Parallel Self-Timed Delay Insensitive Adder design based on Hybrid Encoding

VIDYA S. NAIR
M.E VLSI DESIGN
J.P COLLEGE OF ENGINEERING

MR.M.VIGNESH BARATHI
AP/ECE DEPARTMENT
J.P COLLEGE OF ENGINEERING

ABSTRACT---

Self timed pipelined parallel carry-look-ahead adders are designed based on delay-insensitive (DI) Hybrid Encoding. Basic binary gates employ dual-rail encoded data, which include timing information in themselves. One version uses wave pipelining and the other delay-insensitive pipelining with a request-acknowledge data transfer protocol. Comparisons have been carried out with respect to various self-timed full adder design 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 A practical implementation is provided along with a completion detection unit. The implementation is regular and does not have any practical limitations of high fan outs. A high fan-in gate is required though but this is unavoidable for asynchronous logic and is managed by connecting the transistors in parallel. Simulations have been performed using an industry standard toolkit that verifies the practicality and superiority of the proposed approach over existing asynchronous adders.

I. INTRODUCTION

Binary addition is the single most important operation that a processor performs. Most of the adders have been designed for synchronous circuits even though there is a strong interest in clockless /asynchronous processors/circuits [1]. Asynchronous circuits do not assume any quantization of time. Therefore, they hold great potential for logic design as they are free from several problems of clocked (synchronous) circuits. In principle, logic flow in asynchronous circuits is controlled by a request acknowledgment handshaking protocol to establish a pipeline in the absence of clocks. Explicit handshaking blocks for small elements, such as bit adders, are expensive.Therefore, it is implicitly and efficiently managed using dual-rail carry propagation in adders. A valid dual-rail carry output also provides acknowledgment from a single-bit adder block. Thus, asynchronous adders are either based on full dual-rail encoding of all signals.

ST full adder designs based on different approaches 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.

II. DATA ENCODING AND HANDSHAKING PROTOCOL

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 [2]. 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.

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 [4]. Both these coding schemes are known to be unordered. A binary coding scheme is said to be unordered, when one 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 [3]. 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.

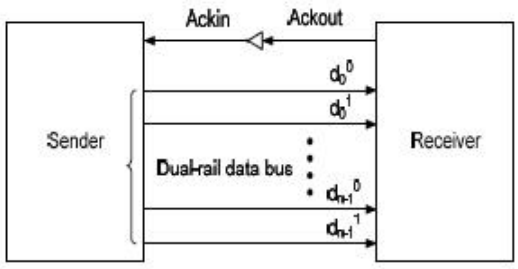

Fig a:Four-phase hand shake protocol

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. BACKGROUND-PASTA

In this section, the architecture and theory behind PASTA is presented. The adder first accepts two input operands to perform half additions for each bit. Subsequently, it iterates using earlier generated carry and sums to perform half-additions repeatedly until all carry bits are consumed and settled at zero level.

A. Architecture of PASTA

The general architecture of the adder is shown in Fig.1. The selection input for two-input multiplexers corresponds to the Req handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL = 0 and will switch to feedback/carry paths for subsequent iterations using SEL = 1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values.

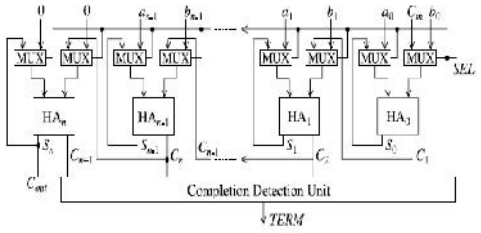

Fig. 1. General block diagram of PASTA.

B. State Diagrams

In Fig. 2, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (Ci+1 Si) pair where Ci+1, Si represent carry out and sum values, respectively, from the ith bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state (11) cannot appear.

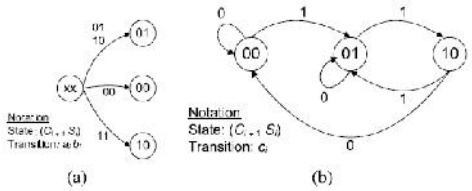

Fig. 2. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

During the iterative phase (SEL = 1), the feedback path through multiplexer block is activated. The carry transitions (C_i) are allowed as many times as needed to complete the recursion. From the definition of fundamental mode circuits, the present design cannot be considered as a fundamental mode circuit as the input–outputs will go through several transitions before producing the final output. It is not a Muller circuit working outside the fundamental mode either as internally, several transitions will take place, as shown in the state diagram. This is analogous to cyclic sequential circuits where gate delays are utilized to separate individual states.

IV. PROPOSED 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,

$Sum1 = i3cin1 + i2cin0 + i1cin0 + i0cin1$ (1)
$Sum0 = i3cin0 + i2cin1 + i1cin1 + i0cin0$ (2)
$Cout1 = i2cin1 + i1cin1 + i0cin0 + i0cin1$ (3)
$Cout0 = i3cin0 + i3cin1 + i2cin0 + i1cin0$ (4)

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. 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. A ST full adder design 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.

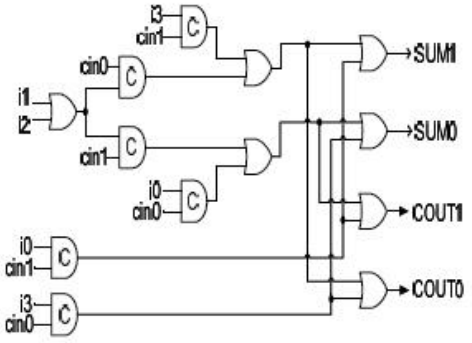

Fig 3:ST Adder based on Hybrid Input Encoding

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].

V. 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. 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.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). Multisim has been used for functional simulation and also to obtain the switching activity files for all the gate level simulations. 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.

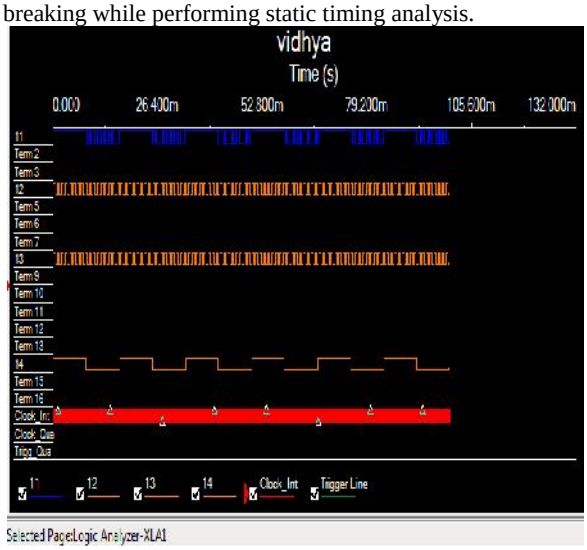

Fig 4: Multisim simulation of proposed HIE Adder

However, even the average case results for dual-rail adders cannot beat the worst-case performance by RCA. We identify a few reasons for this anomaly as follows.

1) All DI designs under evaluation are based on data driven dynamic logic circuits. The performance of these dynamic circuits can be optimized using larger nMOS transistors in the pull-down network. However, it will considerably increase precharge delay as the load capacitance will also increase for pMOS transistors. Therefore, we have kept the standard sizing of nMOS and pMOS transistors that will result in equal rise/fall time in this circuit.

2) The original results were obtained without consideration of factors such as layout, wiring delays, stray capacitance, and noise .
3) Dynamic logic circuits are not considered to be a good design choice for deep submicrometer technologies and beyond . They can be slow or malfunction for some particular operands or conditions due to charge sharing problems or the presence of noise.
4) The data driven dynamic logic cannot attain the same performance as the pure dynamic logic circuits since they often include more than one pull-up pMOS transistor which increases the switching delay.

VI. 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[7]) 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. 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 [10], 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 [11] 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.

VII. REFERENCES

[1] D. Geer, "Is it time for clockless chips? [Asynchronous processorchips]," *IEEE Comput.*, vol. 38, no. 3, pp. 18–19, Mar. 2005.
[2] T. Verhoeff, "Delay-insensitive codes: an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1-8, 1988.
[3] S.J. Piestrak and T. Nanya, "Towards totally self-checking delayinsensitive systems," *Proc. 25th Intl. Symposium on Fault-TolerantComputing*, pp. 228-237, 1995.
[4] D.W. Lloyd and J.D. Garside, "A practical comparison of asynchronousdesign styles," *Proc. 7th ASYNC*, pp. 36-45, 2001.
[5] F.-C. Cheng, S. H. Unger, and M. Theobald, "Self-timed carry-lookahead adders," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 659–672, Jul. 2000.
[6] S. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *IEE Proc. Comput. Digital Tech.*, vol. 143, no. 5,pp. 301–307, Sep. 1996.
[7] A.J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," *Proc. 6th MIT Conf. on Adv. Res. in VLSI*, pp. 263-278, 1990.
[8] F.-C. Cheng, S. H. Unger, and M. Theobald, "Self-timed carry-lookahead adders," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 659–672, Jul. 2000.
[9] T. Verhoeff, "Delay-insensitive codes: an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1-8, 1988.
[10] 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.
[11] W.B. Toms, "Synthesis of Quasi-Delay-Insensitive Datapath Circuits,"*PhD thesis*, University of Manchester, 2006