A LOW POWER HIGH SPEED CONFIGURABLE ADDERS FOR APPROXIMATE COMPUTING

A. Vyshnavi
P.G Student
Jyothishmathi institute of technology and science
vaishnavi.annamaraju@gmail.com

Dr. G. Karthick
Associate Professor
Jyothishmathi institute of technology and science
karthick.sgs@gmail.com

ABSTRACT

Carry Select Adder is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. From the structure of the CSLA, it is clear that there is scope for reducing the area and power consumption in the CSLA. This work uses a simple and efficient gate-level modification to significantly reduce the area and power of the CSLA. Based on this modification 8-, 16-, 32-, and 64-b square-root CSLA architecture have been developed and compared with the regular SQRT CSLA architecture. The proposed design has reduced area and power as compared with the regular SQRT CSLA with only a slight increase in the delay.

Design of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position.

Keywords: CLSA,CSA, Haff Adder, Full Addder

1.1 INTRODUCTION

The design of high-speed and low-power VLSI architectures needs efficient arithmetic processing units, which are optimized for the performance parameters, namely, speed and power consumption. Adders are the key components in general purpose microprocessors and digital signal processors. They also find use in many other functions such as subtraction, multiplication and division. As a result, it is very pertinent that its performance augers well for their speed performance. Furthermore, for the applications such as the RISC processor design, where Single cycle execution of instructions is the key measure of performance of the circuits, use of an efficient adder circuit becomes necessary, to realize efficient system performance.

Additionally, the area is an essential factor which is to be taken into account in the design of fast adders.

Digital Adders are the core block of DSP processors. The final carry propagation adder (CPA) structure of many adders constitutes high carry propagation delay and this delay reduces the overall performance of the DSP processor. This project proposes a simple and efficient approach to reduce the maximum delay of carry propagation in the final stage. Based on this approach a 16, 32 and 64-bit adder architecture has been developed and compared with conventional fast adder architectures.

The basic principle is to relax the energy requirement by allowing possibly incorrect computation results. For devices with probabilistic components, energy should be assigned to each component wisely, in order to achieve a good trade-off between energy consumption and correctness of the outputs. Recently, a few schemes have been proposed for energy assignment of ripple-carry adders, which are often based on intuitive arguments

1.2 literature survey

The single electron tunneling (SET) technology-based computation of basic addition related arithmetic functions, e.g., addition and multiplication, via a novel computation paradigm, which we refer to as electron counting arithmetic, that is based on controlling the transport of discrete quantities of electrons within the SET circuit. First, assuming that the number of controllable electrons within the system is unrestricted, we

prove that the addition of two n-bit operands can be computed with a depth-2 network composed out of 3n+1 circuit elements and that the multiplication of two n-bit operands can be computed with a depth-3 network composed out of 4n-1 circuit elements. Second, assuming that the number of controllable electrons cannot be higher than a given constant r determined by practical limitations, we prove that the addition of two n-bit operands can be computed with a depth-(n/r+3) network composed out of 3n+1+n/r circuit elements.

CMOS-based examples for addition starting from the device level and moving up to the gate, the circuit, and the block level. Our analysis, backed by simulation results, on comparing parallel and serial addition shows that serial adders are more reliable while also dissipating less. Their reliability can be improved by using reliability-enhanced gates and/or other redundancy techniques (like e.g., multiplexing). Additionally, the architectural technique of short-circuiting the outputs (of several redundant devices/gates/blocks) exhibits "vanishing" voting and an inherent fault detection mechanism, as both transient and permanent faults could be detected based on current changes. The choice of CMOS is due to the broad design base available (but the ideas can be applied to other technologies), while addition was chosen due to its very solid background (both theoretical and practical). The design approach will constantly be geared towards enhancing reliability as much as possible at all the levels. Theory and simulations will support the claim that a serial adder is a very serious candidate for highly reliable and low power operations.

Approximate computing leverages the intrinsic resilience of applications to inexactness in their computations, to achieve a desirable trade-off between efficiency (performance or energy) and acceptable quality of results. To broaden the applicability of approximate computing, we propose quality programmable processors, in which the notion of quality is explicitly codified in the HW/SW interface, i.e., the instruction set. The ISA of a quality programmable processor contains instructions associated with quality fields to specify the accuracy level that must be met during their execution. We show that this ability to control the accuracy of instruction execution greatly enhances the scope of approximate computing, allowing it to be applied to larger parts of programs. The micro-architecture of a quality programmable processor contains hardware mechanisms that translate the instruction-level quality specifications into energy savings. Additionally, it may expose the actual error incurred during the execution of each instruction (which may be less than the specified limit) back to software. As a first embodiment of quality programmable processors, we present the design of Quora, an energy efficient, quality programmable vector processor.

Approximation can increase performance or reduce power consumption with a simplified or inaccurate circuit in application contexts where strict requirements are relaxed. For applications related to human senses, approximate arithmetic can be used to generate sufficient results rather than absolutely accurate results. Approximate design exploits a tradeoff of accuracy in computation versus performance and power. However, required accuracy varies according to applications, and 100% accurate results are still required in some situations. In this paper, we propose an accuracy-configurable approximate (ACA) adder for which the accuracy of results is configurable during runtime. Because of its configurability, the ACA adder can adaptively operate in both approximate (inaccurate) mode and accurate mode. The proposed adder can achieve significant throughput improvement and total power reduction over conventional adder designs.

Approximate circuit designs allow us to tradeoff computation quality (e.g., accuracy) and computational effort (e.g., energy), by exploiting the inherent error-resilience of many applications. As the computation quality requirement of an application generally varies at runtime, it is preferable to be able to reconfigure approximate circuits to satisfy such needs and save unnecessary computational effort. In this paper, we present a reconfiguration-oriented design methodology for approximate circuits, and propose a reconfigurable approximate adder design that degrades computation quality gracefully. The proposed design methodology enables us to achieve better qualityeffort tradeoff when compared to existing techniques, as demonstrated in the application of DCT computing.

Low power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, human beings can gather useful information

from slightly erroneous outputs. Therefore, we do not need to produce exactly correct numerical outputs. Previous research in this context exploits error resiliency primarily through voltage overscaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors. In this paper, we propose logic complexity reduction at the transistor level as an alternative approach to take advantage of the relaxation of numerical accuracy. We demonstrate this concept by proposing various imprecise or approximate full adder cells with reduced complexity at the transistor level, and utilize them to design approximate multi-bit adders. In addition to the inherent reduction in switched capacitance, our techniques result in significantly shorter critical paths, enabling voltage scaling. We design architectures for video and image compression algorithms using the proposed approximate arithmetic units and evaluate them to demonstrate the efficacy of our approach.

1.3 METHODOLOGY

For high-speed, low power and area efficient addition and multiplication has always been a fundamental requirement of high-performance processors and systems. The major speed limitation of adders arises from the huge carry propagation delay encountered in the conventional adder circuits, such as ripple carry adder and carry save adder.

The main advantage of CSA is its reduced propagation delay characteristics. This is realized by the use of parallel stages that results from multiple pairs of ripple carry adder. The ripple carry adders generate their interim sum and carry for the CSA structure by considering the carry input to be 0 and 1 respectively. The final sum and carry output is chosen by the use of multiplexers. The carry-out bit of the preceding block of the adder acts as the select signal to the multiplexer.

The Code for Carry Select Adder has been developed and simulated in Verilog and has been realized for 16, 32 and 64 bits.

1.4 SIGNIFICANCE OF THE WORK

Carry Select Adder (CSA) is known to be the fastest adder among the conventional adder structures. It is used in many data processing units for realizing faster arithmetic operations. Adders are the key components in general purpose microprocessors and digital signal processors. They also find use in many other functions such as subtraction, multiplication and division. As a result, it is very pertinent that its performance augers well for their speed performance

The multipliers use adders in their final stage and the speed performance of multipliers is determined by the type of adders they actually employ in the addition process. CSA is preferred for the addition operation of the most significant bits (MSBs) of multiplier. This is due to the fact that the addition can be accomplished while awaiting the arrival of the carry signal from the least significant bit region. The CSA thus combines the advantage of reduced delay and area efficient architecture when compared to conditional sum adder.

In this project, we employ a novel fast incrementer circuit in place of adders to increment the interim sum when the carry in is obtained as logic 1. This is proved to be more advantageous than the conventional CSA method.

2.1 CONVENTIONAL ADDER CIRCUITS

2.1.1 ADDER

In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar.

Although adders can be constructed for many numerical representations, such as binary-coded decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–sub tractor. Other signed number representations require a more complex adder.

2.1.2 HALF ADDER

The half adder adds two one-bit binary numbers *A* and *B*. It has two outputs, *S* and *C* (the value theoretically carried on to the next addition); the final sum is $2C + S$. The simplest half-adder design, pictured on the right, incorporates an XOR gate for *S*

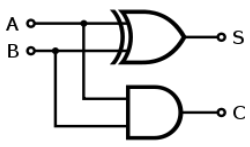


Fig 2.1: Half Adder Logic Diagram

and an AND gate for *C*. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder.

2.1.3 FULL ADDER

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as *A*, *B*, and C_{in}; *A* and *B* are the operands, and C_{in} is a bit carried in from the next less significant stage.

The full-adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. binary numbers. The circuit produces a two-bit output sum typically represented by the signals C_{out} and *S*, where.

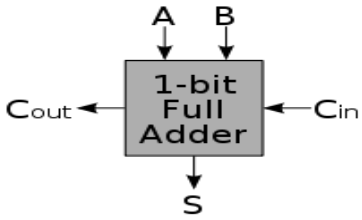


Fig 2.2: Schematic Symbol for a 1-Bit Full Adder with C_{in} and C_{out} Drawn On Sides of Block to Emphasize Their Use in a Multi-Bit Adder

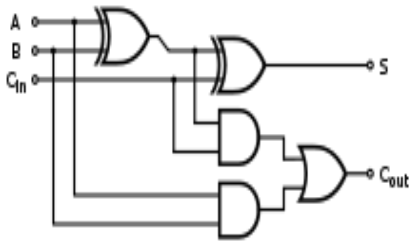


Fig 2.3: Full-Adder Logic Diagram

A full adder can be implemented in many different ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with

In this implementation, the final OR gate before the carry-out output may be replaced by an XOR gate without altering the resulting logic. Using only two types of gates is convenient if the circuit is being implemented using simple IC chips which contain only one gate type per chip. In this light, C_{out} can be implemented as

.

A full adder can be constructed from two half adders by connecting *A* and *B* to the input of one half adder, connecting the sum from that to an input to the second adder, connecting C_{in} to the other input and OR the two carry outputs. Equivalently, *S* could be made the three-bit XOR of *A*, *B*, and C_{in}, and C_{out} could be made the three-bit majority function of *A*, *B*, and C_{in}.

2.1.4 CARRY LOOK AHEAD ADDER

A Carry-look Ahead Adder (CLA) is a type of adder used in digital logic. A Carry-look Ahead Adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, *ripple carry adder* for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits (see adder for detail on ripple carry adders). The Carry-look Ahead Adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits.

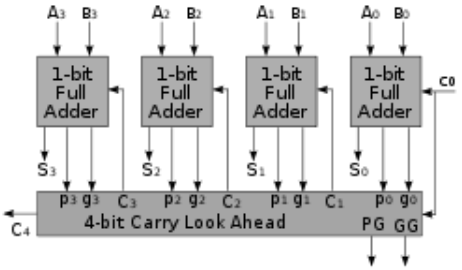


Fig 2.4: 4-Bit Adder with Carry Look Ahead

In ripple carry adders, the carry propagation time is the major speed limiting factor. Most other arithmetic operations, e.g. multiplication and division are implemented using several add/subtract steps. Thus, improving the speed of addition will improve the speed of all other arithmetic operations.

Accordingly, reducing the carry propagation delay of adders is of great importance. Different logic design approaches have been employed to overcome the carry propagation problem.

One widely used approach employs the principle of carry look-ahead solves this problem by calculating the carry signals in advance, based on the input signals.

A ripple-carry adder works in the same way as pencil-and-paper methods of addition. Starting at the rightmost (least significant) digit position, the two

corresponding digits are added and a result obtained. It is also possible that there may be a carry out of this digit position (for example, in pencil-and-paper methods, "9+5=4, carry 1"). Accordingly all digit positions other than the rightmost need to take into account the possibility of having to add an extra 1, from a carry that has come in from the next position to the right.

This means that no digit position can have an absolutely final value until it has been established whether or not a carry is coming in from the right. Moreover, if the sum without a carry is 9 (in pencil-and-paper methods) or 1 (in binary arithmetic), it is not even possible to tell whether or not a given digit position is going to pass on a carry to the position on its left. At worst, when a whole sequence of sums comes to ...99999999... (in decimal) or ...11111111... (in binary), nothing can be deduced at all until the value of the carry coming in from the right is known, and that carry is then propagated to the left, one step at a time, as each digit position evaluated "9+1=0, carry 1" or "1+1=0, carry 1". It is the "rippling" of the carry from right to left that gives a ripple-carry adder its name, and its slowness. When adding 32-bit integers, for instance, allowance has to be made for the possibility that a carry could have to ripple through every one of the 32 one-bit adders.

Carry look ahead depends on two things:

1. Calculating, for each digit position, whether that position is going to propagate a carry if one comes in from the right.
2. Combining these calculated values to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

Suppose that groups of 4 digits are chosen. Then the sequence of events goes something like this:

1. All 1-bit adders calculate their results. Simultaneously, the look ahead units perform their calculations.
2. Suppose that a carry arises in a particular group. Within at most 3 gate delays, that carry will emerge at the left-hand end of the group and start propagating through the group to its left.
3. If that carry is going to propagate all the way through the next group, the look ahead unit will already have deduced this. Accordingly, before the carry emerges from the nex group the look ahead unit is immediately (within 1 gate delay) able to tell the *next* group to the left that it is going to receive a carry - and, at the same time, to tell the next look ahead unit to the left that a carry is on its way.

The net effect is that the carries start by propagating slowly through each 4-bit group, just as in a ripple-carry system, but then move 4 times as fast, leaping from one look ahead carry unit to the next. Finally, within each group that receives a carry, the carry propagates slowly within the digits in that group.

3.1 CONVENTIONAL CARRY SELECT ADDER

A carry save adder is used to compute the sum of three or more bits in binary format. It is widely used in the final stages of fast multipliers for summing the partial products to give out the final value. The advantage of carry save adder is that the sum is computed faster than the conventional RCA. The carry save adder is better than the conventional carry select adder, in terms of area and power consumption while slower than carry select adder. In this project, we have compared the conventional carry save adder with our proposed carry select adder. The results prove that our adder is advantageous than the conventional adder in terms of speed, area and low power consumption. Hence, this makes it a good choice to replace the carry save adder structure, in the final stages of fast multipliers. This improves the speed of operation of the high performance VLSI circuits.

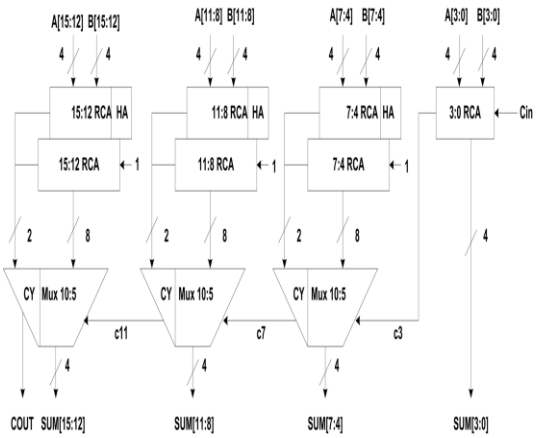


Fig 3.1: Conventional 16-Bit Carry Select Adder

Fig 3.1 shows the conventional 16-bit carry select adder with two RCA blocks each for every 4-bit groups. The first 4-bit RCA block adds the 4-bits assuming a 0 carry bit. On the other hand, the second RCA adds the 4-bits with logic 1 carry bit. The final sum is obtained based on the carry bit from the previous stage. Hence, it has a replicated RCA block for every 4-bit group.

Design of high speed data path logic systems are one of the most substantial research area in VLSI system design. High-speed addition and multiplication has always been a fundamental requirement of high-performance processors and systems. The major speed limitation in any adder is in the production of carries and many authors have considered the addition problem.

3.2 CONVENTIONAL INCREMENTER CIRCUITS

Figure 3.2shows the internal logic circuit schematic of an *incrementer*, based on the conventional 4-bit ripple carry adder. The half adders, denoted by HA, are used for constructing the incrementer structure. Fig3.3 depicts the internal logic diagram of a 4-bit ripple carry *decrementer*. In these designs, the critical delay for the incrementing / decrementing operation is determined by the delay incurred in the carry bit propagation time, through the chain of AND gates. As the size of the chain grows, this delay also increases, making the delay of the most significant bit (MSB), the largest one in the process. In the proposed design, the carry is calculated using parallel AND chain thereby reducing the delay incurred by MSB.

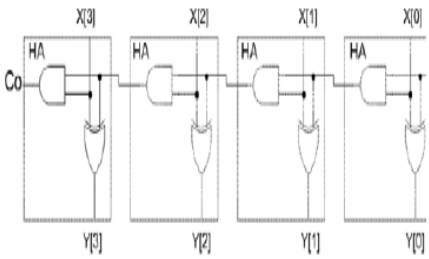


Fig 3.2: 16 Ripple Carry Incrementer

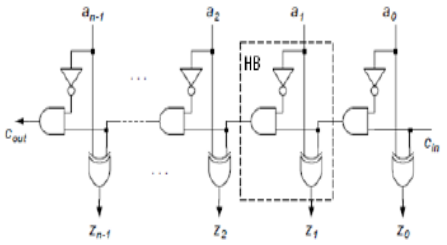


Fig3.3: Ripple Carry Decrementer

The basic idea of the proposed work is using n-bit binary to excess-1 code converters (BEC) to improve the speed of addition. The detailed structure and function of BEC is discussed in section 4.2. This logic can be implemented with any type of adder to further improve the speed. The proposed 16, 32 and 64-bit adders are compared in this project with the conventional fast adders such as carry save adder (CSA) and carry look ahead adder (CLA). This project has realized the improved performance of the CSA with BEC logic through custom design and layout. The final stage CPA constitutes a dominant component of the delay in the parallel multiplier.

Signals from the multiplier partial products summation tree do not arrive at the final CPA at the same time. This is due to the fact that the number of partial-product bits is larger in the middle of the multiplier tree. Due to un-even arrival time of the input signals to the final CPA, the selection of the ASIC Implementation of Modified Faster Carry Save Adder 54 final adder is an important work in parallel multipliers.

Therefore, decrease in carry propagation delay will result in major enhancement of the speed of the adder and multiplier.

The problem is again, it requires more circuitry because it requires two full adders at each stage of three bits addition. That is replaced by one RCA and one add-one circuit. There again the same problem that is eliminated by this proposed system CSA using BEC.

The basic idea of this work is to use Binary to Excess-1 Convertor (BEC) instead of RCA with Cin=1 in the regular CSA to achieve lower area and power consumption. The main advantage of this BEC logic comes from the lesser number of logic gates than the Full adder (FA) structure. The details of the BEC logic are discussed below. This brief is

structured as follows. Section 4.1deals with the delay and area evaluation methodology of the basic adder blocks.

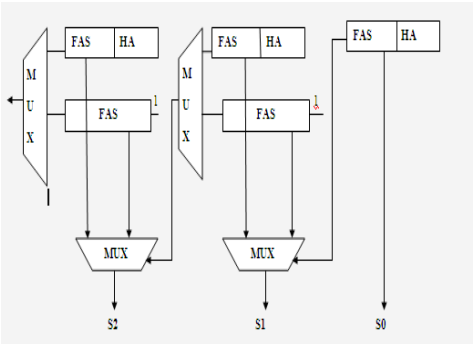


Fig4.1: Conventional CSA using Dual RCAs

4.1 DELAY AND AREA EVALUATION METHODOLOGY OF THE BASIC ADDER BLOCKS

The AND,OR and Inverter(AOI) implementation of an XOR gate is shown below in Fig.4.2.The gates between the dotted lines are performing the operations in parallel and the numeric representation of each gate indicates the delay contributed by that gate. The delay and area evaluation methodology considers all gates to be made up of AND, OR and Inverter, each having delay equal to 1 unit and area equal to 1 unit.

We then add up the number of gates in the longest path of a logic block that contributes to the maximum delay. The area evaluation is done by counting the total number of AOI gates required for each logic block. Based on this approach, the CSA adder blocks of 2:1mux, Half Adder (HA), and FA are evaluated and listed in Table-III.

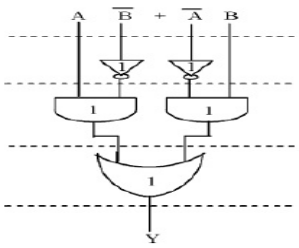


Fig4.2: Delay and Area Evaluation of

an XOR Gate

It can be seen that the carry out (C0) of the block is calculated in parallel along with B3 by using a parallel chain of AND gates, whereas a series pattern of carry propagation is used in RCA structure, which reduces the delay of incrementing in CSA when compared with the conventional RCA. Figure 3 shows the proposed 16-bit carry select adder, which equally divides the word size of the adder into blocks of 4-bit each.

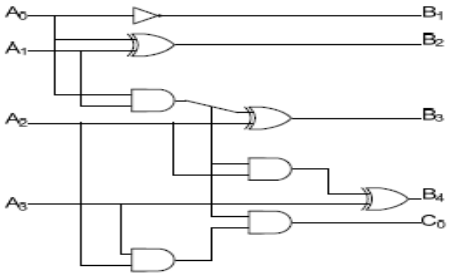


Fig 4.5: 4 Bit Basic Block used in Proposed Carry Select Adder

It can be seen that the carry out (C0) of the block is calculated in parallel along with B3 by using a parallel chain of AND gates, whereas a series pattern of carry propagation is used in RCA structure, which reduces the delay of incrementing in CSA when compared with the conventional RCA. Figure 4.6 shows the proposed 16-bit carry select adder, which equally divides the word size of the adder into blocks of 4-bit each.

The least significant 4-bits are added using conventional RCA, while other blocks are added in parallel along with the given incrementer

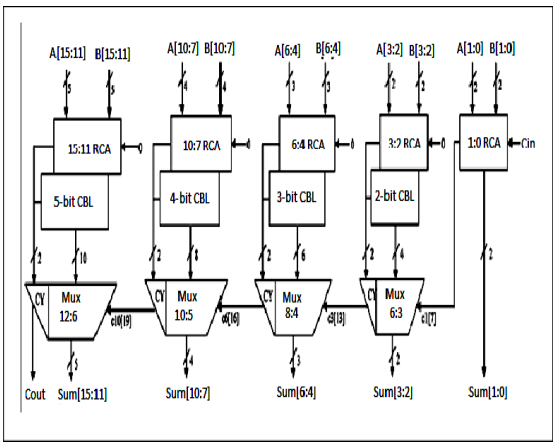


Fig 4.6: Proposed 16-bit Carry Select Adder

Once all the interim sums and carries are calculated, the final sums are computed using multiplexers having minimal delay. The multiplexer block receives the two sets of 5-bit input (four sum bits and one carry bit each) and selects the final sum based on the select input from the previous stage. Use of the basic unit with the 10-to-5 multiplexer thus achieves fast incrementing action with reduced device count. Thus, the proposed CSA excels the

conventional CSA circuit in terms of speed by reducing the carry propagation latency.

The invention of the transistor by William B. Shockley, Walter H. Brattain and John Bardeen of Bell Telephone Laboratories was followed by the development of the integrated circuit. The very first IC emerged at the beginning of 1960 and since that time there have already been four generations of ICs: SSI (Small scale integration), MSI (Medium scale integration), LSI (Large scale integration), and VLSI (Very large scale integration). Now we are beginning to see the emergence of the fifth generation, ULSI (Ultra large scale integration). Further miniaturization is still to come and more revolutionary advances in the application of this technology must inevitably occur.

Two approaches to VLSI IC design are philosophically identifiable. In the first, called a BOTTOM-UP design illustrated in figure3.1.1, the designer starts at the transistor or at gate level and designs sub –circuits of increasing complexity, which are then interconnected to realize the required functionality. In the second, termed the TOP-DOWN design methodology illustrated in figure 3.1.1, the designer repeatedly decomposes the system-level specifications into groups and subgroups of similar tasks. The lowest level tasks are ultimately implemented in silicon, either with standard circuits that have been previously designed and tested (often termed standard cells) or with low- level circuits designed to meet the required specifications.

Several trends in the production of VLSI circuits are readily identifiable. The most visible is the continual shrinking of the minimum geometrical feature size. The rate at which the minimum features size decreases is slowing. This slowing is partially attributable to inherent physical limitations in the photolithographic process and rapidly increasing costs associated with very fine resolution processing equipment.

5. SIGNIFICANCE OF HDL IN VLSI DESIGN

Programming languages such as FORTRAN, PASCAL and C were being used to describe computer programs that were sequential in nature. In digital design field Hardware Description Languages (HDLs) came into existence. HDLs allowed the designers to model the concurrency of process found in hardware elements. Hardware description languages such as Verilog HDL and VHDL became popular. Both Verilog and VHDL simulators to simulate large digital circuits quickly gained acceptance.

Even though HDLs were popular for logic verification, designers had to translate the HDL-based design into a schematic circuit with interconnection between gates. The advent of logic synthesis in the design, digital circuits could be described at Registered Transfer Level (RTL) by use of a HDL. Thus the designer had to specify how the data flows between registers and how the design processes the data. Logic synthesis tools from the RTL description automatically extracted the details of gates and their interconnections to implement the circuit.

Designers no longer had to manually place gates to build digital circuits. They could describe complex circuits at an abstract level in terms of functionality and data flow by designing those circuits in HDLs. Logic synthesis tools would implement the specified functionality in terms of gates and gate interconnections. HDLs also began to be used for system level design. HDLs were used for simulation of system boards, interconnects buses, FPGAs and PLAs.

6. RESULTS:

Simulation is the process of verifying the functional characteristics of models at any level of abstraction. We use simulators to simulate the Hardware models. To test if the RTL code meets the functional requirements of the specification, we must see if all the RTL blocks are functionally correct. To achieve this we need to write a test bench, which generates clk, reset and the required test vectors. A sample test bench for a counter is shown below. Normally we spend 60-70% of time in design verification.

In this project, I have simulated the results for 32 each with Cin=0 and Cin=1 case respectively and observed the waveforms.

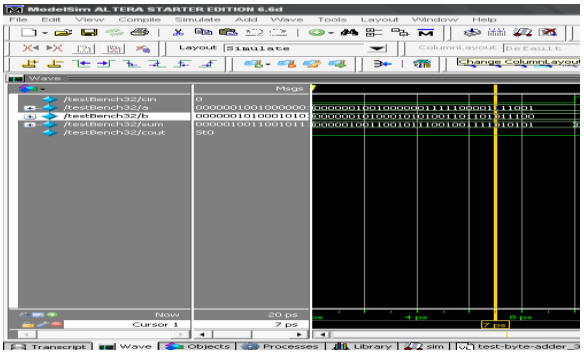


Fig.6.3: Simulation result for 32 bit

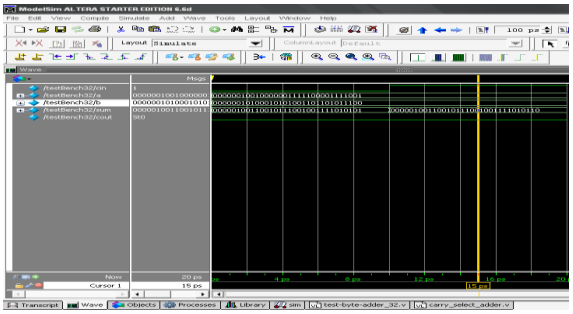
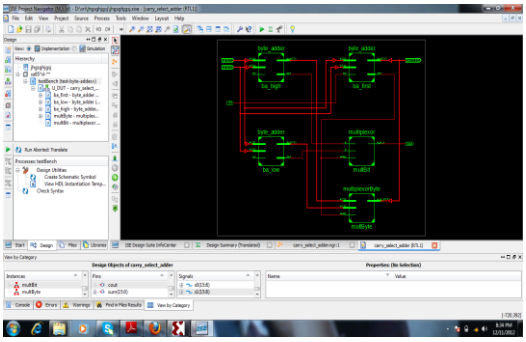


Fig 6.4: Simulation result for 32 bit:

Multiplier	Time Delay
RCA	18.772ns
CLA	15.628ns

Table 1: Above tables shows timing Report



The marginal improvement in speed increases with the rise in the word size of the adders. This shows that the design can be very well incorporated into complex VLSI Designs and DSP applications in order to increase the operating speed of the circuits

CONCLUSION

The adder is the basic element in the CPU. Digital Adders are the core block of DSP processors. The final carry propagation adder (CPA) structure of many adders constitutes high carry propagation delay and this delay reduces the overall performance of the DSP processor. All of the Adder-Subtractor, Multiplier etc are constructed with adders. Therefore to speed up the Adder efficiently is very important to the CPU or Processor.

The proposed structure proves to be an easier solution for improving the speed of carry select adder. The conventional CSA suffers from the disadvantage of occupying more chip area, which has been overcome using the proposed 4-bit incrementer unit. The proposed unit is also found to consume less power. The proposed carry select adder can be used to speed up the final addition in parallel multiplier circuits and other architectures which uses adder circuits.

FUTURE SCOPE:

Now days, Carry Select Adder is used in many Data-processing processors to perform fast arithmetic operations. The speed of the Proposed Carry Select Adder is greater than the Conventional Carry Select Adder, but the area and power are reduced. So, the Conventional Carry Select Adder can be replaced by Proposed Carry Select Adder where the Speed and power are the major constraints.

REFERENCES

[1] S. Cotofana, C. Lageweg, and S. Vassiliadis, “Addition related arithmetic operations via controlled transport of charge”, *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 243-256, Mar. 2005.

[2] V. Beiu, S. Aunet, J. Nyathi, R. R. Rydberg, and W. Ibrahim, “Serial A ddition: Locally Connected Architectures”, *IEEE Transactions on Circuits and Systems-I: Regular papers*, vol. 54, no. 11, pp. 2564-2579, Nov. 2007.

[3] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Quality programmable vector processors for approximate computing”, *46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1-12, Dec. 2013.

[4] A. B. Kahng, and S. Kang, “Accuracy-configurable adder for approximate arithmetic designs”, *IEEE/ACM Design Automation Conference (DAC)*, pp. 820-825, Jun. 2010.

[5] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, “On Reconfiguration- Oriented Approximate Adder Design and Its Application”, *IEEE/ACM International*

Conference on Computer-Aided Design (ICCAD), pp. 48-54, Nov. 2013.

[6] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power digital signal processing using approximate adders", *IEEE Transactions on Comptuer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124-137, Jan. 2013.

[7] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired imprecise computational blocks for efficient VLSI implementation of Soft-Computing applications", *IEEE Transactions on Circuits and Systems I: Regular papers*, vol. 57, no. 4, pp. 850-862, Apr. 2010.

[8] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: modeling and analysis of circuits for approximate computing", *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 667-673, Nov. 2011.

[9] NanGate, Inc. NanGate FreePDK45 Open Cell Library, http://www.nangate.com/?page_id=2325, 2008

[10] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders", *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760-1771, Sep. 2013.

[11] M. S. Lau, K. V. Ling, and Y. C. Chu, "Energy-Aware probabilistic multiplier: Design and Analysis", *International Conference on Compliers, architecture, and synthesis for embedded systems*, pp. 281-290, Oct. 2009.

[12] T. Yang, T. Ukezono, and T. Sato, "A Low-Power Configurable Adder for Approximate Applications", *19th International Symposium on Quality Electronic Design*, Mar. 2018.

[13]. O.J.Bedrij, "Carry-Select Adder", IRE Transactions on Electronic Computers, Pp. 340-344, 1962.

[14]. A.K.W. Yeung and R.K. Yu, "A self-timed multiplier with optimized final adder", Univ. California Berkeley, Final Rep.,CS 2921, Fall 1989.

[15]. C.S. Wallace, "A suggestion for a fast multiplier", IEEE Trans.on Computers, Vol.13, Pp, 14-17, 1964.

[16]. J. Skansky, "Conditional-Sum Addition Logic", IRE Trans. onElectronic Computers, EC-9, Pp. 226-231, 1960.

[17]. V.G. Oklobdzija, "High-Speed VLSI Arithmetic Units: Adders and Multipliers", in "Design of High-Performance Microprocessor Circuits", Book edited by A.Chandrakasan, IEEE press, 2000.

[18]. B.Ramkumar, Harish M Kittur and P.Mahesh Kannan, "ASIC Implementation of Modified Faster Carry Save Adder",European Journal of Scientific Research, Vol.42 No.1,Pp.53-58, 2010.

[19] Stan. M.R.; Tenca. A.F.: Ercegovac, M.D "Long and fastup/down counters." *IEEE Transactions* on *Computers.V*ol. 477 . July 1998. Pp. 722 -735.

[20] Lutz, D.R., Jayasimha, D.N., "Programmable modulo-Kcounters." *IEEE Transactions* on *Circuits and SystemsI:Fundamental Theory &Applications.*Volume: 43 11.Nov.1996. Pp. 939 -941.

[21] M.W. Evans, "Minimal Logic Synchronous Up/Down Counter – Implementations for CMOS." *US. Patent* no.4.611,337, Sept. 1986

[22] N.West. and K. Eshraghian. " Principles of CMOS VLSIDesign." Reading. MA: Addison-Wesley. 1985, Chapter 8

[23] Chung-Hsun Huang, Jinn-Shyan Wang and Yan-Chao*Hziang "*A High-speed CMOS Incrementer-Decrementer", Institute of Electrical Engineering, National Chung Cheng University 160 San-Hsing, Chia-Yi, 621, Taiwan, R.O.C.

[24] Shaoqiang Bi, Wei Wang, and Asim Al-Khalili, Multiplexer-based Binary Incrementer/decrementers, The 3rd International IEEE NEWCAS Conference, 2005,