INTERNATIONAL JOURNAL OF EMINENT ENGINEERING TECHNOLOGIES

Volume 5, Issue 1 AUG 2017

DESIGN AND SIMULATION OF CARRY SAVE ADDER MULTIPLIER ANDCOMPARISION WITH PASTA MULTIPLIER

BATHINI VENKATA MADHURI
PG Scholar
Department of VLSI
Krishna Chaitanya Institute of Technology & Sciences, Devarajugattu, Peddaaraveedu, Prakasam, Andhra Pradesh, India.
madhurionemail@gmail.com

J.SRINIVAS
Assistant Professor
Department of VLSI
Krishna Chaitanya Institute of Technology & Sciences, Devarajugattu, Peddaaraveedu, Prakasam, Andhra Pradesh, India.
jsrinivaskits@gmail.com

Abstract-As the scale of integration keeps growing, a lot of and a lot of refined signal process systems are being enforced on a VLSI chip. These signal process applications not solely demand nice computation capability however conjointly consume extensive amounts of energy whereas performance and space stay to be two major style goals, power consumption has become a crucial concern in today's VLSI system style. Multiplication may be a elementary operation in most arithmetic computing systems. Multipliers have giant space, long latency and consume extensive power. Multiplication may be a basic operation that is gift in any a part of the digital computer, computing machine, computing device data processor, electronic computer, information process system particularly in signal processing systems completely different techniques are used for multiplication. A number of the techniques are CSA, CSD, Booth's, Grid, Lattice, combinatory, Sequential, Array, Vedic, Wallace-treeetc.

Keywords:Multiplier, VHDL, FPGA.

I. INTRODUCTION

Over the last twenty years, adaptive signal process has developed into a self-contained field that finds big selection of real-life applications similar to adaptive exploit, noise and echo cancellation, linear prophetic committal to writing, and adaptive beam-forming. Adaptive signal process algorithms square measure characterised by their algorithmic operations for realizing recursive self-designing/adaptation. To comprehend high-throughput VLSI implementation of adaptive signal process algorithms, architecture-level technique pipelining is usually used. Pipelined adaptive signal process systems square measure primarily subject to a trade-off between systems turnout and signal process performance, i.e., deeper pipelined adaptation electric circuit will noticehigher turnout, however the delayed feedback can incur larger performance degradation. It ought to be discovered that, for different algorithmic algorithmic rules similar to infinite impulse response (IIR) filtering and Viterbi algorithm, direct pipelining might merely ruin their practicality and applicable algorithm-level modification is needed for the employment of pipelining. A pipelined adaptive signal process algorithmic rule enforced exploitation the traditional synchronous pipeline generally encompasses a fastened pipeline depth that's determined within the style part to accommodate the best run-time turnout demand. though it's attainable to on-the-fly piece the pipeline depth of synchronous pipeline by by selection bypassing bound levels of registers, this is often terribly inflexible and can't notice fine-grain sleek configuration on the throughput/performance trade-offs. as an instance, think about associate 8-stage pipelined algorithmic adaptation loop during which the registers square measure nearly equally placed on the loop for increasing the turnout. If we have a tendency to bypass one level of registers to comprehend a 7-stage pipeline, the delay of the essential path might double and also the turnout can cut back nearly by [*fr1].

Self-timed pipeline works during a completely different means from its synchronous counterpart while not a typical and distinct notion of your time, self-timed pipeline depends on the handclasp between parts to perform the synchronization and communication. Every distinct knowledge propagating through a selftimed pipeline is conventionally known as a token. The pipeline depth of a self-timed pipeline merely equals the amount of tokens gift within the pipeline at a similar time. Hence, we will dynamically tack the pipeline

depth by dominant the amount of tokens gift within the pipeline. This property of self-timed pipeline has been exploited within the style of a mixed synchronous-asynchronous FIRfilter which will support variable latency (in terms of clock cycles)associate degreed power management of an embedded, single-issue processor. In pipelined adaptive signal process systems, the pipeline depth of the difference feedback loops is that the key to tune the inherent exchange between output and signal process performance. This directly motivates North American country to use self-timed pipeline for theimplementation of adaptive signal process systems to understand configurable throughput/performance exchange. this could be leveraged to enhance the system performance in several circumstances. parenthetically, for adaptive signal process systems with variable rate, we will dynamically alter the pipeline depth to the minimum allowable price in line with the present rate to understand the simplest signal process performance.

Though the fundamental plan of the on top of style approach is easy and intuitive, a way to implement it within the real systems involves the subsequent 3 important style issues:

1) What form of self-timed pipeline structure ought to be used? Clearly, to justify the usefulness of this style approach, the utilized self-timed pipeline should be ready to support a similar (or comparable) output as its synchronous counterpart once they have a similar pipeline depth. this implies that the algorithmic self-timed pipeline datapath ought to have a similar (or comparable) propagation delay as its synchronous counterpart. this is often a awfully strict demand since most self-timed pipeline style schemes involve further delay overhead for realizing self-timed handclasp and have the longer latency than their synchronous counterparts, though they'll support terribly fine-grain pipeline to understand high output. during this work, we have a tendency to propose to use the well-known Ted William's high-speed self-timed pipeline [4], [8] as a result of its zerodelay-overhead feature (i.e., no further handclasp delay is incurred once knowledge propagate through the pipeline). thence the zero-delay-overhead pipeline are able to do a similar latency performance as its synchronous counterpart.

2) a way to notice the self-timed knowledge flow synchronization within the algorithmic adaptation loop? In self-timed knowledge path, synchronization of parallel procedure threads depends on forks and joins, wherever fork refers to a stage with one input channel and multiple output channels and be a part of refers to a stage with multiple input channels and one output channel. The algorithmic adaptation loop of adaptive signal process algorithms contains several forks and joins. However, like several different self-timed pipeline designs, the zero-delay-overhead self-timed pipeline was at first planned for linear datapath (i.e., while not forks and joins). Therefore, it should be suitably changed to support forks and joins.

3) a way to notice run-time addition/removal of tokens so as to alter the pipeline depth? during a feed forward solely datapath, the pipeline depth are often promptly modified by adjusting the computer file rate. However, as we'll show later, it's not trivial to alter the pipeline depth in algorithmic adaptation loops. We have got to style some special circuit components which will be placed on the algorithmic adaptation loop to understand run-time addition/removal of tokens.

II. LITERATURE SURVEY

Power could be a drawback primarily once cooling could be a concern. the most power at any time, peak power, is usually used for power and ground wiring style, signal noise margin and responsibleness analysis. Energy per operation or task is a higher metric of the energy potency of a system, particularly in the domain of increasing battery time period. In digital CMOS style, the well-known power-delay product is often wont to assess the deserves of styles. Typically multiplication consists of 3 steps: Generation of partial merchandise or PPs (PPG), reduction of partial merchandise (PPR), and final carry-propagate addition (CPA). Totally different multiplication algorithms vary within the approaches of PPG, PPR, and CA. For PPG, radix-2 digit-vector multiplication is that the simplest kind as a result of the digit-vector multiplication is made by a set of AND gates. The amount of PPs and consequently reduce the area/delay of PP reduction, one quantity is sometimes recoded into high-radix digit sets. The foremost standard one is that the radix-4 digit set. For PPR, 2 alternatives exist: reduction by rows, performed by associate array of adders, and reduction by columns, performed by associate array of counters. In reduction by rows, there square measure 2 extreme classes: linear array and tree array. Linear array has the delay of O (n) whereas each tree array and column reduction have the delay of O (log n), wherever n is that the range of PPs. The ultimate controller needs a quick adder theme as a result of it's on the essential path. Some low-level techniques that has been studied for multipliers embrace victimization voltage scaling, layout improvement, junction transistor rearrangement and size, victimization pass-transistor logic and swing restricted logic, signal polarity improvement, delay equalization and input synchronization. However, these techniques have solely achieved moderate improvement on power consumption in multipliers with abundant style effort or tidy area/delay overhead. The issue of low-power multiplier factor

style lies in 3 aspects. Multiplication is essentially a shift add operation.

As a basic mathematical operation, multiplication hasseveral algorithm-leveland bit-level computation options within which it differs from random logic. These options haven't been thought of well in low-level power improvement. It's conjointly troublesome to contemplate computer file characteristics at low levels.Therefore, it's fascinating to develop algorithmic program and design level power improvement techniques.

Array Multiplier

Array number is renowned thanks to its regular structure. The number circuit relies on add and shift rule. Every partial product is generated by the multiplication of the number with one number bit. The partial product are shifted per their bit orders then additional. The addition will be performed with traditional carry save adder.

Advantages

First advantage of the array multiplier is that it has a regular structure. Since it is regular, it is easy to layout and has a small size. A second advantage of the array multiplier is its ease of design for a pipelined architecture.

III. DESIGN OF PASTA

In this section, the design and theory behind alimentary paste is conferred. The adder 1st accepts 2 input operands to perform 0.5 additions for every bit. Later on, it iterates exploitation earlier generated carry and sums to perform half-additions repeatedly till all carry bits are consumed and settled at zero level.

A. Architecture of PASTA

Let an-1,an-2. . . a0 and bn-1,bn-2. . . b0 be two n-bit binary numbers with sum and carry denoted by Sn-1, Sn-2. . . S0 and cn,cn-1. . . c0 where 0th bit represents the least significant bit.The two input multiplexer has selection input that corresponds to the request handshake signal and will be a single zero to one transition denoted by SEL. It will first select the actual operand during SEL = 0 and will switch to feedback paths for repeated iterations using SEL =1. The HAs feedback path enables the continuous several iterations until all the carry signals will be assumed to have zero values.

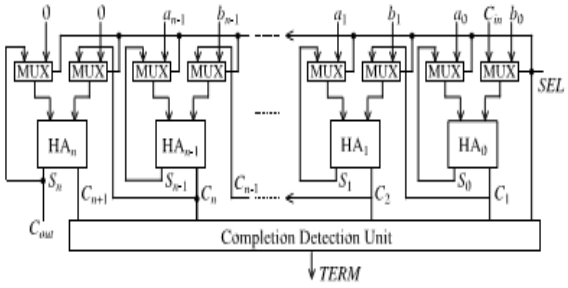


Fig.1. General block diagram of PASTA

B. State Diagrams

In Fig.2, two state diagrams area unit drawn for the initial section and therefore the unvarying section of the planned design. Every state is diagrammatic by (Ci+1Si) pairwhereCi+1, Si represent perform and add values, severally, from the ith bit adder block. Throughout the initial section, the circuit just works as a combinatory hour angle in operation in elementary mode. It's apparent that because of the employment of HAs rather than FAs, state (11) cannot appear.

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

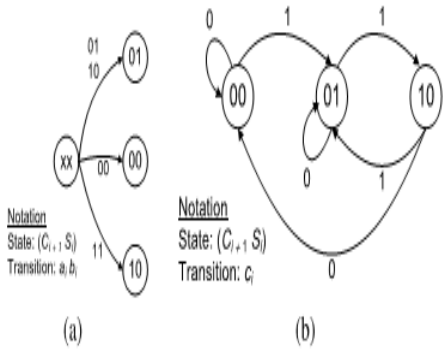


Fig.3. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase

C. Recursive Formula for Binary Addition

Let Cj i+1 denote the carry and Sji denote sum respectively, for ith bit at the jth iteration. The initial condition (j=0) for addition is formulated as follows

$$S_i^0 = a_i \oplus b_i$$
$$C_{i+1}^0 = a_i b_i. \quad (1)$$

Thejth iteration for the recursive addition is formulated by

$$S_i^j = S_i^{j-1} \oplus C_i^{j-1}, \quad 0 \le i < n \quad (2)$$
$$C_{i+1}^j = S_i^{j-1} C_i^{j-1}, \quad 0 \le i \le n. \quad (3)$$

The jth iteration for the recursive addition is formulated by

$$C_n^k + C_{n-1}^k + \cdots + C_1^k = 0, \quad 0 \le k \le n. \quad (4)$$

Now, the correctness of the formulation of recursive approach is primarily 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 terminating condition on required number of iterations for completing the addition.

Basis: Consider the operand a and b choices for which no carry propagation is required, i.e., C0i = 0 for ∀i,i∈[0..n]. The proposed recursive formulation will produce the correct result inparallel by single bit computation time and terminates instantly as condition (4) is met.

Induction: Assume that at kth iteration Cki+1≠0 for some ith bit. Let l be such a bit for which Ckl+1 =1. First we show that it will be killed in the (k + 1) iteration and nextwe will show that it will be successfully transmitted to next higher bit in the (k+1)th iteration. As shown in the state diagram, the kth iteration of ith bit state (Ckl+1,Skl) and (i+1)thbit state Ckl+2,Skl+1) could be in any of (0,0), (0,1),or(1,0) states. As Ckl+1 =1, it implies that Skl=0. Hence, from (3),Ck+1l+1 =0 for any input condition between 0to l bits.

We now consider the next higher bit (i +1)th bit state (Ckl+2,Skl+1) for kth iteration. By observation 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 forms the kth iteration will correctly produce output of(0,1) following (2) and (3).For (1, 0) form, the carry is supposed to propagate through this bit level as the sum value is 1.

Thus, by fulfilling the terminating condition all the single-bit adders will kill or propagate the carries successfully until all carries are zero. The mathematical form presented above under the condition is valid that for all the bit levels the iterations progress synchronously and for a specific iteration the required input and outputs will also be in synchrony with the progress of one iteration.

IV DESIGN OF CSA MULTIPLIER

Generally multiplication consists of 3 steps: generation of partial merchandise or PPs (PPG), reduction of partial merchandise (PPR), and final carry-propagate addition (CPA). Totally different multiplication algorithms vary within the approaches of PPG, PPR, and CA.

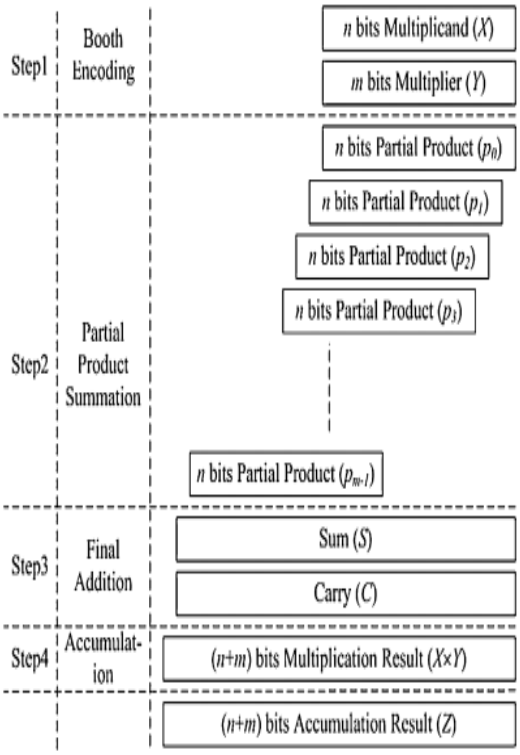


Fig 3.Basic arithmetic steps of multiplication and accumulation.

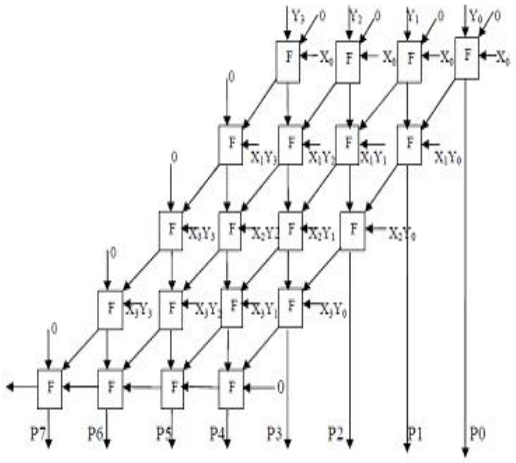


Fig.4.Multiplier with Carry saves Adder Architecture

In the Carry Save Addition methodology, the primary row are either Half-Adders or Full-Adders. If the primary row of the partial product is enforced with Full-Adders, Cin are thought of "0". Then the carries of every Full- Adder is diagonally forwarded to consequent row of the adder. The ensuing multiplier factor is claimed to be Carry Save multiplier factor, as a result of the carry bits aren't straightaway another, however rather are saved for consequent stage. Within the style if the complete adders have 2 input file the third input is taken into account as zero. Within the finish, carries and sums are integrated in an exceedingly quick carry-propagate (e.g. ripple carry or carry look ahead) adder stage.

V. SIMULATION RESULTS

CSA Multiplier:

Design Summary:

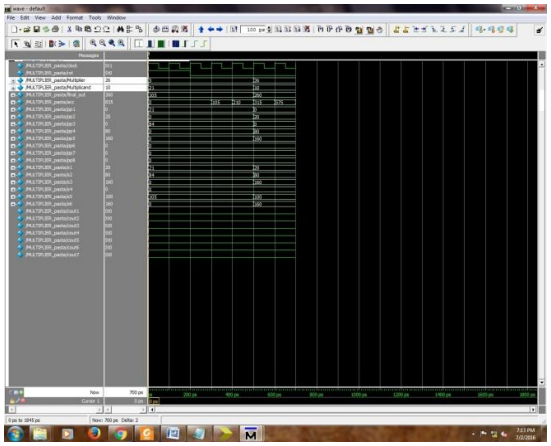
Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	42	4656	0%
Number of Slice Flip Flops	16	9312	0%
Number of 4 input LUTs	83	9312	0%
Number of bonded IOBs	50	232	21%
Number of GCLKs	1	24	4%

Timing Summary:

Speed Grade: -5

Minimum period: 3.750ns (Maximum Frequency: 266.670MHz)
Minimum input arrival time before clock: 17.021ns
Maximum output required time after clock: 4.063ns
Maximum combinational path delay:17.021ns

PASTA Multiplier:



Design Summary:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	95	4656	2%
Number of Slice Flip Flops	32	9312	0%
Number of 4 input LUTs	170	9312	1%
Number of bonded IOBs	50	232	21%
Number of GCLKs	1	24	4%

Timing Summary:

Speed Grade: -5

Minimum period: 3.667ns (Maximum Frequency: 272.706MHz
Minimum input arrival time before clock: 11.344ns
Maximum output required time after clock: 4.063ns
Maximum combinational path delay: 12.430ns

VI. CONCLUSION

These days speed of the multiplier has become an asset or constraint due to the importance of multiplier circuit in a wide variety of microelectronic systems. In this paper we analyzed different multiplier techniques taking speed as the main criteria. Carry save adder is proved to be more efficient in terms of speed compared to conventional multiplication techniques generated the output in 2.06 sec, whereas the booth multiplier generated the output in 3.09sec while the shift &add multiplier produce d the output in 2.31 sec. However the array multiplier generated the output in 2.75sec and modified booth multiplier in around 3.08 sec.The carry save adder on the other hand consumes less hardware than other multiplication techniques.

REFERENCES

[1] GarimaTiwari "Analysis, Verification and FPGA Implementation of Low Power Multiplier".
[2]KripaMathew,S.AshaLatha,T.Ravi, E.Logashanmugam "design and analysis of an Array Multiplier using an Area Efficient full adder cell in 32 nm CMOS Technology".
[3]ChakibAlaoui "Design and Simulation of a Modified Architecture of Carrysave Adder".
[4]DeepaliChandel,GaganKumawat, PranayLahoty,VidhiVartChandrodaya,ShailendraSharma.International Journal of Emerging Technology and Advanced Engineering Volume 3, Issue 3, March 2013"Booth Multiplier: Ease of multiplication".
[5] International Journal of Engineering Science InventionShaik.KalishaBaba,D.Rajaramesh "Design and Implementation of Advanced Modified Booth Encoding Multiplier".
[6] G.W. Bewick, "Fast Multiplication: Algorithms and Implementation."Ph.D. dissertation, Stanford University, Feb. 1994
[7] Shiann-RongKuang, Jiun-Ping Wang, and Cang-Yuan Guo, "Modified Booth multipliers with a Regular Partial Product Array," IEEE Transactions on circuits and systems-II, vol 56, No 5, May 2009.
[8] 8.M. Zamin Ali Khan1, Hussain Saleem2, Shiraz Afzal3 and Jawed Naseem4, — An Efficient 16-Bit Multiplier based on Booth Algorithm, international Journal of Advancements in Research & Technology, Volume 1, Issue 6, November-2012 ISSN 2278-7763
[9]Dr.RaviShankaMishra,Prof.PuranGour,BrajBihari Soni, —Design and Implements of Booth and Robertson, multipliers algorithm on FPGA.‖ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622.
[10] F.C Cheng, S. H. Unger, "Self-Timed Carry-Look Ahead Adders", IEEE Transactions on Computers, Vol. 49, No. 7, July 2000.