Simulation*, J.J. Chico and E. Macii (Eds.), *Lecture Notes in Computer Science*, vol. 2799, 2003, pp. 181-191.

[21] M. Zhao and S.S. Sapatnekar, "A new structural pattern matching algorithm for technology mapping," *Proc. 38th Annual ACM/IEEE Design Automation Conference*, 2001, pp. 371-376.

[22] C. Jeong and S.M. Nowick, "Block-level relaxation for timing-robust asynchronous circuits based on eager evaluation," *Proc. 14th IEEE International Symposium on Asynchronous Circuits and Systems*, 2008, pp. 95-104.

[23] K.M. Fant and S.A. Brandt, "NULL convention logic: a complete and consistent logic for asynchronous digital circuit synthesis," *Proc. IEEE International Conference on Application Specific Systems, Architectures and Processors*, 1996, pp. 261-273.
[24] K.M. Fant and S.A. Brandt, "Null convention logic system," *US Patent* 5828228, October 1998.
[25] K.M. Fant and G.E. Sobelman, "Null convention threshold gate," *US Patent* 5664211, February 1997.
[26] A.J. Martin, "Asynchronous datapaths and the design of an asynchronous adder," *Formal Methods in Systems Design*, vol. 1, no. 1, July 1992, pp. 117-137.
[27] W.B. Toms, "Synthesis of quasi-delay-insensitive datapath circuits," *PhD Thesis*, University of Manchester, 2006.