International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.05, Issue.02,
February-2017,
Pages:0246-0251

www.ijvdcs.org

Design of a Parallel Self-Timed Adder with Recursive Approach Using Verilog HDL

S. SAI PRASANNA[1], M. CHANDRA MOAHAN[2]

[1]PG Scholar, Malla Reddy College of Engineering, Maisammaguda, Secunderabad, TS, India,
Email: saiprasannashaganti@gmail.com.

[2]Assistant Professor, Malla Reddy College of Engineering, Maisammaguda, Secunderabad, TS, India,
Email: mohanmulinti@gmail.com.

Abstract: As technology scales down into the lower nanometer values power, delay, area and frequency becomes important parameters for the analysis and design of any circuits. This brief presents a parallel single-rail self-timed adder. It is based on a recursive formulation for performing multi-bit binary addition. The operation is parallel for those bits that do not need any carry chain propagation. Thus, the design attains logarithmic performance over random operand conditions without any special speedup circuitry or look-ahead schema. A practical implementation is provided along with a completion detection unit. The implementation is regular and does not have any practical limitations of high fanouts. 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 verify the practicality and superiority of the proposed approach over existing asynchronous adders.

Keywords: Digital Arithmetic, Binary Adders, Recursive Adder.

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 clock less 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 On the other hand, wave pipelining (or maximal rate pipelining) is a technique that can apply pipelined inputs before the outputs are stabilized [7]. The proposed circuit manages automatic single-rail pipelining of the carry inputs separated by propagation and inertial delays of the gates in the circuit path.

II. CONVENTIONAL ADDERS

As mentioned previously, adders in VLSI digital systems use binary notation. In that case, add is done bit by bit using Boolean equations. Consider a simple binary add with two n-bit inputs A;B and a one-bit carry-in C_{in} along with n-bit output S.

$$S = A + B + C_{in}$$

where $A = a_{n-1}, a_{n-2}\ldots\ldots a_0$;$B = b_{n-1}, b_{n-2}\ldots\ldots b_0$.

The + in the above equation is the regular add operation. However, in the binary world, only Boolean algebra works. For add related operations, AND, OR and Exclusive-OR (XOR) are required. In the following documentation, a dot between two variables (each with single bit), e.g. a.b denotes 'a AND b'. Similarly, a + b denotes 'a OR b' and a⊕b denotes 'a XOR b'. Considering the situation of adding two bits, the sum s and carry c can be expressed using Boolean operations mentioned above.

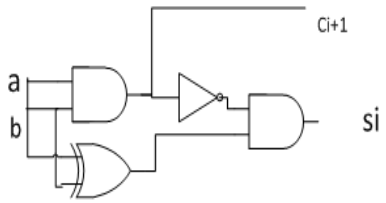

Fig1. 1-bit Half Adder.

$$s_i = a_i \oplus b_i$$
$$c_{i+1} = a_i . b_i$$

The Equation of c_{i+1} can be implemented as shown in Fig1. In the figure, there is a half adder, which takes only 2 input bits. The solid line highlights the critical path, which indicates the longest path from the input to the output. Equation of c_{i+1} can be extended to perform full add operation, where there is a carry input.

$$s_i = a_i \oplus b_i$$
$$c_{i+1} = a_i . b_i + a_i . c_i + b_i . c_i$$

A full adder can be built based on Equation above. The block diagram of a 1-bit full adder is shown in Figure 2. The full adder is composed of 2 half adders and an OR gate for

computing carry-out. Using Boolean algebra, the equivalence can be easily proven. To help the computation of the carry for each bit, two binary literals are introduced. They are called carry generate and carry propagate, denoted by $g_i$and p_i. Another literal called temporary sum t_i is employed as well.

There is relation between the inputs and these literals.

$$g_i = a_i . b_i$$
$$p_i = a_i + b_i$$
$$t_i = a_i \oplus b_i$$

where i is an integer and $0 \leq i < n$.

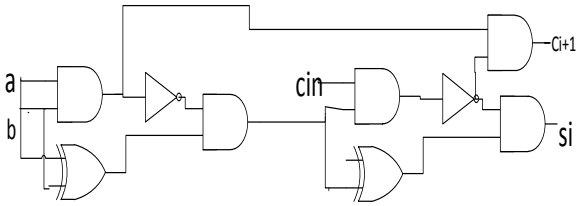

Fig.2. 1-bit Full Adder.

With the help of the literals above, output carry and sum at each bit can be written as

$$c_{i+1} = g_i + p_i . c_i$$
$$s_i = t_i \oplus c_i$$

In some literatures, carry-propagate pi can be replaced with temporary sum t_i in order to save the number of logic gates. Here these two terms are separated in order to clarify the concepts. For example, for Ling adders, only p_i is used as carry-propagate. The single bit carry generate/propagate can be extended to group version G and P. The following equations show the inherent relations.

$$G_{i:k} = G_{i:j} + P_{i:j} . G_{j-1:k}$$
$$P_{i:k} = P_{i:j} . P_{j-1:k}$$

where i : k denotes the group term from i through k.
Using group carry generate/propagate, carry can be expressed as expressed in the following equation.

$$c_{i+1} = G_{i:j} + P_{i:j} . c_j$$

The simplest way of doing binary addition is to connect the carry-out from the previous bit to the next bit's carry-in. Each bit takes carry-in as one of the inputs and outputs sum and carry-out bit and hence the name ripple-carry adder. This type of adders is built by cascading 1-bit full adders. A 4-bit ripple-carry adder is shown in Figure.3. Each trapezoidal symbol represents a single-bit full adder. At the top of the figure, the carry is rippled through the adder from c_{in} to c_{out}.

It can be observed in Fig3 that the critical path, highlighted with a solid line, is from the least significant bit (LSB) of the input (a0 or b0) to the most significant bit (MSB)of sum (s_{n-1}). Assuming each simple gate, including AND, OR and XOR gate has a delay of 2Δ and NOT gate has a delay of 1Δ. All the gates have an area of 1 unit. Using this analysis and assuming that each add block is built with a 9-gate full adder, the critical path is calculated as follows.

$a_i , b_i \rightarrow s_i = 10\Delta$
$a_i , b_i \rightarrow c_{i+1} = 9\Delta$
$c_i \rightarrow s_i = 5\Delta$
$c_i \rightarrow c_{i+1} = 4\Delta$

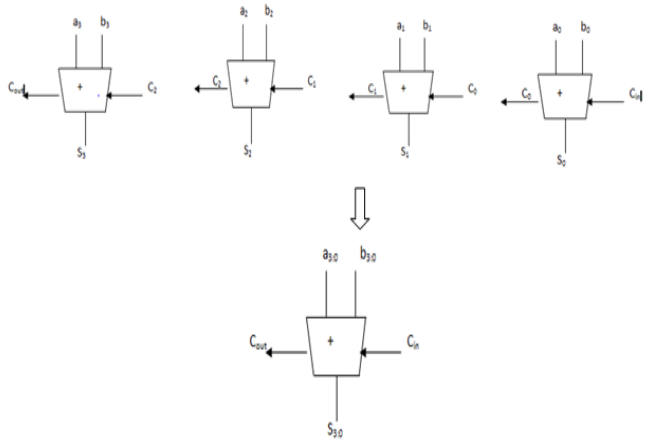

Fig.3. Ripple-Carry Adder.

The critical path, or the worst delay is

$$t_{rca} = \{9 + (n- 2) \times 4 + 5\}\Delta = \{f4n + 6\}\Delta$$

As each bit takes 9 gates, the area is simply 9n for a n-bit RCA. Simple adders, like ripple-carry adders, are slow since the carry has to to travel through every full adder block. There is a way to improve the speed by duplicating the hardware due to the fact that the carry can only be either 0 or 1. The method is based on the conditional sum adder and extended to a carry-select adder. With two RCA, each computing the case of the one polarity of the carry-in, the sum can be obtained with a 2x1 multiplexer with the carry-in as the select signal. An example of 16-bit carry-select adder is shown in Figure.4. In the figure, the adder is grouped into four 4-bit blocks. The 1-bit multiplexors for sum selection can be implemented as Figure 4 shows. Assuming the two carry terms are utilized such that the carry input is given as a constant 1 or 0:

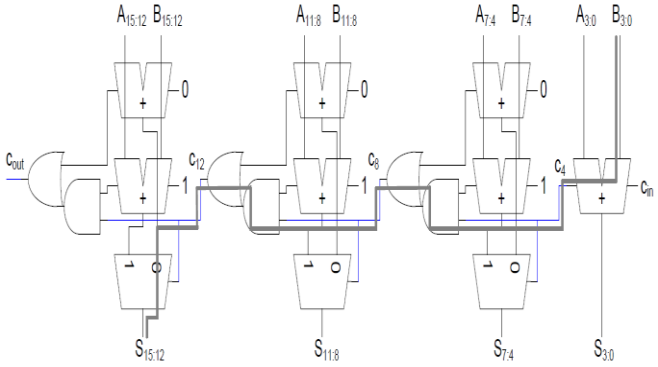

Fig.4. Carry-Select Adder.

A. Self-Timed Adders

Self timed refers to logic circuits that depend on timing assumptions for the correct operation. Self-timed adders have the potential to run faster averaged for dynamic data, as early completion sensing can avoid the need for the worst case bundled delay mechanism of synchronous circuits.

B. Pipelined Adders Using Single-Rail Data Encoding

The asynchronous Req/Ack handshake can be used to enable the adder block as well as to establish the flow of carry signals. In most of the cases, a dual-rail carry convention is used for internal bitwise flow of carry outputs. These dual-rail signals can represent more than two logic values (invalid, 0, 1), and therefore can be used to generate bit-level acknowledgment when a bit operation is completed. Final completion is sensed when all bit *Ack*signals are received (high).The carry-completion sensing adder is an example of a pipelined adder [8], which uses full adder (FA) functional blocks adapted for dual-rail carry. On the other hand, a speculative completion adder is proposed in [9]. It uses so-called abort logic and early completion to select the proper completion response from a number of fixed delay lines. However, the abort logic implementation is expensive due to high fan-in requirements.

C. Delay Insensitive Adders Using Dual-Rail Encoding

Delay insensitive (DI) adders are asynchronous adders that assert bundling constraints or DI operations. Therefore, they can correctly operate in presence of bounded but unknown gate and wire delays [2].There are many variants of DI adders, such as DI ripple carry adder (DIRCA) and DI carry look-ahead adder (DICLA). DI adders use dual-rail encoding and are assumed to increase complexity. Though dual-rail encoding doubles the wire complexity, they can still be used to produce circuits nearly as efficient as that of the single-rail variants using dynamic logic or nMOS only designs. An example 40 transistors per bit DIRCA adder is presented in [8] while the conventional CMOS RCA uses 28 transistors. Similar to CLA, the DICLA defines carry propagate, generate, and kill equations in terms of dual-rail encoding [8]. They do not connect the carry signals in a chain but rather organize them in a hierarchical tree. Thus, they can potentially operate faster when there is long carry chain. A further optimization is provided from the observation that dual rail encoding logic can benefit from settling of either the 0 or 1 path. Dual-rail logic need not wait for both paths to be evaluated. Thus, it is possible to further speed up the carry look-ahead circuitry to send carry-generate/carry-kill signals to any level in the tree. This is elaborated in [8] and referred as DICLA with speedup circuitry (DICLASP).

D. Parallel Self Timed Adders

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.

III. ARCHITECTURE OF PASTA

The general architectureof the adder is shown in Fig1. 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.

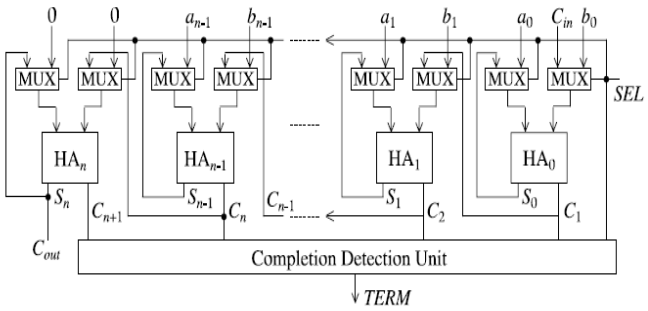

Fig5. Block diagram of PASTA.

A. State Diagrams

In Fig. 6, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (C_{i+1} S_i) pair where C_{i+1}, $S_i$represent carry out and sum values, respectively, from the i^{th}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. During the iterative phase (SEL = 1), the feedback path through multiplexer block is activated.

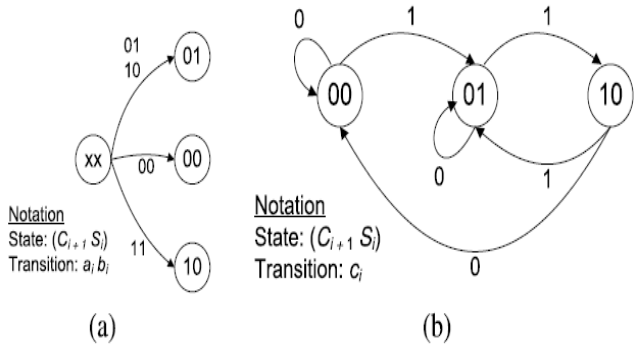

Fig.6. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

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 [4].

G. Recursive Formula for Binary Addition

Let S^j_i,and C^j_{i+1} denote the sum and carry, respectively, for i^{th} bit at the j^{th}iteration. The initial condition (j= 0) for addition is formulated as follows:

$$S^0_i = a_i \oplus b_i$$
$$C^0_{i+1} = a_i . b_i \quad (1)$$

The *j* th iteration for the recursive addition is formulated by

$$S^j_i=S^j_j \oplus C_i^{j-1} \quad 0<i<n \qquad (2)$$

$$C^j_{i+1}= S^j_j.C_i^{j-1} \quad 0<i<n \qquad (3)$$

The recursion is terminated at k^{th} iteration when the following condition is met:

$$C_n^k+C_{n-1}^k+\ldots\ldots..+ C_1^k=0 \quad 0<k<n \qquad (4)$$

Now, the correctness of the recursive formulation is inductively proved as follows.

Theorem 1: The recursive formulation of (1)–(4) will produce correct sum for any number of bits and will terminate within a finite time.

Proof: We prove the correctness of the algorithm by induction on the required number of iterations for completing the addition (meeting the terminating condition).

Basis: Consider the operand choices for which no carry propagation is required, i.e., $C^0_j= 0$ for $\forall i$, $i \in [0..n]$. The proposed formulation will produce the correct result by a single-bit computation time and terminate instantly as (4) is met.

Induction: Assume that $C^k_{i+1}!=0$ for some i^{th} bit at k^{th} iteration. Let l be such a bit for which $C^k_{l+1}= 1$. We show that it will be successfully transmitted to next higher bit in the $(k + 1)^{th}$ iteration.

As shown in the state diagram, the k^{th} iteration of l^{th} bit state(C^k_{l+1}, S^k_l) and $(l + 1)^{th}$ bit state (C^k_{l+2}, S^k_{l+1}) could be in any of (0, 0), (0, 1), or (1, 0) states. As $C^k_{l+1}= 1$, it implies that $S^k_l= 0$. Hence, from (3), $C^{k+1}_{l+1}= 0$ for any input condition between0 to l bits. We now consider the $(l + 1)^{th}$ bit state (C^k_{l+2}, S^k_{l+1}) for k^{th} iteration. It could also be in any of (0, 0), (0, 1), or (1, 0) states. In $(k+1)^{th}$ iteration, the (0, 0) and (1, 0) states from the k^{th}iteration will correctly produce output of (0, 1) following (2) and (3). For(0,1) state, the carry successfully propagates through this bit level following (3).Thus, all the single-bit adders will successfully kill or propagate the carries until all carries are zero fulfilling the terminating condition. The mathematical form presented above is valid under the condition that the iterations progress synchronously for all bit levels and the required input and outputs for a specific iteration will also being synchrony with the progress of one iteration. In the next section, we present an implementation of the proposed architecture which is subsequently verified using simulations.

B. Design of PASTA

A CMOS implementation for the recursive circuit is shown in Fig.3. For multiplexers and AND gates we have used Xilinx ISE implementations while for the XOR gate we have used the faster ten transistor implementation based on transmission gate XOR to match the delay with AND gates [4]. The completion detection following (4) is negated to obtain an active high completion signal (TERM). This requires a large fan-in n-input NOR gate. Therefore, an alternative more practical pseudo-nMOSratioed design is used. The resulting design is shown in Fig. 3(d). Using the pseudo-nMOS design, the completion unit avoids the high fan-in problem as all the connections are parallel. The pMOS transistor connected to *V*DD of this ratio-ed design acts as a load register, resulting in static current drain when some of the nMOS transistors are on simultaneously. In addition to the *Ci*s, the negative of SEL signal is also included for the TERM signal to ensure that the completion cannot be accidentally turned on during the initial selection phase of the actual inputs. It also prevents the pMOS pull up transistor from being always on. Hence, static current will only be flowing for the duration of the actual computation.

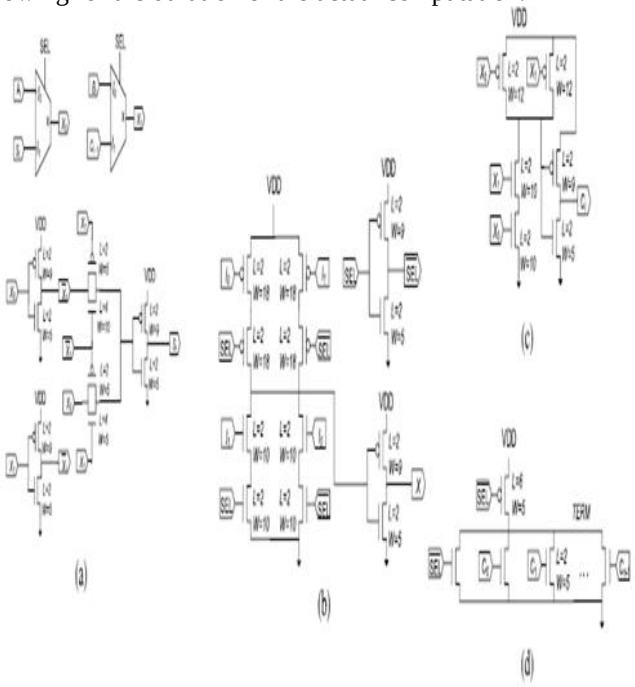

Fig.7. CMOS implementation of PASTA. (a) Single-bit sum module. (b) 2×1 MUX for the 1 bit adder. (c) Single-bit carry module. (d) Completion signal detection circuit.

IV. SIMULATION RESULTS

The corresponding simulation results of the PASTA adders are shown below. All the synthesis and simulation results are performed using Verilog HDL. The synthesis and simulation are performed on Xilinx ISE 14.4. The simulation results are shown below figures.

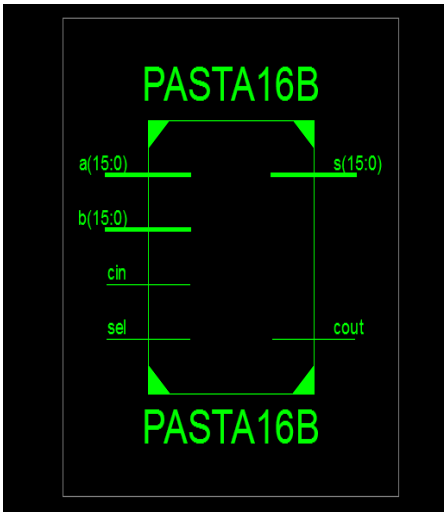

Fig.8. RTL schematic of Top-level 16 bit PASTA adders.

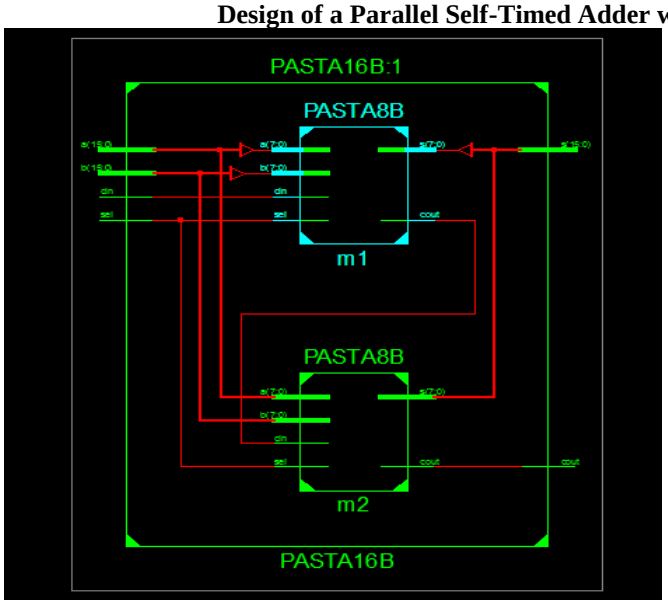

Fig9. RTL schematic of Internal block 16 bit PASTA adders.

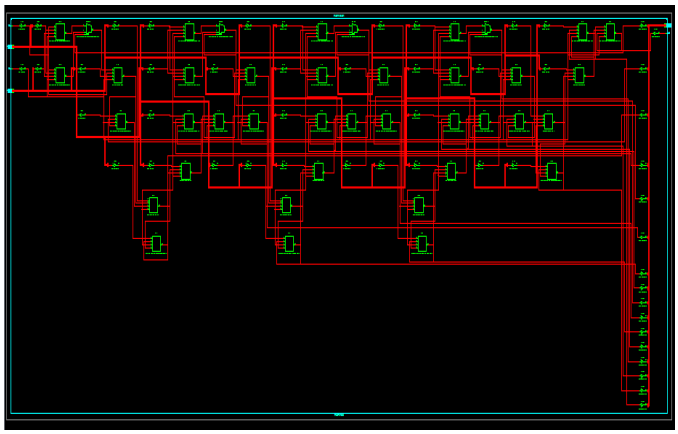

Fig10. Technology schematic of Top-level 16 bit PASTA adders.

Table 1. Design summary report of 16 bit PASTA adders

PASTA16B Project Status			
Project File:	PASTA_ISE.xise	**Parser Errors:**	No Errors
Module Name:	PASTA16B	**Implementation State:**	Synthesized
Target Device:	xc3s500e-4fg320	**•Errors:**	No Errors
Product Version:	ISE 14.4	**•Warnings:**	29 Warnings (29 new)
Design Goal:	Balanced	**•Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	20	4656	0%
Number of 4 input LUTs	36	9312	0%
Number of bonded IOBs	51	232	21%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Tue Aug 16 22:17:13 2016	0	29 Warnings (29 new)	0
Translation Report					
Map Report					

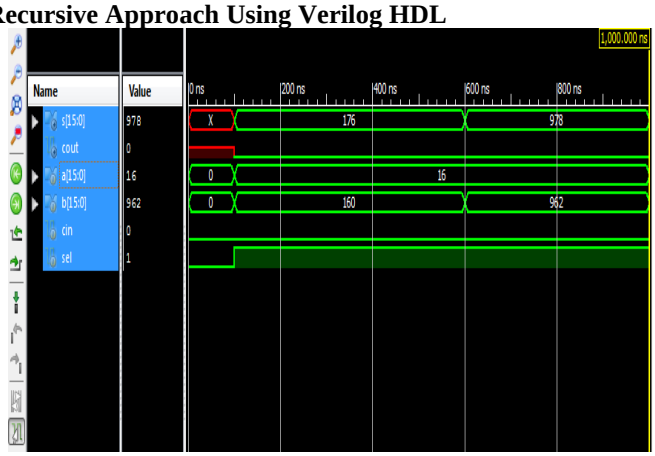

Fig12. Simulated output for 16 bit PASTA adders.

V. CONCLUSION

This brief presents an efficient implementation of a PASTA. Initially, the theoretical foundation for a single-rail wave-pipelined adder is established. Subsequently, the architectural designs are presented. The design achieves a very simple n-bit adder that is area and interconnection-wise equivalent to the simplest adder namely the RCA. Moreover, the circuit works in a parallel manner for independent carry chains, and thus achieves logarithmic average time performance over random input values. The completion detection unit for the proposed adder is also practical and efficient. Simulation results are used to verify the advantages of the proposed approach.

VI. REFERENCES

[1] D. Geer, "Is it time for clockless chips? [Asynchro-nous processorchips]," IEEE Comput., vol. 38, no. 3, pp. 18–19, Mar. 2005.

[2] J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design.Boston, MA, USA: Kluwer Academic, 2001.

[3] P. Choudhury, S. Sahoo, and M. Chakraborty, "Im-plementation of basicarithmetic operations using cel-lular automaton," in Proc. ICIT, 2008,pp. 79–80.

[4] M. Z. Rahman and L. Kleeman, "A delay matched approach forthe design of asynchronous sequential circuits," Dept. Comput.Syst.Technol., Univ. Malaya, Kuala Lumpur, Malaysia, Tech. Rep. 05042013,2013.

[5] M. D. Riedel, "Cyclic combinational circuits," Ph.D. dissertation, Dept.Comput. Sci., California Inst. Tech-nol., Pasadena, CA, USA, May 2004.

[6] R. F. Tinder, Asynchronous Sequential Machine Design and Analysis:A Comprehensive Development of the Design and Analysis ofClock-Independent State Machines and Systems. San Mateo, CA, USA:Morgan, 2009.

[7] W. Liu, C. T. Gray, D. Fan, and W. J. Farlow, "A 250-MHz wavepipelined adder in 2-µm CMOS," IEEE J. Sol-id-State Circuits, vol. 29,no. 9, pp. 1117–1128, Sep. 1994.

[8] F.-C. Cheng, S. H. Unger, and M. Theobald, "Self-timed carry-lookaheadadders," IEEE Trans. Comput., vol. 49, no. 7, pp. 659–672, Jul. 2000.

[9] S. Nowick, "Design of a low-latency asynchronous adder using speculativecompletion," IEE Proc. Comput. Digital Tech., vol. 143, no. 5,pp. 301–307, Sep. 1996.

[10] N. Weste and D. Harris, CMOS VLSI Design: A Circuits and SystemsPerspective. Reading, MA, USA: Addison- Wesley, 2005.
[11] C. Cornelius, S. Koppe, and D. Timmermann, "Dynamic circuit techniquesin deep submicron technolo-gies: Domino logic reconsidered," inProc. IEEE ICICDT, Feb. 2006, pp. 1–4. 2
[12] M. Anis, S. Member, M. Allam, and M. Elmasry, "Impact of technologyscaling on CMOS logic styles," IEEE Trans. Circuits Syst., AnalogDigital Signal Process., vol. 49, no. 8, pp. 577–588, Aug. 2002.