Self-Timed Full Adder Designs based on Hybrid Input Encoding

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

Abstract—**Self-timed full adder designs based on commercial synchronous resources (standard cells), constructed using a mix of complete delay-insensitive codes adopted for inputs are described in this paper. While one of the adder designs incorporates redundancy into the logic, the other design does not. Comparisons have been carried out with respect to various self-timed full adder designs which employ only a single widely used delay-insensitive input encoding for both the inputs and outputs. It has been found out from exhaustive simulations that incorporating redundancy into the logic actually benefits in terms of delay, but a non-redundant implementation proves to be beneficial with respect to power and area parameters.**

I. INTRODUCTION

Future deep sub-micron technologies are characterized by parametric variations of devices. The latest Semiconductor Industry Association's ITRS design update projects increase of parameter uncertainty from a current 10% to 25% by 2020 [1]. In such a scenario, self-timed (ST) design gathers interest as a promising solution. This is mainly because ST circuits, in general, guarantee the correctness of operation irrespective of delays encountered in the design components or in the communicating signal wires, as they have the innate ability to absorb the deviations/variations of device characteristics. Although it is an attractive alternative to conventional digital logic design, it can be noticed that the vast majority of existing commercial EDA tools ideally support synchronous circuits. Therefore, in order to utilize the sophistication and advantages offered by industry-standard EDA tools and synchronous resources (standard cells), an attempt was made to realize ST logic (especially, a ST full adder design) and also validate them using the above in [2]. Additionally, realization of higher order C-elements functionality using standard cells was done, whilst preserving the property of indication (completion). ST full adder designs based on different approaches [3] – [8] were also realized in a similar fashion using elements of a standard cell library. However, a majority of the above approaches consider implementation targeting a widely used delay-insensitive (DI) encoding scheme, namely dual-rail encoding (DRE). This paper considers implementation, using a mixture of two well known DI encoding schemes for inputs and elucidates the benefits gained by such an approach for the case of a robust ST ripple carry adder (RCA) realization.

A major part of this research is funded by EPSRC, UK through the SEDATE project grant EP/D052238/1.

II. DATA ENCODING AND HANDSHAKING PROTOCOL

In this paper, we shall restrict our focus to full adder designs adhering to the 4-phase handshake protocol employing DI encoding; the robust and classic approach rooted in Muller's pioneering work [9].

Though 1-of-2 (DR) and 1-of-4 data encodings are the well-known DI codes, the DR code has been widely preferred owing to its simplicity and the resulting ease of circuit design. In fact, it is the simplest member of the general family of DI *m*-of-*n* codes [10]. In a DRE scheme, a data bit x is encoded into two wires, namely $x1$ and $x0$, where $x1$ and $x0$ are identified as true and false bits respectively. A logic '1' is represented by $x1$ assigned a logic '1' and $x0$ assigned a logic '0', while a logic '0' is represented in the reverse manner. The state of $x1$ and $x0$, both becoming '0' is referred to as the spacer state and both $x1$ and $x0$ are not allowed to become '1' simultaneously, as this is an invalid and illegal state. While DRE is used to represent only one bit of information, a 1-of-4 code, on the other hand, can be used to represent two non-redundant bits of information at a time by asserting only one of the four physical lines as logic high, as shown in Table 1. However, as in DRE, an all-zeroes state represents the spacer.

TABLE I. INPUT DATA REPRESENTATION IN DUAL-RAIL AND 1-OF-4 ENCODING STYLES

Single-rail inputs	**DRE**	**1-of-4 encoding**
a b	***(a1 a0); (b1 b0)***	***(i0 i1 i2 i3)***
00	(01); (01)	(0001)
01	(01); (10)	(0010)
10	(10); (01)	(0100)
11	(10); (10)	(1000)

It can be noticed from Table I, that for representation of 2 bits (a, b) of information, a 1-of-4 encoding approach would require only half as many transitions as that of a DRE approach. Consequently, the dynamic power dissipation of the former scheme is very likely to be better than that of the latter, due to reduced switching activity. This phenomenon was confirmed with the practical example of an ARM thumb instruction decoder in [11].

Both these coding schemes are known to be unordered [12]. A binary coding scheme is said to be unordered, when none of its code words is contained in any other codeword. In simple terms, the positions of ones in a codeword are never a subset of the positions of ones in a different codeword. When a DR code and a 1-of-4 code are used to represent exactly one bit and two bits of information respectively, they are said to be complete [13]. A code is said to be complete if and only if it contains all code words, as implied by its definition. Even with one missing codeword, it would be labeled 'incomplete'.

The 4-phase handshake protocol is also known as the return-to-zero protocol, wherein input data alternates between a sequence of valid data and a sequence of empty data (all zeroes, also called spacer). It is explained through figure 1, using a simple DR encoded data bus. Nevertheless, the explanation remains valid for encoding using any DI code.

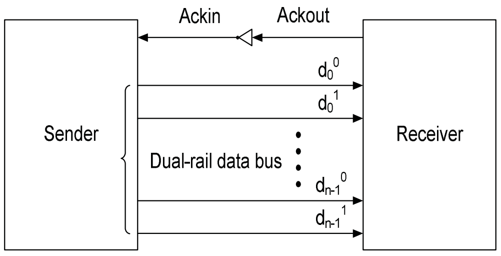

Figure 1. Four-phase handshake protocol

The 4-phase protocol consists of the following 4 steps:

- The DR data bus is initially in the spacer state. The sender transmits the codeword (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. it is driven to the spacer state).
- After an unbounded (positive), but finite amount of time, the receiver drives the Ackout (Ackin) wire low (high); thereby the system is made ready to proceed with the next transaction.

III. Adder Designs based on Hybrid Input Encoding

The hybrid input encoding (HIE) approach comprises of two different encoding schemes, adopted for the adder inputs: 1-of-4 encoding scheme for the augend and addend bits combined, and a DRE scheme for the input carry. The assignment of 1-of-4 encoding states for the augend and addend inputs are as mentioned in Table I. However, a different assignment can also be made. The sum and carry outputs are encoded in a DR format. Let the augend and addend inputs of the ST full adder be identified by a 1-of-4 codeword as (*i*0, *i*1, *i*2 and *i*3) and let *cin*1 and *cin*0 be the DR carry input. The adder's sum and carry outputs are specified by *Sum*1, *Sum*0 and *Cout*1 and *Cout*0 respectively. The general equations governing the DR sum and carry outputs would then be given by,

$$Sum1 = i3cin1 + i2cin0 + i1cin0 + i0cin1 \quad (1)$$

$$Sum0 = i3cin0 + i2cin1 + i1cin1 + i0cin0 \quad (2)$$

$$Cout1 = i2cin1 + i1cin1 + i0cin0 + i0cin1 \quad (3)$$

$$Cout0 = i3cin0 + i3cin1 + i2cin0 + i1cin0 \quad (4)$$

Amongst the different ST design methods proposed [3] – [8], [8] is suitable for logic implementation with any generic *m*-of-*n* codes, while the remaining suit realizations primarily targeting a DRE approach. Hence, the proposed adder designs would be compared, only with that resulting from [8]. Methods [3] and [6] can be used for function block realization (asynchronous equivalent of a synchronous combinatorial logic circuit), pertaining to strongly indicating or weakly indicating regimes; [4] and [8] enable function block realization corresponding to strong-indication alone and [7] facilitates function block implementation corresponding to the weak-indication timing model. Function blocks need to be indicating, apart from satisfying the required functionality. This property enables them to be transparent to handshaking, as implemented by their surrounding latches.

Function blocks could adhere to strong-indication or weak-indication timing models: strongly indicating – if no outputs (spacer/valid) are produced until all inputs (spacer/valid) have arrived, and weakly indicating – if some outputs (spacer/valid) could be produced based on even a subset of the inputs (spacer/valid). However, in the latter case, at least one output (spacer/valid) should not have been produced till all the inputs (spacer/valid) have arrived. The above indication criteria have been formulated by Seitz in [3]. Though [5] deals with ST implementation of Boolean functions, it suffers from certain drawbacks. For the worst scenario of all the false outputs of a function block evaluating to logic high, when suitable valid input data has been applied, all the sum terms of the monotonic DR network would have become enabled. When spacer is applied, even with a single sum term becoming disabled, and with OR-network and CE-network being reset, all the false outputs may evaluate to the correct empty state. This is a problematic situation, as transitions on the other intermediate gate output nodes would not get properly acknowledged, thereby giving room for creation of gate orphans. Gate orphans are unacknowledged transitions on gate output nodes. Though they may not necessarily be hazardous, they are undesirable as they can lead to erroneous output states. Hence, timing assumptions are necessary to guarantee proper ST operation. Moreover, the design method is bound to suffer from high power dissipation, since all the sum terms are activated for the worst case, leading to high switching activity.

It is to be noted that all the above methods are bound by physical or practical limitations, in that, with increase in the number of inputs, either the designs become physically unrealizable, mainly because of the fact that there occurs a linear growth in the size of the canonical product term which is accompanied by an exponential increase in the input state space or that the synthesis procedure cannot be expected to terminate in a realistic amount of time, though a practical solution is feasible. In general, indicating synthesis solutions for combinatorial logic functions are large, more so for functions with several inputs. But they inherently consist of the attractive features of asynchronous design; low EMI, high modularity, elasticity, power consumption only for useful activity and robustness, by being tolerant to variations in supply, noise, process and temperature variations. Hence, indicating synthesis solutions prominently figure in data path logic realizations, in that, the resulting circuitry is usually iterative; not the case with arbitrary combinational logic. Nevertheless, it should be noted that methods [7] and [8] would lead to synthesizable solutions, as they incorporate speed-independent decomposition.

A ST full adder design based on [8], conforming to the HIE approach, is shown in figure 2. As mentioned earlier, it can be classified as strongly indicating, thereby the DR sum and carry outputs acknowledge the arrival of all the inputs. Henceforth, we shall refer to this as Toms_HIE adder.

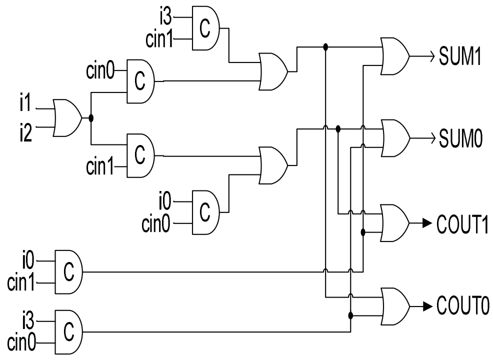

Figure 2. Toms' ST full adder based on hybrid input encoding

Two novel ST full adder designs have been proposed in this work, based on the HIE approach. One of these does not contain logic redundancy and we shall identify it by the terminology proposed_HIE_NRL adder; the other design encompassing logic redundancy shall be identified as proposed_HIE_RL adder, where the acronyms NRL and RL expand as **n**on-**r**edundant **l**ogic and **r**edundant **l**ogic respectively.

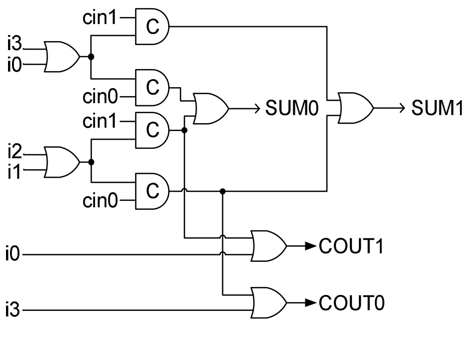

Figure 3. Proposed ST full adder design without redundant logic

Figure 3 shows the realization of the proposed_HIE_NRL ST full adder. This adder pertains to weak-indication, wherein only the DR sum output acknowledges the arrival of all the inputs, while the DR carry output need not. This property enables the DR carry to propagate faster from a lower order adder stage to its successive higher order adder stage, in the cascade. It is also well known that a valid combinatorial cascade of strong/weak-indication function blocks is itself a strong/weak-indication function block [3].

The following figure shows the proposed_HIE_RL adder design. Similar to the previous one, this full adder is also weakly indicating, with the responsibility of indication wholly entrusted on the DR sum output.

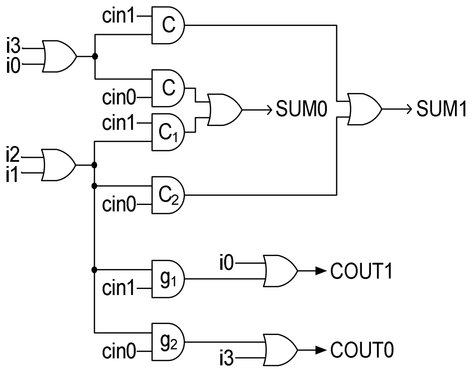

Figure 4. Proposed ST full adder design with logic redundancy

In figure 4, gates C_1 and C_2 denote 2-input C-elements, while gates g_1 and g_2 represent 2-input AND gates. It can be noticed in the diagram, that the logic realized by C_1 and C_2 are equivalent to that of g_1 and g_2 respectively, for transitions. Hence, redundancy is made implicit in the design. This proves to be beneficial in two ways. During the spacer phase, all the sum outputs could be reset in a parallel fashion, as the DR carry output of the k^{th} stage could be reset based on its 1-of-4 encoded augend and addend inputs, and the DR sum output of the $(k+1)^{th}$ stage depends only on the DR carry input from the k^{th} stage. However, this unique feature of fast reset could not be captured using a static timing analyzer that is predominantly used for timing analysis of synchronous circuits, and in this work, designs have been implemented using synchronous resources and validated using widely used industry-standard synchronous tools. There is also a benefit in terms of improvement in delay during the valid data phase. This would become obvious by comparing the designs portrayed by figures 3 and 4; it can be observed that the carry propagation path delay is lesser in case of proposed_HIE_RL adder than the proposed_HIE_NRL adder. This is further substantiated by the results mentioned in the next section.

IV. SIMULATION MECHANISM, RESULTS AND DISCUSSION

The ST full adder designs, based on HIE, are analyzed using the delay-insensitive version of an *n*-bit RCA topology, depicted in figure 5. As can be seen, the augend and addend adder inputs are 1-of-4 encoded, while its carry inputs, sum and carry outputs are DR encoded.

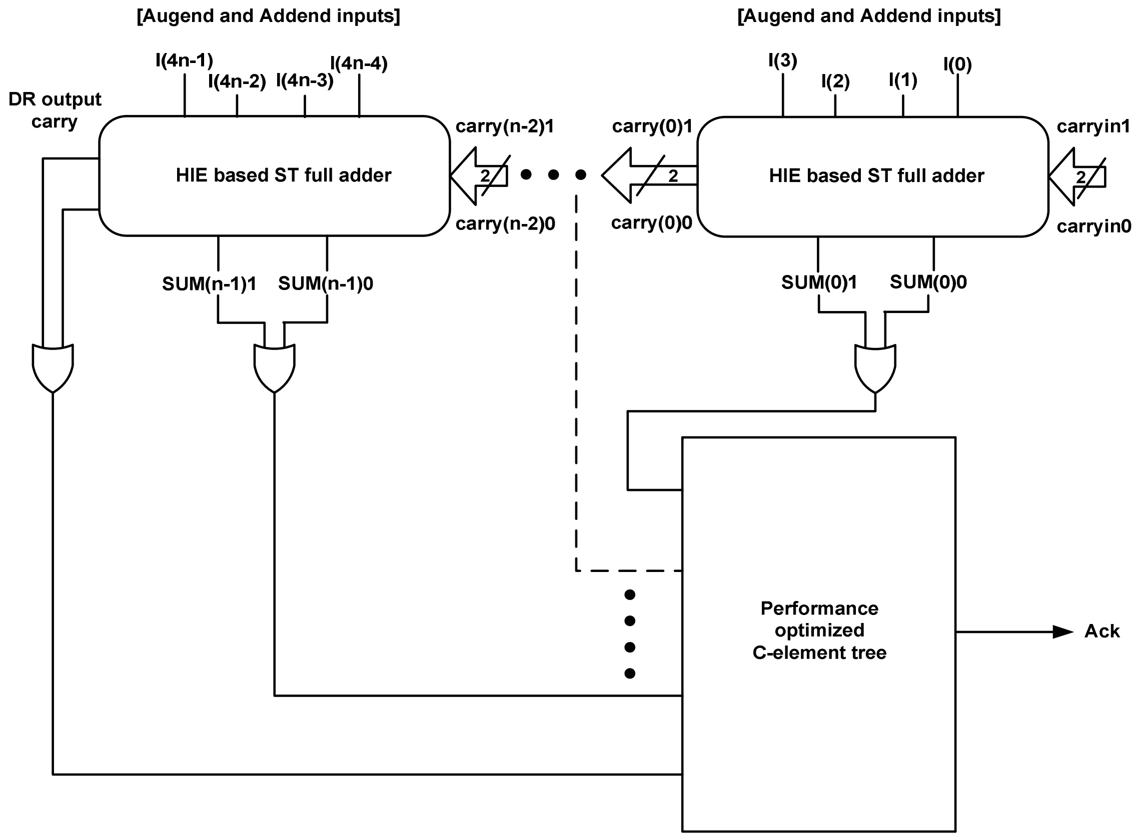

Figure 5. Delay-insensitive (self-timed) version of an *n*-bit ripple carry adder topology. The target library (130nm Faraday bulk CMOS process) features an AND gate with a maximum fan-in of 4 and an OR gate with a maximum fan-in of 3

All the adder's outputs have been uniformly configured to possess fanout-of-4 drive strength, while their inputs are configured with the driving capability of a minimum sized inverter in the library. Similar delay-optimized completion detection (CD) circuits were used for all the ST adders, to maintain uniformity. Minimum sized buffer cells were provided within all the adder modules, mainly to eliminate timing violations, that results from a single acknowledge input feeding all the adder outputs, in every stage of the cascade. Experimentation has been carried out across the typical, worst and best case corners of the high-speed 130nm Faraday CMOS process (which is compatible with the 130nm UMC CMOS foundry process). Cadence NC-Verilog has been used for functional simulation and also to obtain the switching activity files for all the gate level simulations, while Synopsys PrimeTime and PrimeTime PX have been used for delay, cells area and power evaluation respectively, inclusive of wire load information. A virtual clock has been used, only to act as a remote reference to guide the application of inputs to the ST adders at a specific data rate and does not form a part of the designs in any way. The minimum support for asynchronous logic, offered by synchronous tools has been exploited, by avoiding timing loop breaking while performing static timing analysis. The inputs of all the ST full adders correspond to the input trace of a simple combinatorial benchmark circuit, *dc1*, of the MCNC benchmark set. The inputs are assumed to arrive from the environment and are fed to the adders every 30ns, 50ns and 20ns for the typical, worst and best case library specifications respectively.

TABLE II. AREA METRIC FOR DIFFERENT ST ADDERS

Adder realization style	Cells area (μm^2)
Seitz_DRE (Strong) [3]	7100
Seitz_DRE (Weak) [3]	6276
Singh_DRE (Strong) [4]	7364
DIMS_DRE (Strong) [6]	8932
DIMS_DRE (Weak) [6]	9508
Folco et al._DRE (Weak) [7]	5476
Toms_DRE (Strong) [8]	6404
Proposed_DRE (Weak) [2]	5924
Toms_HIE (Strong) [8]	4868
Proposed_HIE_NRL (Weak)	3940
Proposed_HIE_RL (Weak)	4260

The simulation results pertain to a 32-bit DI RCA, constructed using different ST full adder modules. Table II lists the area metric for realization of a 32-bit DI RCA based on various ST full adder modules. Table III gives the delay metric of the different ST adders and Table IV lists the power components of the adders, for a typical case corner. In a similar manner, Tables V and VI and Tables VII and VIII specify the design metrics of the various ST adders, across worst case and best case library corners respectively. It can be observed from the previous tabular column, that amongst all the other ST adders, Toms_HIE adder has the best area metric. Nevertheless, it is inferior to the proposed_HIE_NRL and proposed_HIE_RL adders by 23.6% and 14.3% respectively.

TABLE III. DELAY METRICS OF VARIOUS ST ADDERS (TYPICAL CASE – 1.2V, 25°C)

Adder realization style	Maximum data path delay (ns)	Function block delay (ns)
Seitz_DRE (Strong)	20.54	19.88
Seitz_DRE (Weak)	11.36	10.71
Singh_DRE (Strong)	26.01	25.34
DIMS_DRE (Strong)	22.45	21.79
DIMS_DRE (Weak)	21.04	20.39
Folco et al._DRE (Weak)	14.67	14.02
Toms_DRE (Strong)	17.82	17.16
Proposed_DRE (Weak)	10.52	9.86
Toms_HIE (Strong)	17.78	17.12
Proposed_HIE_NRL (Weak)	14.04	13.38
Proposed_HIE_RL (Weak)	10.13	9.48

TABLE IV. POWER COMPONENTS OF ST ADDERS (TYPICAL CASE)

Adder realization style	Power dissipation components		
	Total (μW)	*Dynamic (μW)*	*Leakage (nW)*
Seitz_DRE (Strong)	248.57	246.34	2225.66
Seitz_DRE (Weak)	198.52	196.65	1875.81
Singh_DRE (Strong)	246.40	244.36	2040.49
DIMS_DRE (Strong)	186.43	184.28	2147.89
DIMS_DRE (Weak)	195.67	193.46	2206.59
Folco et al._DRE (Weak)	183.85	182.27	1579.70
Toms_DRE (Strong)	186.18	184.40	1785.47
Proposed_DRE (Weak)	192.21	190.63	1586.49
Toms_HIE (Strong)	151.46	150.00	1462.99
Proposed_HIE_NRL (Weak)	160.32	159.08	1242.43
Proposed_HIE_RL (Weak)	162.91	161.58	1323.54

TABLE V. DELAY METRICS OF VARIOUS ST ADDERS (WORST CASE – 1.08V, 125°C)

Adder realization style	Maximum data path delay (ns)	Function block delay (ns)
Seitz_DRE (Strong)	35.02	33.82
Seitz_DRE (Weak)	19.62	18.44
Singh_DRE (Strong)	44.81	43.64
DIMS_DRE (Strong)	38.70	37.54
DIMS_DRE (Weak)	36.37	35.22
Folco et al._DRE (Weak)	25.11	23.94
Toms_DRE (Strong)	30.39	29.22
Proposed_DRE (Weak)	18.10	16.94
Toms_HIE (Strong)	30.30	29.12
Proposed_HIE_NRL (Weak)	24.12	22.95
Proposed_HIE_RL (Weak)	17.24	16.07

In Tables III, V and VII, function block delay (FBD) specifies the maximum delay encountered for traversal of a logic path from the least significant adder stage to the most significant stage. Maximum data path delay (MDPD) is the summation of FBD and the delay associated with the CD circuitry. The CD logic comprises of all the 2-input OR gates, used to combine the DR sum and final stage carry outputs and the C-element tree, which is used to synchronize (indicate) the arrival of all the adder outputs. In Tables IV, VI and VIII, total power dissipation denotes the sum of dynamic and static (leakage) power parameters. In turn, dynamic power dissipation is the gross of switching and internal power components. Both delay and power figures of the various ST adders report a consistency in all the three evaluation corners.

TABLE VI. POWER COMPONENTS OF ST ADDERS (WORST CASE)

Adder realization style	Power dissipation components		
	Total (μW)	*Dynamic (μW)*	*Leakage (nW)*
Seitz_DRE (Strong)	119.62	114.31	5312.77
Seitz_DRE (Weak)	95.77	91.24	4529.75
Singh_DRE (Strong)	119.53	114.57	4957.46
DIMS_DRE (Strong)	90.48	85.18	5301.35
DIMS_DRE (Weak)	95.40	89.90	5494.11
Folco et al._DRE (Weak)	88.79	84.96	3824.22
Toms_DRE (Strong)	90.28	85.94	4336.19
Proposed_DRE (Weak)	92.90	88.99	3913.68
Toms_HIE (Strong)	72.96	69.46	3502.02
Proposed_HIE_NRL (Weak)	77.02	74.06	2957.59
Proposed_HIE_RL (Weak)	78.41	75.23	3182.09

TABLE VII. DELAY METRICS OF VARIOUS ST ADDERS (BEST CASE – 1.32V, -40°C)

Adder realization style	Maximum data path delay (ns)	Function block delay (ns)
Seitz_DRE (Strong)	13.47	13.04
Seitz_DRE (Weak)	7.37	6.95
Singh_DRE (Strong)	16.99	16.56
DIMS_DRE (Strong)	14.69	14.27
DIMS_DRE (Weak)	13.69	13.27
Folco et al._DRE (Weak)	9.65	9.23
Toms_DRE (Strong)	11.71	11.28
Proposed_DRE (Weak)	6.89	6.46
Toms_HIE (Strong)	11.74	11.32
Proposed_HIE_NRL (Weak)	9.21	8.78
Proposed_HIE_RL (Weak)	6.68	6.25

TABLE VIII. POWER COMPONENTS OF ST ADDERS (BEST CASE)

Adder realization style	Power dissipation components		
	Total (μW)	*Dynamic (μW)*	*Leakage (nW)*
Seitz_DRE (Strong)	466.71	465.59	1115.92
Seitz_DRE (Weak)	373.43	372.49	942.21
Singh_DRE (Strong)	458.59	457.61	977.70
DIMS_DRE (Strong)	351.76	350.70	1059.72
DIMS_DRE (Weak)	367.92	366.84	1083.13
Folco et al._DRE (Weak)	343.37	342.61	762.67
Toms_DRE (Strong)	347.86	346.99	861.61
Proposed_DRE (Weak)	359.57	358.81	758.83
Toms_HIE (Strong)	283.99	283.28	714.69
Proposed_HIE_NRL (Weak)	299.99	299.39	604.87
Proposed_HIE_RL (Weak)	304.70	304.05	648.17

We shall first consider the issue with DRE and HIE based ST full adder designs separately, and then proceed towards a combined comparison. It can be noticed that among all the full adder designs based on DRE, the proposed_DRE full adder [2] yields the minimum delay values, both in terms of MDPD and FBD, across all the three corners. In terms of total power dissipation, Folco et al._DRE is economical. Nevertheless, in terms of the static power metric, the proposed_DRE adder is comparable with Folco et al._DRE adder.

Among the adders which adopt HIE, Toms_HIE adder is found to be power efficient, mainly in terms of total and dynamic power parameters. This is evident from the results highlighted in Tables IV, VI and VII. Between the two proposed full adder designs viz. proposed_HIE_NRL and proposed_HIE_RL, the former is found to be better in terms of all the power components. This is because of reduced activity and lesser number of cells due to absence of redundant logic. However, proposed_HIE_NRL adder is expensive than Toms_HIE adder with respect to total and dynamic power dissipation, on an average, across all the three corners, by 5.7% and 6.1% respectively. Notwithstanding, with respect to the static power component, Toms_HIE adder is expensive than both proposed_HIE_NRL and proposed_HIE_RL adders, on an average, across all the three corners by 18.1% and 10.3% respectively. With respect to MDPD and FBD parameters, Toms_HIE adder suffers an increase compared to the proposed_HIE_NRL adder, on an average, across all the three corners by 26.6% and 27.9%, which is considerable. For a similar comparison with the proposed_HIE_RL adder, Toms_HIE exhibits a heavy delay penalty, reporting degradation (increase in delay) to the tune of 75.7% and 81% respectively.

Based on a combined overall comparison between various ST full adders adopting DRE and HIE, we find that the proposed_HIE_RL adder features the best MDPD and FBD values, in comparison with that of the proposed_DRE adder. On an average, for all the three cases combined, the former reports reduction over the latter by 3.8% (in MDPD) and 4.1% (in FBD) respectively. For a comprehensive comparison, the reduction in delay should also be simultaneously viewed from a power and area perspective, wherein the former proves to be economical with less total power consumption (across all the library cases) and reduced area occupancy by 15.4% and 28.1% respectively.

V. Conclusion

Throughout this article, the term 'self-timed' has been used to generalize the notions of quasi-delay-insensitivity (delay-insensitivity with isochronic fork assumption included [14]) and speed-independency. This paper has presented two new ST designs for a full adder functionality with HIE – one consisting of logic redundancy and the other without any redundant logic. While the augend and addend bits of every full adder module is 1-of-4 encoded, the input and output carries as well as the sum output are encoded in a DR format. The motivation being that a 1-of-4 code experiences only half the transitions as that of a DR code and therefore it is most likely to yield a power efficient solution.

The designs have been analyzed on the basis of a delay-insensitive (in fact, quasi-delay-insensitive) RCA. A delay-insensitive RCA, constructed using only DR encoded full adder blocks, would have a similar structure as shown in figure 5. The logic has been implemented using the elements of a standard cell library and validated across typical, worst and best case library targets. Since this work relies on utilizing synchronous standard cells for realizing robust ST designs, comparison with [15], or improvisations based on it, is not possible, as they are based on the requirement of custom macros (proprietary NCL macro cells) for a cell library.

While the proposed adders feature the optimum delay and area metrics, the full adder design employing HIE based on [8] is found to be somewhat economical in terms of total average power and dynamic power parameters. Nevertheless, the proposed adders report the least static power in all the cases. This is mainly attributable to the less number of C-gates that were required. However, in comparison with a standard synchronous RCA, where a full adder is constructed using two half adder modules, the proposed_HIE_NRL adder (which occupies the least area amongst all the ST adders) is found to be 2.9× expensive in terms of area. On the other hand, the synchronous adder is found to exhibit 30.5% reduced delay compared to the proposed_HIE_RL adder (which features the least delay), on an average, across all three process corners.

References

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

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

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

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

[5] I. David et al., "An efficient implementation of Boolean functions as self-timed circuits," *IEEE Trans. on Comp.,* 41(1), pp. 2-11, Jan. '92.

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

[7] B. Folco et al., "Technology mapping for area optimized quasi-delay-insensitive circuits," *Proc. IFIP VLSI-SoC*, pp. 55-69, 2005.

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

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

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

[11] D.W. Lloyd and J.D. Garside, "A practical comparison of asynchronous design styles," *Proc. 7th ASYNC,* pp. 36-45, 2001.

[12] B. Bose, "On Unordered Codes," *IEEE Trans. on Computers*, vol. 40, no. 2, pp. 125-131, February 1991.

[13] S.J. Piestrak and T. Nanya, "Towards totally self-checking delay-insensitive systems," *Proc. 25th Intl. Symposium on Fault-Tolerant Computing*, pp. 228-237, 1995.

[14] A.J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," *Proc. 6th MIT Conf. on Adv. Res. in VLSI*, pp. 263-278, 1990.

[15] K.M. Fant and S.A. Brandt, "Null convention logic: a complete and consistent logic for asynchronous digital circuit synthesis," *Proc. Intl. Conf. on Application-specific Sys., Arch., and Proc.,* pp. 261-273, 1996.