A Delay Efficient Robust Self-Timed Full Adder

P. Balasubramanian and D.A. Edwards
School of Computer Science
The University of Manchester
Oxford Road, Manchester M13 9PL, United Kingdom.
E-mail ID: {padmanab, doug}@cs.man.ac.uk

***Abstract*—Addition forms the basis of digital computer systems. A gate level self-timed full adder design, utilizing a pre-defined set of gates, available in a commercial synchronous standard cell library is discussed in this paper. The proposed adder satisfies Seitz's weak-indication specifications and exhibits reduced data path delay in comparison with other existing adders, which satisfy the property of indication. In terms of power and area, it is competitive to the best of other self-timed adders.**

I. Introduction

Self-timed (ST) circuit design, in general, guarantees that the requisite functionality is satisfied irrespective of delays in the circuit components or communicating wires. In addition, the potential advantages of low power consumption, less emission of electro-magnetic noise, greater modularity, no clock-distribution and clock-skew problems and tolerance to variations in supply voltage, temperature and fabrication process parameters are inherent in ST designs. The latest edition of the Semiconductor Industry Association's ITRS report [1] predicts that parametric variation of device delay (as a percentage effect on sign-off delay) is expected to increase from a current figure of 8% to 25% by 2020. Design blocks reuse (otherwise referred to as design modularity), as a percentage of all logic is anticipated to increase from a current figure of 36% to 55% by 2020. Interest in ST design is gaining ground mainly due to increase in parameter uncertainties, projected for future ultra deep submicron technologies. In this scenario, ST design attracts attention, as it encompasses the ability to deal with device irregularities.

Though ST circuit designs are appealing, the lack of commercially available sophisticated EDA tools (for design, testing and validation) necessitate the need for developing novel full-custom solutions in many cases. Especially, in the context of this paper, full custom self-timed adder design becomes necessary. This work takes a different approach by looking for utilization of readily available off-the-shelf synchronous resources (standard cells) to realize asynchronous logic designs. Specifically, this paper incorporates two main original contributions. Firstly, C-elements which form the backbone of robust asynchronous architectures are implemented using standard library cells with provision of good granularity. Secondly, is the design of a new gate level self-timed full adder using C-elements, complex and ordinary logic gates. Hence, in view of these, it becomes clear that this design is sort of semi-custom rather than being full-custom.

II. Function Block and Realization of C-elements

Before proceeding further, the terminology "function block" is first explained. A function block is the asynchronous equivalent of a synchronous combinational logic circuit [2]. But apart from satisfying the specified functionality, it must also be transparent to handshaking as implemented by its surrounding latches. Henceforth, we shall restrict ourselves to a function block characterized by the 4-phase dual-rail (DR) protocol, a robust design approach having its roots in the pioneering work of Muller in the early 60's [3]. The 4-phase signaling protocol is also known as the return-to-zero protocol, wherein input data alternates between valid data and empty data (also called spacer). The DR input encoding protocol is a delay-insensitive (DI) protocol and is the simplest and widely used member of the family of DI codes [4]. According to this protocol, a signal x is encoded into two bits as $x1$ (for true) and $x0$ (for false). A spacer refers to an input combination where both $x1$ and $x0$ are 0 and valid data of 0 and 1 are represented by input combinations ($x1 = 0$, $x0 = 1$) and ($x1 = 1$, $x0 = 0$) respectively. The state of $x1$ and $x0$ being simultaneously high is invalid and illegal in a DR protocol.

A function block can be classified as either strongly indicating or weakly indicating depending on how it behaves with respect to the handshaking transparency. A strong-indication function block waits for all of its inputs to become valid/empty before it produces valid/empty outputs, while a weak-indication function block may start to produce valid/empty outputs as soon as a subset of the inputs have become valid/empty. However, unless all its inputs have not become valid/empty, all its outputs would not become valid/empty. The above constraints ensure that the generation of all the primary outputs in a function block wholly indicates the arrival of all the input data and also the completion of computation within it. From the preceding discussion, it can be understood that an asynchronous logic circuit (here, self-timed adder) can be likened to a function block. Since Muller C-elements are fundamental to robust self-timed implementations, the standard gate-orphan (unacknowledged transition on a gate output) free realization of a 2-input Muller C-element (CE2) is first given, as portrayed by figure 1.

This research is sponsored by EPSRC, UK through the SEDATE project grant EP/D052238/1.

Node *n* in this figure represents an isochronic fork junction (which is a weak but indispensable practical compromise to delay-insensitivity) [5]. A logic level change on node *n* is therefore visible simultaneously both to the downstream logic which it feeds, as well as on the inputs of the complex gate (to which it is fed back). As per the property of acknowledgement defined by Martin in [6], when a rising or falling transition on an output is acknowledged and hence completed, then a similar transition on all outputs are also considered to be acknowledged and thereby completed. Complying with this property, 3-input and 4-input Muller C-elements functionality were also implemented using combinations of appropriate standard cells. The realizations satisfy the required functionality and also preserve indication constraints, wherein transitions get properly acknowledged.

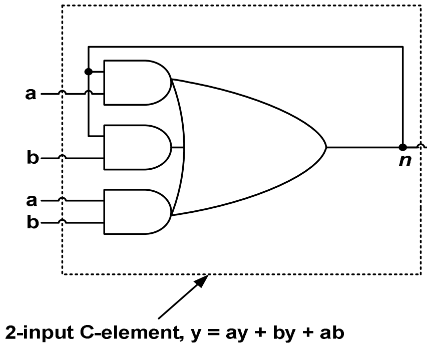

Figure 1. An AO222 standard cell configured as a CE2

III. Prior Related Work

The general equations governing the true and false sum and carry outputs of a full adder are given by the following:

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

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

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

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

The different self-timed adder designs that follow, which satisfy the above functionality, are suitable for realization with standard library cells. They primarily differ on the basis of their indication property and also based on their timing model, whether quasi-delay-insensitive (QDI) or speed-independent (SI) [2]. Different self-timed full adder designs are possible based on the methods of [7] [8] [9] [10] [11] and [12]. It is not the intention of this article to detail the relative merits, de-merits and limitations of the above and so we restrict our attention to the full adder functionality alone, but provide some useful insights. The C-element is represented by an AND gate labeled with letter C, in the diagrams that follow.

The adders based on the above approaches satisfy the monotonic cover constraint (MCC) [13], except the one based on [9]. In this case, all the sum terms in the subnet DRN (though monotonic) of the combinatorial network need to become active in order for the false outputs to evaluate to the correct state in case of appropriate valid input data. In case of empty data, even with a single sum term becoming disabled and with ORN and CEN being reset, the outputs can evaluate to the correct empty state. This is a problematic situation, as the function block outputs then do not properly indicate the transitions on all its intermediate nodes. Alternatively, the internal nodes need not have stabilized as well. Thus the design cannot be classified as either QDI or SI, though it is self-timed, necessitating the consideration of timing assumptions to ensure proper self-timed operation.

The full adder design based on [7] is shown in figure 2. It is basically a strong-indication adder, which implies that both sum and carry outputs depend on the arrival of all the inputs and so carry propagation to the next stage could not be fast. Multiple acknowledgements are possible on some wire forks.

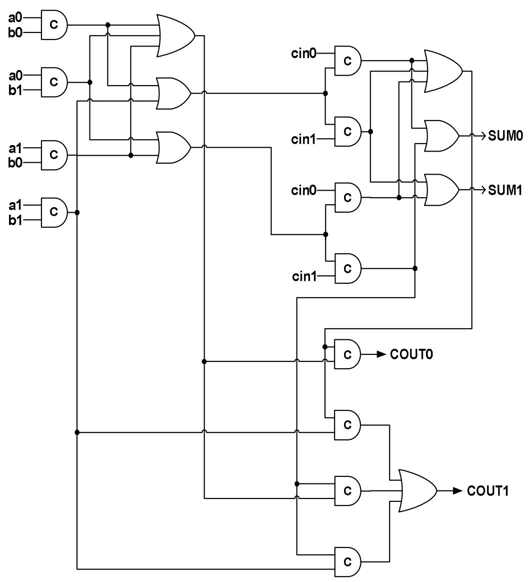

Figure 2. Singh's strong-indication full adder

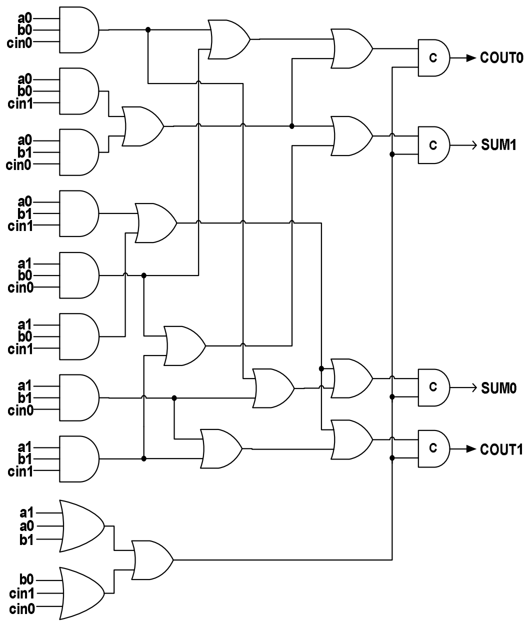

Figure 3. Seitz's strong-indication full adder

The strongly and weakly indicating SI full adders based on Seitz's approach [8] are shown in figures 3 and 4. The latter incorporates a fast carry propagation path as only the sum outputs depend on all the inputs, while the carries do not. The strong-indication version deteriorates the data path delay. The weak-indication full adder is slightly modified to facilitate faster carry logic for all but the least significant stage.

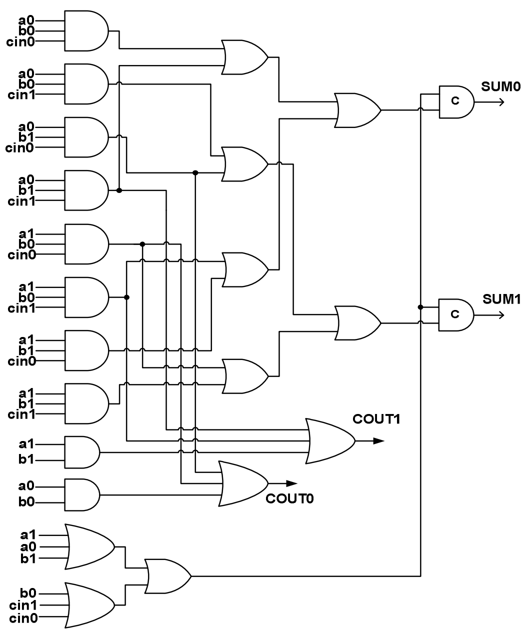

Figure 4. Seitz's weak-indication full adder

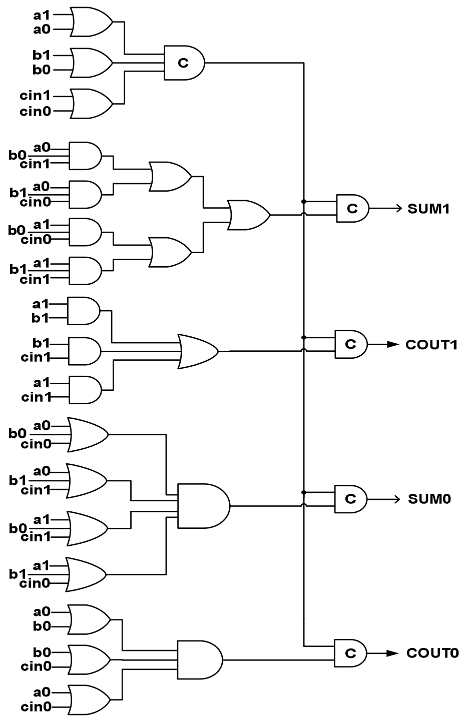

Figure 5. David et al.'s self-timed full adder

The full adder design based on the approach of [9] is shown in figure 5. Timing assumptions are implicit in this design. As was mentioned earlier, since the entire sum terms of the monotonic [14] DRN sub-network need to be enabled to produce a correct false function output during valid data phase; switching activity would be high for this adder thereby increasing the power dissipation. This is substantiated by the power figures listed in Table II.

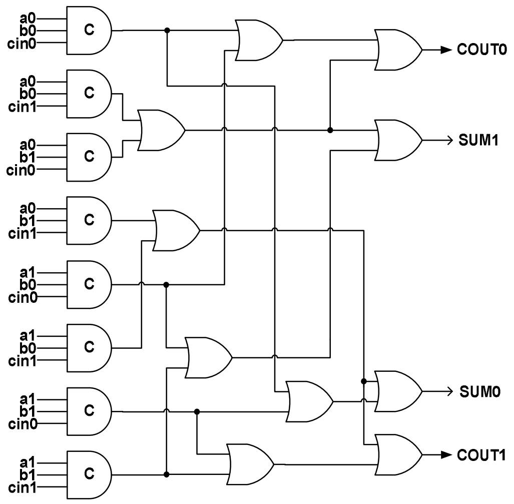

Figure 6. DIMS strong-indication full adder

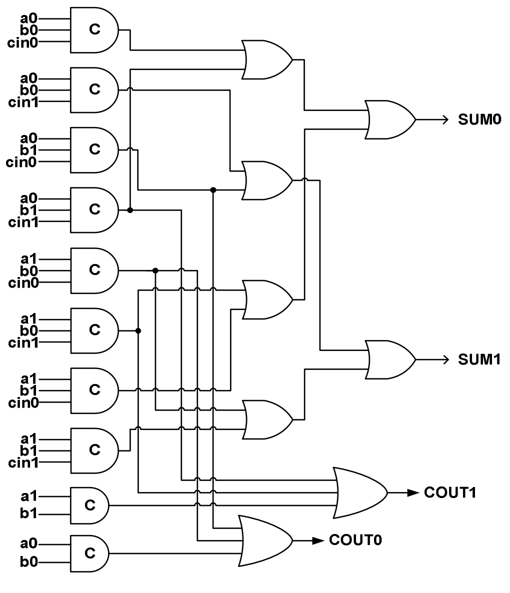

Figure 7. DIMS weak-indication full adder

The strong and weak-indication full adder realizations based on the delay-insensitive minterm synthesis (DIMS) approach of [10] are shown in figures 6 and 7. In theory, the latter is comparable to Seitz's weak-indication full adder, entrusting the responsibility of indicating transitions on all the adder inputs on its sum outputs, with the circuit indication distributed between sum and carry outputs. However, it is slower than Seitz's design as it employs C-elements instead of

AND gates for synthesizing the outputs, but it conforms to a more robust QDI timing model.

A weakly indicating full adder design based on the robust QDI model was conceived in [11], shown in figure 8. As in the case of the weak-indication DIMS full adder, the indication is distributed between the sums and carry outputs. However, carry propagation logic is relatively faster owing to minimum delay encountered in the data path.

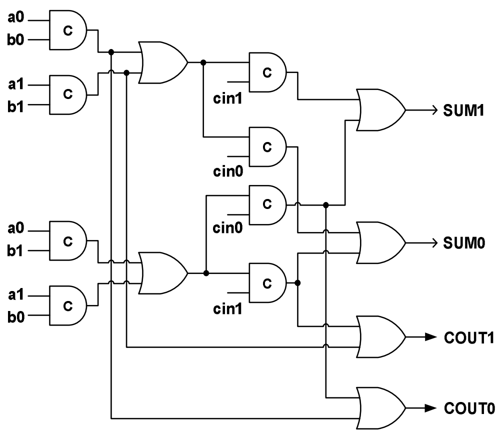

Figure 8. Folco et al.'s weak-indication full adder

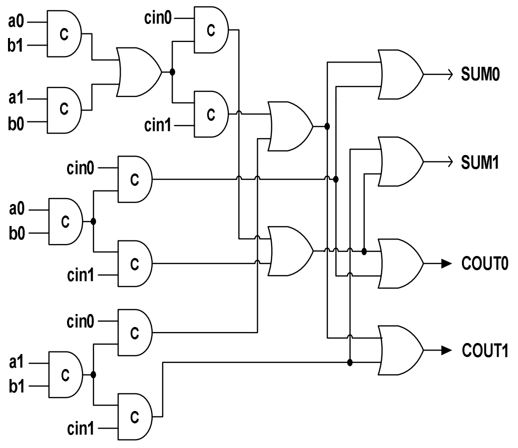

Figure 9. Toms' strong-indication full adder

The full adder based on the strongly indicating combinational logic synthesis procedure of [12] is depicted by figure 9. It is comparable with the DIMS strong-indication adder in that it is also QDI, but the indication is not distributed and so both sum and carry outputs indicate the validity of all the adder inputs. But the data path delay is lesser than the DIMS adder shown in figure 5, thanks to speed-independent logic decomposition.

The adders have all been realized using the standard cells of the 130nm Faraday CMOS process, which features an AND gate with a maximum fan-in of 4 and an OR gate with a maximum fan-in of 3. Hence, 4-input disjunctive logical operators have been split in a delay-optimized fashion, which is noticeable in the circuits depicted by figures 3, 4, 5, 6 and 7.

IV. PROPOSED SELF-TIMED ADDER

The proposed robust self-timed full adder is portrayed by figure 10. Apart from C-elements and OR gates, it features complex gates (AO222) commonly found in a cell library. This adder is quite unique in that while the sum outputs satisfy the MCC, the carry outputs algebraically do not. Since they are described by a complex gate, this does not pose a problem. So there is no possibility for gate-orphans. The adder is weakly indicating with inputs indication dedicated to the sum outputs, thereby the carry logic is optimized for fast propagation to the subsequent stages. The carry outputs become zero during the return-to-zero phase, with isochronic fork assumption applied to the inputs. The ambiguity in determining the arrival of augend and addend bits through the carry outputs is resolved completely by the sum outputs. Also, there are no multiple acknowledgements on any wire fork. This design is found to be the fastest, evident from Table I.

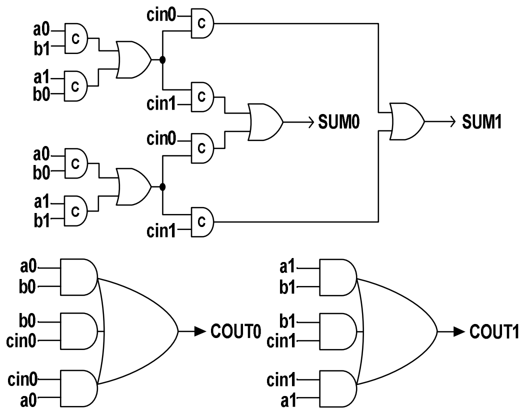

Figure 10. Proposed full adder

V. SIMULATION MECHANISM, RESULTS AND DISCUSSION

All the simulations were performed targeting a high-speed 130nm Faraday bulk CMOS process technology (typical case) at an ambient temperature of 25°C, using Synopsys and Cadence tools on a Linux platform. A supply voltage of 1.2V was used. A DI asynchronous version of an *n*-bit ripple carry adder (RCA) topology, shown in figure 11, was used to analyze the characteristics of different ST full adder blocks.

Table I highlights delay and cell area metrics for the ST adders of various sizes. The power dissipation components are listed in Table II. All the adder outputs exhibit approximately fanout-4 drive strength. The data path delay is indicative of the propagation delay, encountered while traversing the longest path from a primary input of the current stage function block to the inverted acknowledge signal generated out of it, after crossing the output latches. This signal is in turn meant to be fed back as the acknowledge input for the previous stage. The forward latency specifies the worst case combinational delay from a primary input to a primary output of the function block. Cell area is a measure of the area occupancy of the elements present in the data path logic, including the output registers and completion detection circuitry. The input acknowledge signal is embedded into the data path logic, where possible, to eliminate the need for extra cells. The primary inputs associated with the current stage module and the acknowledge input for its output registers are assumed to come from the environment. Similar completion detection circuitry was used for adders of the same size, to maintain uniformity.

TABLE I. DELAY AND AREA METRICS FOR DIFFERENT ST RCAS

ST RCA Size	Adder Design Style	Data Path Delay (ns)	Forward Latency (ns)	Cells Area (μm^2)
4 bits	Singh_Strong [7]	4.24	3.70	935
	Seitz_Strong [8]	3.14	2.63	895
	Seitz_Weak [8]	2.20	1.70	799
	David et al._ST [9]	4.54	4.03	1029
	DIMS_Strong [10]	3.39	2.90	1131
	DIMS_Weak [10]	3.30	2.80	1203
	Folco et al._Weak [11]	2.68	2.17	699
	Toms_Strong [12]	3.02	2.52	815
	Proposed_Weak	2.06	1.53	755
8 bits	Singh_Strong [7]	7.28	6.78	1836
	Seitz_Strong [8]	5.59	5.08	1764
	Seitz_Weak [8]	3.51	2.97	1564
	David et al._ST [9]	8.45	7.94	2026
	DIMS_Strong [10]	6.08	5.58	2228
	DIMS_Weak [10]	5.80	5.30	2372
	Folco et al._Weak [11]	4.36	3.85	1364
	Toms_Strong [12]	5.10	4.59	1596
	Proposed_Weak	3.28	2.71	1476
16 bits	Singh_Strong [7]	13.44	12.94	3635
	Seitz_Strong [8]	10.52	9.99	3499
	Seitz_Weak [8]	6.32	5.52	3091
	David et al._ST [9]	16.26	15.75	4017
	DIMS_Strong [10]	11.46	10.96	4419
	DIMS_Weak [10]	10.80	10.31	4707
	Folco et al._Weak [11]	7.91	7.21	2691
	Toms_Strong [12]	9.35	8.75	3155
	Proposed_Weak	5.90	5.06	2915
32 bits	Singh_Strong [7]	25.91	25.25	7236
	Seitz_Strong [8]	20.46	19.79	6972
	Seitz_Weak [8]	11.28	10.63	6148
	David et al._ST [9]	32.04	31.37	8002
	DIMS_Strong [10]	22.37	21.72	8804
	DIMS_Weak [10]	20.96	20.31	9380
	Folco et al._Weak [11]	14.58	13.92	5348
	Toms_Strong [12]	17.72	17.06	6276
	Proposed_Weak	10.43	9.77	5796

For a 32-bit ST RCA version, the proposed adder is less expensive than Seitz's adder in terms of data path delay and forward latency by 7.5% and 8.1% respectively, while it reports 1.8% reduction in reverse latency. However, it occupies 8.4% more area than Folco et al.'s adder.

The power results depicted by Table II have all been obtained by applying the input trace of a simple combinatorial benchmark (dc1) to all the adders, at durations of 35ns so as to accommodate the slowest adder [9]. The proposed full adder is more power consuming than DIMS, Toms' strong-indication adders and Folco et al.'s weak-indication adder. In terms of the leakage power component, it is comparable with Folco et al.'s adder block. In terms of total power, dynamic power (summation of switching and internal power components) and leakage power, the 32-bit ST RCA based on the proposed full adder module is more expensive than Folco et al.'s adder by 5.1%, 5.2% and 0.4% respectively. The low leakage power component of the proposed adder is likely to emphasize its power efficiency in case of lower geometries. Nevertheless, the proposed ST adder would attain the top slot amongst all other adders with respect to power-delay and energy-delay product parameters.

TABLE II. POWER DISSIPATION COMPONENTS OF DIFFERENT ST RCAS

ST RCA Size	Adder Design Style	Total Power (μW)	Dynamic Power (μW)	Leakage Power (nW)
4 bits	Singh_Strong [7]	25.11	24.85	265.76
	Seitz_Strong [8]	24.58	24.30	277.24
	Seitz_Weak [8]	19.99	19.76	239.64
	David et al._ST [9]	33.32	33.01	311.81
	DIMS_Strong [10]	18.00	17.73	276.82
	DIMS_Weak [10]	19.35	19.07	286.33
	Folco et al._Weak [11]	18.32	18.11	206.80
	Toms_Strong [12]	18.71	18.48	233.50
	Proposed_Weak	19.09	18.88	210.42
8 bits	Singh_Strong [7]	49.13	48.62	512.13
	Seitz_Strong [8]	49.06	48.51	547.63
	Seitz_Weak [8]	39.04	38.57	465.97
	David et al._ST [9]	65.92	65.31	612.38
	DIMS_Strong [10]	35.61	35.08	537.31
	DIMS_Weak [10]	37.98	37.43	553.65
	Folco et al._Weak [11]	35.62	35.23	396.04
	Toms_Strong [12]	36.28	35.83	448.18
	Proposed_Weak	37.33	36.93	400.21
16 bits	Singh_Strong [7]	96.62	95.62	1002.09
	Seitz_Strong [8]	97.58	96.49	1086.11
	Seitz_Weak [8]	76.56	75.65	916.58
	David et al._ST [9]	130.52	129.31	1210.57
	DIMS_Strong [10]	70.47	69.42	1055.11
	DIMS_Weak [10]	74.70	73.62	1085.28
	Folco et al._Weak [11]	69.73	68.96	771.26
	Toms_Strong [12]	70.84	69.96	874.48
	Proposed_Weak	73.26	72.48	776.32
32 bits	Singh_Strong [7]	192.83	190.85	1984.48
	Seitz_Strong [8]	195.67	193.51	2168.41
	Seitz_Weak [8]	152.71	150.89	1819.97
	David et al._ST [9]	260.84	258.43	2408.59
	DIMS_Strong [10]	141.44	139.34	2092.29
	DIMS_Weak [10]	149.37	147.22	2150.76
	Folco et al._Weak [11]	139.16	137.63	1523.88
	Toms_Strong [12]	141.19	139.46	1729.51
	Proposed_Weak	146.32	144.79	1530.68

VI. CONCLUSION AND FURTHER WORK

A new gate level ST full adder, based on synchronous standard cells is presented in this paper. Muller C-elements (especially 3 and 4-input elements), which form the backbone of robust self-timed architectures, have also been realized using suitable standard cells of the library, providing good granularity. Performance of the proposed adder vis-à-vis other existing adders has been analyzed in terms of power, delay and area metrics on a DI RCA topology of various sizes. Higher order adders are constructed by cascading copies of the individual ST full adder modules.

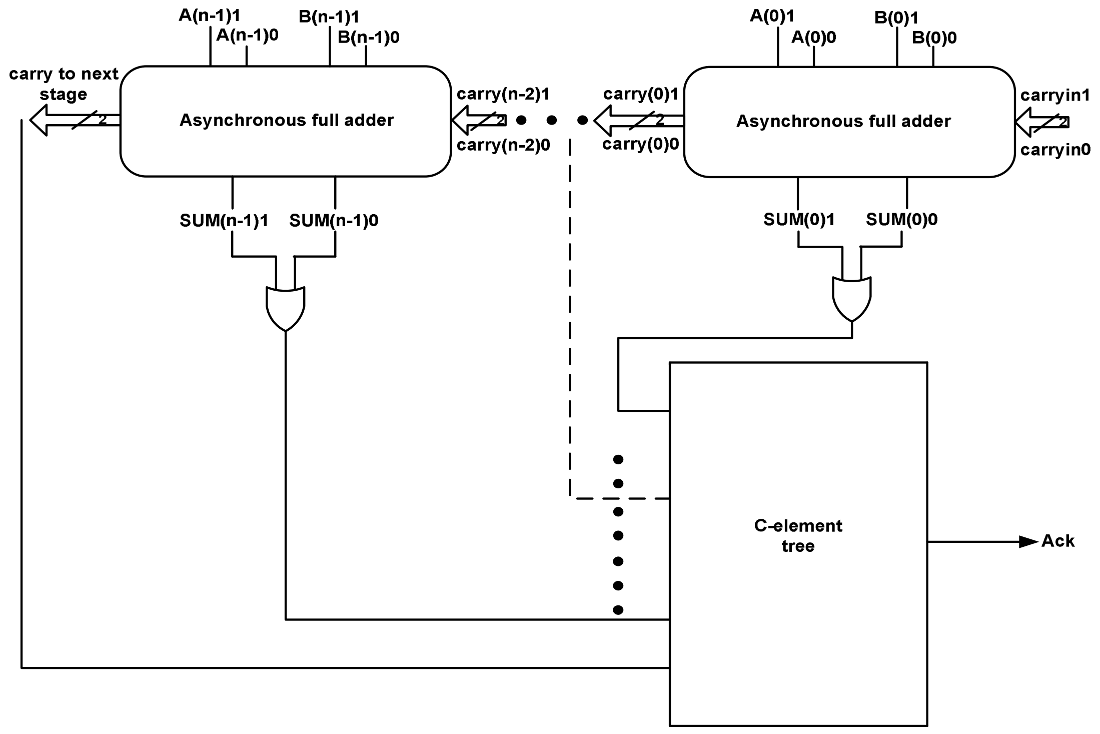

Figure 11. A delay-insensitive *n*-bit ripple carry adder topology

This is possible as a valid combinatorial circuit cascade of strongly/weakly indicating function blocks is itself a strong/weak-indication function block [2] [8]. The proposed full adder is based on a robust ST design approach and is found to exhibit less data path delay in comparison with others, while only inferior to DIMS (strong indication), Folco et al.'s and Toms' adders in terms of total power (mainly dynamic power). In terms of area occupancy, it is inferior to only Folco et al.'s adder. Further work is planned to implement some high-speed synchronous adder topologies in asynchronous style. This involves careful and effective realization of the supporting data path logic and is quite hard, as it should guarantee that transitions on all inputs are properly acknowledged. Also, a new asynchronous adder topology is currently being developed and the preliminary results are encouraging. It is anticipated that the proposed adder would benefit in achieving optimality of design metrics in case of other asynchronous adder realizations as well.

REFERENCES

[1] SIA's ITRS report 2007 edition, Available: http://www.itrs.net

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

[3] D.E. Muller, "Asynchronous logics and application to information processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.

[4] T. Verhoeff, "Delay-insensitive codes: an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1-8, 1988.

[5] A.J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," *Proc. 6th MIT Conference on Advanced Research in VLSI*, pp. 263-278, MIT Press, 1980.

[6] A.J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," *Distributed Computing*, vol. 1, no. 4, pp. 226-234, 1986.

[7] N.P. Singh, "A design methodology for self-timed systems," *MIT Computer Science Laboratory Tech. Report* TR-258, Feb. 1981.

[8] C.L. Seitz, "Chapter 7 – System Timing", in *Introduction to VLSI Systems*, C.A. Mead and L.A. Conway (Eds.), Addison-Wesley, 1980.

[9] 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, pp. 2-11, Jan. 1992.

[10] J. Sparso and J. Staunstrup, "Delay-insensitive multi-ring structures," *Integration, the VLSI journal*, vol. 15, no. 1, pp. 313-340, Oct. 1993.

[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*, pp. 55-69, 2005.

[12] W.B. Toms, "Synthesis of Quasi-Delay-Insensitive Datapath Circuits," *PhD thesis*, University of Manchester, 2006.

[13] A. Kondratyev, M. Kishinevsky and A. Yakovlev, "Hazard-free implementation of speed-independent circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17, no. 9, pp. 749-771, 1998.

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