DESIGN OF HIGH SPEED SORTING NETWORK USINGCOUNTERS AND COMPRESSORS FOR FAST BINARY COMMUNICATION SYSTEMS

[1]P.Suryanarayana,[2]Dr.B.Raja Rao, [3]Lanke meena
[1]Assistant Professor, Department of Electronics & communication Engineering
Eluru college of Engineering and Technology,Eluru,Andhra Pradesh
[2] Professor, Head of the Department , Electronics & communication Engineering
Eluru college of Engineering and Technology,Eluru,Andhra Pradesh
[3]M.Tech student, Department of Electronics & communication engineering Eluru
college of Engineering and Technology,Eluru,Andhra Pradesh

ABSTRACT

Parallel counters are essential in many number juggling circuits, notably in fast multipliers. An important feature of many digital signal processors units is the summing of several operands in parallel. Counters and compressors with high compression ratios are required for faster summing. Fast saturation binary counters and exact/approximate (4:2) compressor built on the sorting network are shown in this article. Sorting networks are used to reorganize the sequences of the counter's inputs, which may be represented only by one-hot code sequence. There are three particular Boolean equations that may greatly simplify the output of the counter between both the reorganized sequences with the one-hot code sequence. In addition, this research uses parallel sorting algorithms to find/sort M biggest values from N inputs, and then create scalable structures based on provided techniques. Other sorting methods have been developed for determining the greatest values. It is possible to implement BITONIC SORTING efficiently using parameter optimizations of one kind or another.

1.1 INTRODUCTION

Almost all electronic devices, particularly portable ones like smart phones, tablets, and other gadgets, need energy reduction. It is very desirable to achieve this reduction in size while incurring the smallest possible performance (speed) cost for multimedia applications; digital signal processing (DSP) blocks are the most sought-after transportable components. In the ALU, multiplications and additions make up the bulk of these blocks' computations. It is the primary function of the processing components to multiply, and this might result in significant energy and power consumption. The ultimate outputs of many DSP cores are pictures or movies suitable for human consumption, since these algorithms are implemented on many of the DSP cores. It makes it easier to use approximations to improve the arithmetic circuits' speed and energy consumption. This is due to the limitations of human perception while viewing images or videos. Additionally, there are applications where the accuracy of the arithmetic is not important to the system's functioning. Accuracy, speed, and power usage are all advantages of approximate computing. Accurate multipliers have the benefit of reducing mistake and increasing speed. To fix the division mistake, a comparison operation and a memory lookup for each operand is needed, which increases the delay time for the whole multiplication procedure. The estimate is processed at several levels of abstraction, including circuit, logic, and architecture. These approximation approaches in function include

many distinct arithmetic building elements, such as full adder and multipliers, at various design levels. VLSI implementation efficiency necessitated the creation of a broken array multiplier. With respect to the accurate WPA, the imprecise model's mean and variance errors rise only by 0.63 percent and 0.86 percent, respectively, whereas the maximum error rises by 4 percent. Low-Power DSP employs approximation adders, which are used in a variety of signal processing techniques and designs. The ETM's dissipated power decreased from 75% to 90% compared to the normal multiplier. The suggested ETM reduces the cost of a 12 x 12 fixed-width multiplier by more than half while keeping the existing method's lower average inaccuracy.

1.2 ADVANTAGES

- More secured with almost low complexity
- Low density and less market to time Minimum combinational path delay High throughput

1.3 APPLICATIONS

- Adders' multipliers
- Finite Impulse response filtersFast Fourier transformation Parity codes generations

2. LITERATURE REVIEW

An old-fashioned way of slowing down the ageing process is to use overdesign methods like guard-banding and gate over sizing. This method may be wasteful in terms of both space and power. Proposes an NBTI-aware technology mapping approach that guarantees the circuit's performance during its lifespan. An NBTI-aware sleep transistor in improved the lifetime stability of the voltage gated circuits under study. Another method the detection of functional symmetry and transistor stacking effects is the basis of a combined logic restructuring / pin reordering approach in Path sensitization was taken into account while developing this NBTI optimization technique presented dynamic volt scaling and bogy-biasing strategies to minimize power consumption or increase circuit lifespan. Some of these approaches need changes to the circuit or do not optimize certain circuits. VLSI circuits all have their own delay, which lowers the chip's performance. Clock cycles in conventional designs are determined by a factor known as the critical path delay.

OSCALETAL (2003) to create low-power 2's complement multipliers by reducing the switching activity of partial products. Booth codes are generated for the data with a narrower effective dynamic range prior to the calculation of two input data, improving the probability that the partial products are zero. "The multiplier with a column-based adder tree of compressors or counters is created by using the dynamic range determination unit to manage the input data routes.. Row-based and hybrid-based adder trees are used to implement the two multipliers in order to further minimize power usage. They are able to save their prior input states for non-effective dynamic data ranges and so minimize the amount of their switching actions. The theoretical analyses of switching activities of partial products are used to demonstrate the suggested multipliers with reduced power dissipation. The suggested low-power dissipation multiplier is shown, and theoretical assessments of partial product switching activities are generated. This multiplier uses a tiny fraction of the power of traditional

multipliers.

SARAVANAN & MADHESWARAN (2010) The low-power MAC unit created is tested for image processing systems that use unimportant bits in pixels values and the similarityof adjacent pixels in video streams. To decrease dynamic power consumption, the suggested technology analyses bit patterns in input three and applies the proposed encoding technique, or may utilize Booth's method. To reduce power consumption, the MAC block's suggested adder cell has been designed. Processing images is a logical use for this powerful yet energy-efficientMAC.

ECONOMAKOSET AL. (2010), a novel method for designing low-power combinational circuits was explored. The main concept is to use minimal latency and area overhead components to skip logic blocks when their function is not needed (transmission gates). Switching activity is reduced, resulting in low dynamic power consumption, since the internal state of these blocks is not altered. Because of their regular connectivity strategy, array multipliers save the most money with this design, but the smaller footprint and higher speed of tree multipliers attract the designer.

RASHIDI AND POURORMAZD (2011) to minimize power consumption, these solutions include low-power serial multipliers and serial adders as well as combinational booth multipliers and shift/add multipliers in linear phase architectures glitching is also reduced as a result. At 100 MHZ, the shift/add multiplier with 8 taps and 8 bits of input and coefficients results in a minimum output power of 110 mw in fir filter.

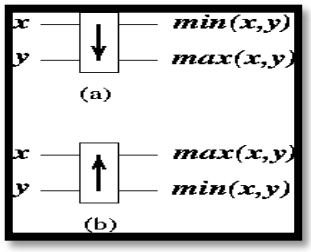
RAO AND COLLEAGUES (2009) Today's digital signal processing relies heavily on low-power multipliers with high clock rates. The performance of Wallace- tree, Array, and Baugh-Woolley multiplier designs is examined in this study. *HSPICE* is used to verify the sub-blocks' functioning and to optimize for low power consumption by scaling transistors.

3.1 COMPRESSOR ADDERS

Traditional adders using full adders and half adders often have a longer delay time than compressor adders. N-r, where N is the number of bits and r is the total count of 1s containedin N bits, is used to represent it. Compressor refers to the fact that this adder circuit has fewer gates and a shorter latency than other adder circuits. Improved lower-order compressor circuits are the subject of research [2]. This section explains the compressor circuits that may be used for multiplication processes such as 5-3, 10-4, 15-4, and 20-5.

The greatest number of bits that may be added in this compressor operational amplifier is five, and the result is three. One hundred one is the three-bit binary equivalent of decimal number five. A 4:1 multiplexer allows only one output to just be high ata time, which reduces the delay and uses less power, as shown in the figure n2. FIG. 2 depicts an incisive simulator-generated schematic representation of a 10-4 compressor adder. A maximum of 10 bits may be added to this compressor and a result output four bits is achieved. The maximum possible value is 1010, which is the four-bit binary representation of the decimal number 10.

3.2 SORTING



An interconnected compare-and-exchange (CAE) block is called a sorting network. Comparators are hardware implementations of the CE operation in sorting networks. There are two inputs and two outputs on a comparator. As seen in figure 1 (a), there is two distinct types of comparators, each with its own unique sense of ordering (b).

3.3 SORTING ALGORITHM

To put it another way, sorting algorithms are algorithms designed to arrange data in a certain way. Numeral and lexicographical order are the most often utilized. Search and merging algorithms, as well as canonicalization and the creation of human-readable output, may benefit from efficient sorting, which is critical for maximizing the usage of other algorithms that need sorted lists of input data. To be more formal, the result must meet two requirements: a permutation of the input results in a non-decreasing output (each element is not smaller than the one before it in terms of total order). It is also common to think of data as an array instead of a list since it is easier to access the data randomly, although techniques may be applied to both types of data. Because of the difficulty of addressing it effectively, the sorting issue has been the subject of a significant lot of study since computing inception. Bubble sort, for example, was studied as early as 1956.

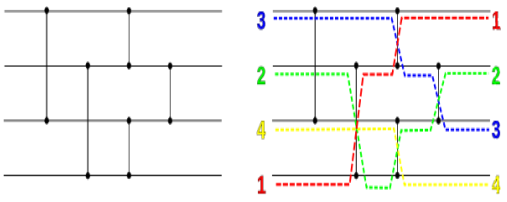
3.4 SIMPLE SORTS

Insertion sort & selection sort are two of the simplest sorts, both of which are effective on little data owing to minimal overhead, but not effective on huge data. For almost-sorted data, insert sort is quicker than selection sort because it needs fewer comparisons and performs better, while selection sort necessitates fewer writes and is thus favored when write speed is a concern.

3.5 SORTING NETWORKS

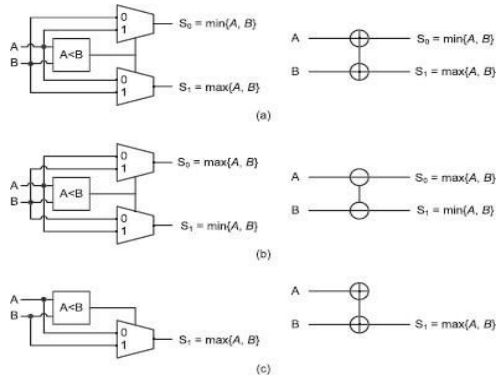
Mathematical models of wires and comparator modules that are used to sort a series of integers are called "sorting networks." Using two separate wires, a comparator may output the smaller of two values and sort them one by one. While comparison sorting algorithms are based on the results of prior comparisons, sorting networks have a fixed order in which comparisons are performed. Parallel algorithms benefit from this comparison sequence independence. Although the concept is rather simple, sorting network theory is quite complicated. Armstrong, Nelson, and O'Connor initially came up with the concept in 1954, and patented it the following year. Hardware or software sorting networks may be developed. Binary integer comparators may be built as simple, three-state electrical devices, according to Donald Knuth's work. When Batcher came up with the idea in 1968, he proposed utilizing them to build computer hardware switching networks instead of crossbar switches.

3.6 PARALLEL SORTING NETWORKS



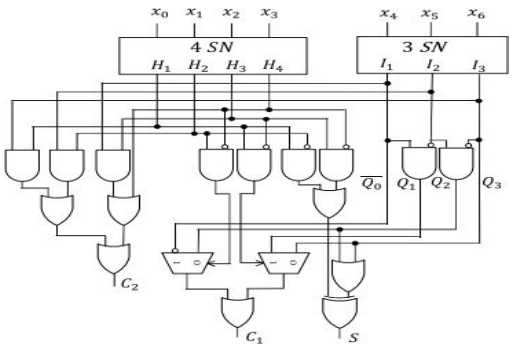
Compare-and-exchange (CAE) blocks are coupled to form a sorting network that

synchronizes a parallel set of inputs and outputs in a sorted order. There are two inputs and two outputs on each CAE block. There is no need to reorder the input values; if so, they are sent to the correct outputs. There are two kinds of CAE blocks used in hardware sorting: rising CAE blocks and decreasing CAE blocks



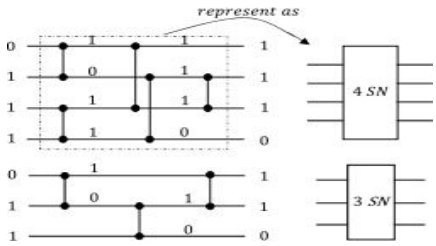
4.1 EXISTING TECHNIQUE:

In order to go to C2, C1, and S, sequences H and I must travel via C2 and C1. Increases the circuit's parallelism. In contrast, the suggested design's area would not grow as a result of parallelism.

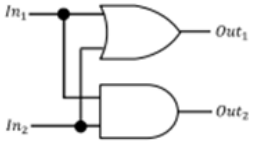


4.2 IMPLEMENTATION:

You can see how this works by looking at the Fig. above, which shows that all '1's' are placed on top of a series if they exist, and all '0's' are placed at the bottom if they do not exist in that sequence. A point in the reorganized sequence must be found where "1" and "0" meet, if there are both "1" and "0."



To ensure that a 0,1-junction exists at all times, we may control the sequence by padding fixed 1 bit "1" at the top and one bit "0" at the bottom of the reorganized sequence. Second, the total number of "1s" and "0s" in the rearranged sequence is the same as in the original (the inputs of two sorting networks). In order to count the amount of "1"s in the padding sequence, we may disregard the padded "1" since it is fixed.

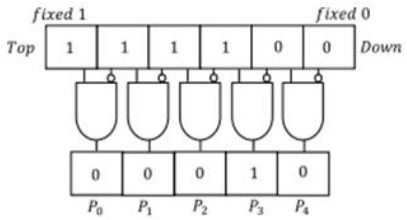


Two-input binary sorter

4.3 ONE-HOT CODE GENERATION CIRCUIT

In both three- and four-way sorting networks, three layers of binary sorter are required.

1) Asymmetric Pre-recorded (the two binary sorters on the same layer in four way sorting network can be calculated in parallel). Each layer of binary sorters uses one basic 2 distinct gate layer. Because of this, the three- and four-way sort networks need almost the same amount of time to complete their tasks. As a result of this, we may split a (7, 3) counter's seven inputs in half. The first component has four bits, whereas the second part has three.



One-hot code generation circuit

CODE SEQUENCE:

Find the 0/1-Junction: As long as "0" and "1" remain constant, the 0, 1- junction can only represent a reordered sequence. Observe that the location of the 0, 1-junction must be 1,0 on the left to right. A Boolean expression (AB) in this structure generates the new sequence P0–P4 from the four-way sorting network, which we will use as an example here. Sequence P0–P4 has just one "1" because the reordered and expanded sequence has only one 0, 1-junction. Code P0–P4 is a one-hot sequence that meets the "|" requirement

4.4 OPTIMIZATION:

Both H1–H4 and I1–I3 are available to us. H0 (the fixed "1" in Figure 4) and H5 (the fixed "0") are added to the end of sequence H1–H4. Let's do this for the first sequence. I0 signifies the fixed "1," and I4 denotes the fixed "0." In other words, we have the equation as follows:

$$P_i = H_i \& \overline{H_{i+1}}, \quad i = 0, 1, 2, 3, 4$$
$$Q_i = I_i \& \overline{I_{i+1}}, \quad i = 0, 1, 2, 3.$$

The outcome of "OR" up the subsequences taken from the sequence Q or P may be stated as sequence I or H (P1|P2|P3 = H1&H4 for example) if their subscripts are sequential (P1, P2, and P3 for example). It is therefore generalized as the Logical equations acontinuous "OR" with the comma "" in it.

$$\sum_{n=i}^{n=j} P_n = H_i \& \overline{H_{j+1}}, \quad (0 \le i \le j \le 4)$$
$$\sum_{n=i}^{n=j} Q_n = I_i \& \overline{I_{j+1}}, \quad (0 \le i \le j \le 3).$$

$$\begin{aligned} C_1 &= (Q_0 \& (P_2|P_3))\,|\,(Q_1 \& (P_1|P_2))\,|\,\left(Q_2 \& \left(\overline{P_2|P_3}\right)\right) \\ &\quad |\,\left(Q_3 \& \left(\overline{P_1|P_2}\right)\right) \\ &= \left(Q_0 \& \left(H_2|\overline{H_4}\right)\right)|\left(Q_1 \& \left(H_1|\overline{H_3}\right)\right)|\left(Q_2 \& \left(\overline{H_2|H_4}\right)\right) \\ &\quad |\left(Q_3 \& \left(\overline{H_1|H_3}\right)\right). \end{aligned} \tag{14}$$

$$I_i = I_i|I_{i+j}, \quad (i = 0, 1, 2, 3;\ j \ge 0;\ i + j \le 4)$$
$$H_i = H_i|H_{i+j}, \quad (i = 0, 1, 2, 3, 4;\ j \ge 0;\ i + j \le 5).$$

$$\begin{aligned} C_2 &= (P_4 \& (Q_0|Q_1|Q_2|Q_3))|(P_3 \& (Q_1|Q_2|Q_3))| \\ &\quad (P_2 \& (Q_2|Q_3))|(P_1 \& Q_3) \\ &= H_4|\left(H_3 \& \overline{H_4} \& I_1\right)|\left(H_2 \& \overline{H_3} \& I_2\right)| \\ &\quad \left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& (I_1|I_2))|\left(H_2 \& \overline{H_3} \& I_2\right)|\left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& I_1)|\left(\left(H_2 \& \overline{H_3}|H_3\right) \& I_2\right)|\left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& I_1)|((H_2|H_3) \& I_2)|\left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& (I_1|I_2))|(H_2 \& I_2)|\left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& (I_1))|(H_2 \& I_2)|\left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& (I_1))(H_2 \& (I_2|I_3))|\left(H_1 \& \overline{H_2} \& I_3\right) \\ &= H_4|(H_3 \& I_1)|(H_1 \& I_3)|(H_2 \& I_2). \end{aligned}$$

5. PROPOSED BITONIC SORTING

Bitonic sort is a comparison-based parallel sorting algorithm. When compared to other well-known sorting algorithms, Bitonic sort performs a greater number of comparisons. In a parallel implementation, this sort is superior since the user compares the components in a specified order that is independent of the data itself. That's why bits are a good choice for hardware implementations. First, we need to know what a Bitonic sequence is and how it's produced bitonic before we can grasp Bitonic sort.

STEPS:

- It's necessary to start with a comparison of element 1 from half one to element 1 from the second half, then go on to element 2

from half one and so on until the last element of half oneis compared to the final element of the second half. If the second half's element is smaller than the first half's, swap them.

- In this step, the initial half of the components will shrink in comparison to the second half. Two sequences of length n/2 are compared and swapped. Recursively apply step 2 to each sequence until there is just one n-length sorted sequence.

- Based on the 0-1-principle, bionic sort is established here. Every series of zeroes and ones is sorted according to the 0-1-principle; hence any comparator network that does the same sorts all sequences of arbitrary values. In order to sort in parallel, the Bitonicmerge sort algorithm is used. It may also be used to design a network of sorting nodes. By Ken Batcher, the algorithm was developed. In this case, the sorting networks are composed of comparators and have a delay of, where is the number of items to be sorted.

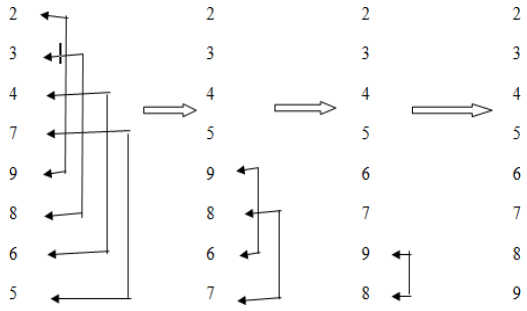
- A monotonically decreasing (or rising) sequence is what we mean by a "sorted" sequence. It is possible to think of a bitonic sequence as one where there is a change in one direction but not the other.

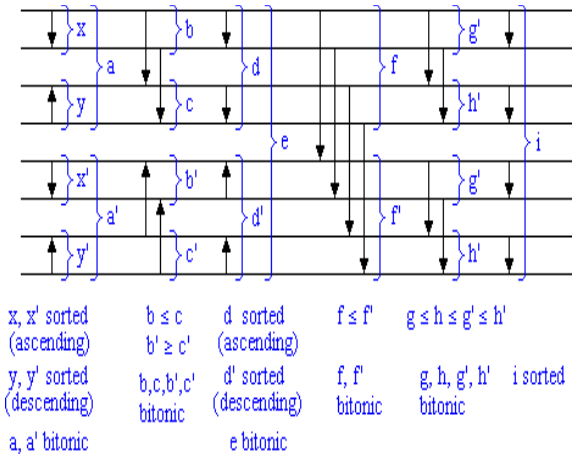
- A total of 16 digits are fed into the system at the left end, where they travel down each of the 16 wires before exiting out of the right end. The network is set up to place the most numerous items at the bottom of the list.

- They serve as indicators. It is necessary to compare the values at either end of an arrow to guarantee that the arrow always points toward the greater number. They are exchanged if they are out of sequence. For the sake of clarity, the colored boxes have no bearing on the algorithm.

- Then the boxes in that column sort the sequences in that column. A sorted list of all the input is created by starting with each column as an unsorted list of one element and working its way down to the final column, which combines everything into one list. Due to the final stage being blue, the biggest item will be at the bottom of the list.

- Let us see in practical How Bitonic sort is done, following the above algorithm, Bitonic sequence: 2, 3, 4, 7, 9, 8, 6, 5



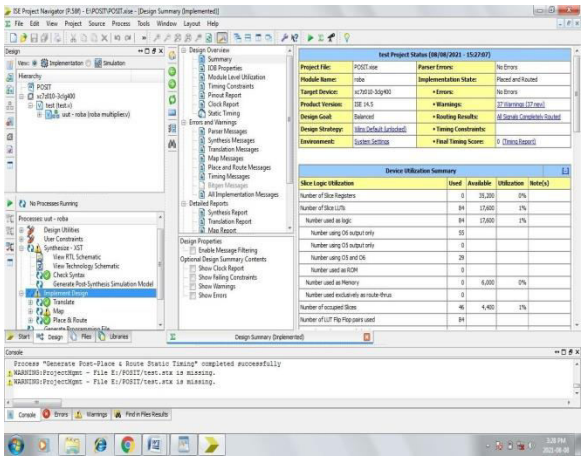


Sorting network *BITONIC* Sort for n = 8

6. SOFTWARE IMPLEMENTATION USING XILINX

In this section, we'll go through the project's software requirements. Xilinx ISE is the programmed utilized in this case. VERILOG HDL, a programming language, may be used to createthe programmed code. The behavior of the system is also generated for the specific target device.

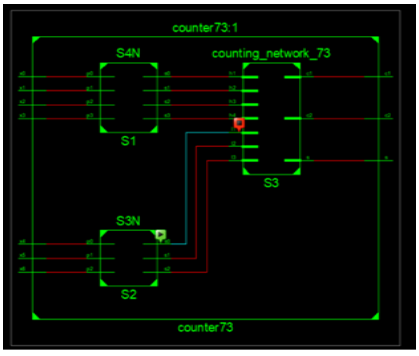
- Click on the Task Manager Icon to begin using Xilinx Tools
- The desktop of a Windows computer. This should bring up a new window in Project Navigatorfor you.
- This window displays the most recent project that was visited.



Xilinx Project Navigator window (snapshot from Xilinx ISE software)

7. RESULTS

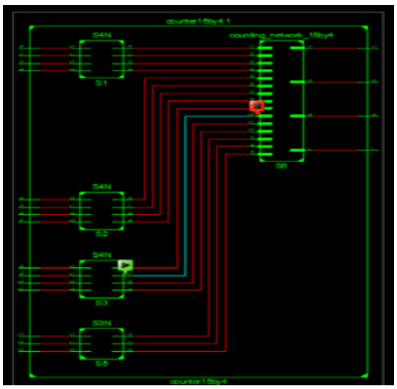
In the current system, the (6, 3) and (7, 3) Counters have a symmetrical number of inputs. This design has both full and half adders. There is a longer delay and a bigger amount of circuit area used by the increased number of XOR gates. Because of the increased number of XOR gates, the circuit's hardware complexity and power consumption are both high. The suggested counters use a sorting network to arrange the (7, 3) and (15,4) counts in symmetrical order. Using this new design would reduce the number of full and half adders required. Because there are no XOR gates in these counters, the latency is less and the amount of space used up by the circuit is reduced. The device uses a little amount of electricity.



Schematic diagram of (7, 3) Counter

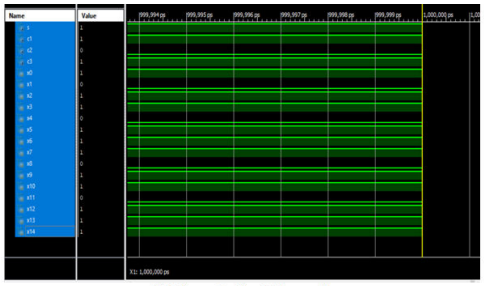
(7, 3) Output waveform

The (15, 4) counter design schematic required connections are givenusing the wire and generate the symbols



Schematic diagram of (15,4) Counter

Insertion sort is a simple and efficient sorting algorithm useful for small lists and mostly sorted list Quick sort is the fastest general purpose internal sorting algorithm on the average among other sophisticated algorithms The read phase effectively sorts and outputs the data elements using a matrix multiplication (ANDING) operation, rather than comparison operations, as in prior work



(15, 4) counter Simulation result

COMPARISON TABLE OFEXISTING SYSTEMITH PROPOSED SYSTEMS

S/no	Design	Area (no.of.LUT's)	Power (W)	Delay (ns)
1	Symmetric stacking (11)	8	0.081	1.8
2	Proposed (7,3) Counter	3	0.042	1.728
3	Parallel (15,4) Counter (12)	15	0.095	4.1
4	Proposed (15,4) Counter	13	0.042	4.061

8.1 CONCLUSION:

Here, an effective, but regionally feasible, countermeasure approach is devised and implemented. Symmetrical piece: A two-fold counter A method for aggregating and transmitting computation has been presented. On the basis of this novel counter- design methodology, we build counters such as 7 (3), 15 (4), based on biotonic sorting. Since there is reduced latency and greater performance in ADP, the proposed counter are more adaptable than current designs.

9. FUTURE SCOPE:

An approximation multiplier may be used to test the picture, video, and video clarity. In addition, they're worn out from trying to match the precision of the precise multipliers. We may utilize Brent Kung Adder the minimize area and Kogge Stone Adder to speed up multiplication.

10. REFERENCES

[1]WEN-CHANG YEH AND CHEIN-WEI JEN, "High-speed Booth encoded parallel multiplier design," IEEE Trans. on Computers, vol. 49, issue 7, pp. 692-701, July2000.

[2]JUNG-YUP KANG AND JEAN-LUC GAUDIOT,

"A simple high-speed multiplier design," IEEE Trans. on Computers, vol. 55, issue 10, Oct. pp. 1253-1258, 2006.

[3]SHIANN-RONG KUANG, JIUN-PING WANG AND CANG-YUAN GUO, "Modified Booth multipliers with a regular partial product array," IEEE Trans. on Circuit and Systems, vol.56, Issue 5, pp. 404-408, May 2009.

[4]LI-RONG WANG, SHYH-JYEJOU AND CHUNG-LEN LEE, "A well-structured modified Booth multiplier design," Proc. of IEEE VLSI-DAT, pp. 85-88, April 2008.

[5]A. KHATIBZADEH, K. RAAHEMIFAR AND M. AHMADI, "A 1.8V 1.1GHz NovelDigital Multiplier," Proc. of IEEE CCECE, pp. 686-689, May 2005.

[6] S. HUS, V. VENKATRAMAN, S. MATHEW, H. KAUL, M. ANDERS, S. DIGHE, W. BURLESON AND R. KRISHNAMURTHY, "A 2GHZ 13.6mW 12x9b multiplier for energy efficient FFT accelerators," Proc. of IEEE ESSCIRC, pp. 199-202, Sept. 2005.

[7]HWANG-CHERNG CHOW AND I-CHYN WEY, "A 3.3V 1GHz high speed pipelined Booth multiplier," Proc. of IEEE ISCAS, vol. 1, pp. 457-460, May 2002.