IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 6, Ver. I (Nov. - Dec. 2016), PP 31-39
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Fast Processing Unit Design to Mulitplier Using Block Unit Counting

N. Srinivas[1], Y. Rajasree Rao[2]
[1]*Department of ECE, Guru Nanak Institute of Technology, Hyderabad, Telangana,India*
[2]*Department of ECE, St. Peters Engineering College, Hyderabad,Telangana, India*

Abstract: *Multipliers are considered to be the basic building unit of all-basic to complex design units. This design unit operates on two operands and generates a result after successive bit wise anding and oring operation. With higher-level logical operation, the anding and oring operation is defined by addition and bit multiplication. The two operands are buffered on a register and temporary registers are used to buffer the intermediate results for adding these results. For faster processing, this block unit is defined in multiple ways for achieving optimization. Among these approaches, the multiplied design using parallel processing is adaptively been used. However, the resource overhead in such design is very large resulting in high area coverage and power dissipation. To overcome the stated issue in this paper, a new design approach following block unit count operation is suggested. The logical overhead in such design is minimized with the approach of redundant bit count in blocks.*

Keyword: *Multiplier design, fast processing, parallel multiplier, block unit counting*

I. Introduction

With the emergence of new technologies, higher level services are getting integrated over portable devices. New features such as conferencing, Televisions, internet application, gaming etc. are coming up. In the area of mobile communication, these applications are in greater demand. To obtain such services in portable devices, new processing units are interfaced. These units are developed with new technologies and definition to give highest level of performances. In all these applications the received analog signal is processed for estimation, filtration and regeneration at the receiver unit. During the communication process, the transmission of digital information is performed over a analog carrier and the receiver unit recover these digital information's using a processor unit and estimating back using estimator unit. In the process of Analog to Digital conversion, it is observed that, a quantization error is observed due to truncation or rounding operation. To obtain an effective design to a processor unit, several popular architectures, such as the flash, pipeline [2], cyclic [3], and delta–sigma converters [4] etc for the design approach for analog-Digital designing. Each of these architectures, however, can work optimally only at a narrow range of resolution, bandwidth, and power. For example, a standard pipeline converter works optimally at low to- medium resolutions and medium-to-high speeds, while a delta–sigma converter works best at low speeds and delivers medium-to-high resolutions. A conventional processor with fixed topology and parameters cannot efficiently be employed for the task of digitizing signals at a wide range of bandwidth at varying resolution with adaptive power consumption. An alternate approach is to employ an array of Processors, each customized to work at narrow ranges of resolution and input bandwidth. Such a converter implementation, however, would require a prohibitively large number of Processors to achieve optimal power consumption with a reasonably fine granularity over input bandwidth and resolution. A single processor with reconfigurable parameters and reconfigurable topology would be able to achieve the above goal. Prior reconfigurable Processors, however, achieve very limited Re-configurability. For example, variable resolution in a delta–sigma converter has been proposed by changing oversampling ratio (OSR) and bias currents of the converter over a predetermined set of values obtained from a lookup table [5]. This fixed arrangement can offer only relatively limited resolution Re-configurability. In addition, relying on predetermined bias currents does not work over different fabrication processes without costly calibration. To achieve a multi rate processing unit in [7], [8] and a cyclic processor that can be configured The introduction of the paper should explain the nature of the problem, previous work, purpose, and the for 8, 13, or 16 bits [9]. Both of these provide only limited Re-configurability. This work proposes the idea of a single converter [10], [11] that can morph itself into different topologies to cover the desired continuum of resolution and bandwidth space with minimum power at each performance level. These processor coefficients are then processed for estimation to recover the transmitted information back. The processed information's are operanded for estimation. Among various approaches of estimation logic turbo based coders are more effectively been used for estimation. The performance is reportedly at least as good as with conventional concatenated codes using a Reed-Solomon outer code and convolution inner code [12]. Power savings is important for towers in especially

remote areas, where recursive codes are used. [13]. Recursive codes used by these two systems are very similar, the differences lie in the interleaving algorithm, the range of allowable input size and the rate of constituent RSC encoders [14]. The iteration of multiplier operations result in higher memory demand and also intern results in higher power consumption. To optimize the resource utilization and to achieve higher precision coding, in this paper a new mode of coding and a modified approach of multiplier operation is presented.

II. Conventional Parallel Multiplier Design

As evident through the transfer function and Figure 1, multiplication and addition are the main functions of digital signal processing. The input values and previous output values must be multiplied by certain coefficients, and then these products must be summed together. Also, each of these must be designed so that they can process negative two's compliment values due to the values of the coefficients. Some differences exist between the design in VHDL and VLSI. In VHDL, the first task was to determine the capabilities of the Xilinx compiler. To do this a ripple carry adder and a carry look-ahead adder were created with VHDL.A carry ripple adder has a delay of 2n+2 gate delays. To decrease this delay a carry look-ahead adder can be created. This type of adder requires more logic, but the speed advantage is significant. For example, a 16-bit carry ripple adder results in 34 gate delays, while a carry look-ahead adder only has 10 gate delays. To test the Xilinx compiler a simple adder (a + b = c) was created in VHDL. Then all three adders were simulated and the delays were compared to determine what type of adder the compiler created. The results showed that the complier created a carry look-ahead or equivalent adder. To improve the stability and accuracy of the adder, overflow protection was also included. If this was not added and an overflow occurred, then the processor would produce erroneous information. The design of the adder in VLSI was different than that in VHDL for different reasons. Because of the complexity of the carry-look-ahead adder, a simply ripple-carry adder was used in its place. Not only would design for a carry-look-ahead adder consume the most time, but it would also require a large amount of chip area. A cellular approach was used in the design of the ripple-carry adder. First, a simple one-bit full adder was designed. This design consisted of three inputs, two addend bits and carry-in bit, and produced the appropriate carry-out bit and sum bit. Many different multipliers were designed in VHDL to determine the best ratio of size versus speed. Two types of multipliers, serial and parallel, were investigated. A serial multiplier is easier to create and takes less area on the FPGA, but the speed is decreased nearly n times for an n-bit multiplier. A parallel multiplier has the advantage of speed and would be utilized if area were of no concern. To conclude what course to take, first, a serial multiplier was created. Digital signal processors utilize signed numbers, so a signed multiplier needed to be designed. A parallel multiplier was created using the Xilinx compiler. 4-bit, 8-bit, and 16-bit multipliers were created for both.

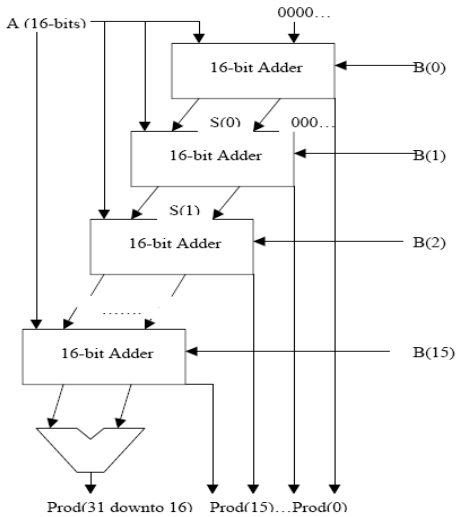


Figure.1. data path for signed 2's complement parallel multiplier

A parallel multiplier was used because it would, in fact, fit on the chip. To increase the speed even further a modified Booth's Algorithm was used. Booth's algorithm increases the speed by cutting the number of necessary additions in half. The 16-bit Modified Booth's Multiplier had decreased the delay of 60 ns. One problem that arose with the multiplier was that it would occasionally produce an extra sign bit. It was discovered that this problem only arose when multiplier two identical numbers. Checking for that specific case and adjusting the product accordingly corrected this dilemma. As with the adder, the design of the multiplier differed for the VLSI portion of the project. Because of time constraints and complexity issues, a 4-bit by 4-bit cellular multiplier was designed instead of a 16-bit by 16-bit Booth's modified multiplier.

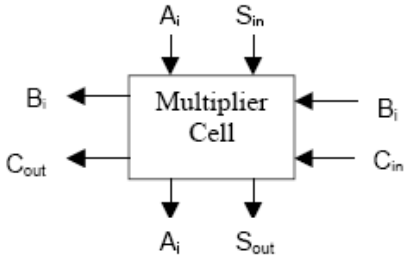


Figure.2: Block diagram of the multiplier cell used in the design of the 8-bit multiplier.

Unlike the adder, a simple cascaded design could not be used in the case of the multiplier. The cellular array shown in Figure.2 displays the interconnections of the multiplier cells that form the full circuit. In all, sixteen cells made up this design, and the total number of transistors exceeded one thousand. To test the circuit, five different simulations tested the sixteen different combinations for inputs a0 through a3 which were multiplied by zero, one, three, seven, and fifteen. The products in simulation were compared to answers from a calculator; the conclusions from these results showed that the multiplier functioned correctly. However, this multiplier only computed positive numbers, so circuitry needed to be added to the input and output stages so that it would compute negative numbers correctly. The input complement blocks must take a two's complement number and convert it to the corresponding positive number before being sent to the multiplier. The corresponding output complement block would then recognize that the product is either positive or negative based on the inputs and adjusts the output value accordingly. To accomplish this, the ripple-carry adder discussed previously and a 2-to-1 multiplexer were added to the input and output stages of the multiplier. To convert to a positive number, the two's complement negative number must undergo bit wise negation; then, one is added to the negated value. The multiplexer uses the sign bit to select the appropriate value to send into the multiplier. At the output, the select bit looks at the sign bits of both inputs and the carry-out of the most significant bit of the ripple carry adder used to convert the two's complement number.

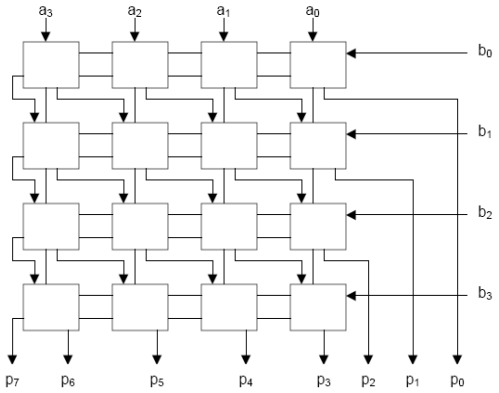


Figure.3. Cellular design of the 4-bit by 4-bit multiplier utilizing the multiplier cell in figure 2.

Based on these signals, the adjustment circuitry selects the appropriate positive number to send through the multiplier. Likewise, the output adjustment circuitry contains circuitry that selects the appropriate value to send out of the multiplier. However, the select bit at this stage looks at the sign bits of both inputs and the carry out of the most significant bit of the two's complement ripple-carry adder. Based on these three signals, the multiplexer selects the appropriate values to send out of the multiplier. In addition, there were two special cases that needed to correct. This occurred when the input was (10000)2, or –15 (the least negative input), and when the output was (100000000)2, or –255 (the least negative output). To adjust for these instances, another stage of multiplexers was added to the adjustment circuitry that specifically looked for these cases. The only time this occurred was when the sign bit of the input or output and the carry-out bit of the two-complement adjustment adder were both HIGH (and 5V). On these occasions, the adjustment circuitry would recognize these numbers and send the correct numbers into or out of the multiplier. These circuits were designed and tested separately. Once both the input and output adjustment circuits were working correctly, their cells were instantiated into the multiplier layout, and the correct connections were made between the appropriate cells. Once connected, the entire layout was checked for design errors, and the circuit file was then extracted and simulated. The results showed that the adjustment circuitry functioned correctly, and the multiplier now produced the correct two's complement numbers when specific inputs were given.

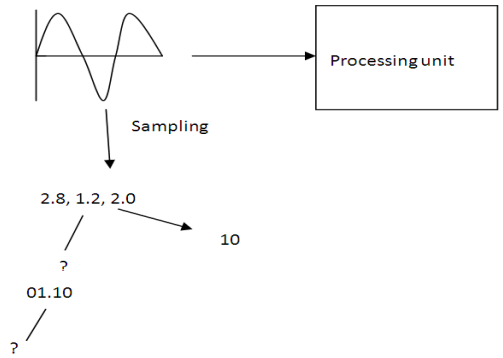


Figure 4: A real time signal sampling for digital processing

- 1.2 —Rounded→ 1 building truncation error.
- Fixed notation —— constraint with decimal value input
- Float notation —— has its advantage.

Figure 5: representation error in logical representation

The proposed system is realized using VHDL language for it's functional definition. The HDL modeling is carried out in top-down approach with user defined package support for floating point operation and structural modeling for recursive implementation of the operand bank logic. For the realization a package is defined with user defined record data type as;

Record type, { float representation},
sign :std_logic;
exp: std_logic_vector(3 downto 0);
mantissa: std_logic_vector(10 downto 0);

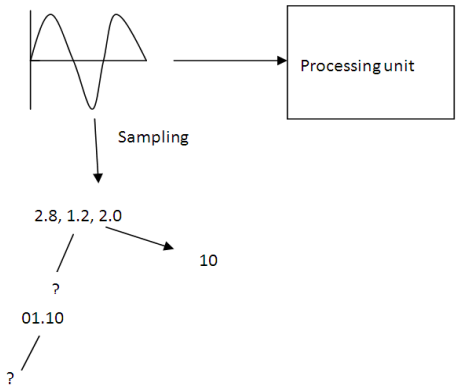
The floating notation is implemented using 16 bit IEEE-754 standards as presented below.

Sign. (1)	Exp. (4)	Mantissa (11)

Figure 6: Float logical representation of a 16 bit binary value

III. Block counter Design

The process of multiplication operation is observed in all real time applications. Considering a real time recorded signal as illustrated in figure 4, the samples are discritized and passed to processing in its equivalent bit level representation. However, the decimal representation of fixed integer notation, gives a full binary value, but the decimal value representation are basically been truncated. The representation due to truncation or rounding error results in precession loss. Hence to achieve the objective of higher precession accuracy, a float number processing is presented.



The floating-point addition, multiplication and shifting operation are implemented as procedures in the user defined package and are repeatedly called in the implementation for recursive operation. The proposed processing unit is defined in three basic functional blocks, multiplier, adder and register management unit. The processing block, handles all of the operand coefficients and previously stored values from the processor.

Looking at the equations for the digital operand, there are five coefficients that need to be stored in the data block: b0, b1, b2, a1, and a2. Also, the values ω(n-1) and ω(n-2) are values that are delayed through the system. These values need to be stored in memory in this data management block so they can be delivered to the adder and multiplier stages at the correct time. This will allow the processor to correctly compute the new output values of the signal. Therefore, this part of memory will be set up in a "shift-and-rotate" fashion so that when a new value enters the system, the old values will be rotated out and the delayed values can be stored correctly.

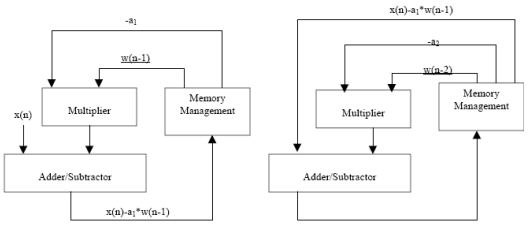


Figure.7. Block diagram showing consecutive steps in the data management

The same general flow was used for both the VLSI and VHDL designs. Seven cycles were used for the entire design in VHDL. The first cycle is used to trigger an analog-to-digital (A/D) converter. A temporary register was also created to store the data until the next cycle. The next five cycles load the data to the multiplier and adder. For each of these cycles the product from the multiplier is fed into adder. For this reason the clock that controls the cycles must be set with a period long enough for the product to be calculated by the multiplier and then summed by the adder. The last two cycles are used to shift the data to be used with the next input. The conventional system is realized using VHDL language for it's functional definition. The HDL modeling is carried out in top-down approach with user defined package support for floating point operation. for the realization of this multiplier operation a recursive multiplication and addition operation is carried out. This recursion results in multiple accumulation operation in successive clocks of duration amounting to that of an addition-time, so that the accumulation of N successive input words are performed in Nclock cycles. For large values of N, a computational latency of Naddition-time over head and high power consumption and slows the accumulation process to obtain an accumulated output could be too high to meet the timing requirement in real-time application. So in order to over this problem in this paper aa new scalable Repetitive multiply accumulates (Multipliers) is proposed. The proposed structure-1, as shown in Fig. 8(a), consists of two stages. The first stage is implemented by a counter module consisting of eight 3-bit ripple counters C3 [shown in Fig. 8(b)]. The second stage is implemented by an adder module consisting of a shift-add (SA)-tree, as shown in Fig. 8(a).

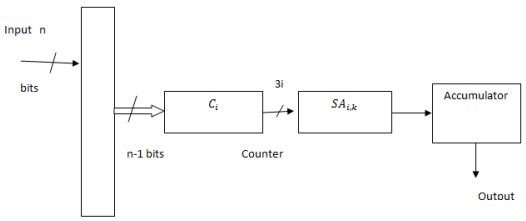


Figure 8: proposed block count multiplier unit

Where,
n= No. of parallel bits to process
i = No. of bit counter
K = No. of shift adder register

The time period of the clock used for the counter module is the same as the worst case propagation delay of a T flip-flop. On the first clock cycle, all the flip-flops of all the counters are reset and each counter is fed with a bit of input word available from the input register. After 7 clock cycles, the states of flip-flops of the counter represent the intermediate operand for (which equals to the number of 1's on the bit stream of a given place value of all the seven inputwords). Outputs of the counters are latched to four SA-1 cells at the first level of SA-tree. Each SA-1 shifts its vertical-input by one location to left; and adds its right-input with the left-shifted word. The second level of SA-tree consists of two SA-2 which shifts its left-input by two locations to left and adds that to its right-input. The last/third level of SA-tree consists of one SA-4 which shifts it's left-input by four locations to left and adds that to the right-input The SA-tree has a combinational path of duration $T_1 = T_{A3} + T_{A5} + T_{A7}$ in non-pipelined form to add the eight 3-bit outputs of counter module, where

T_{A3}, T_{A5} and T_{A7} are, respectively, the time required for 3-bit, 5-bit and 7-bit additions. Total accumulation-time of the structure for a non-pipelined implementation amounts to the sum of the delays of counter module and the adder module $T_c + T_1$, , where $T_c = N \cdot T_{TF}$ is the time involved in counter module. The counter module and the adder module could also be implemented in two separate pipelined stages, with a pipeline period $= max\ \{T_c, T_1\}$. Duration of T_c in general, depends on the number of input word , while T_1depends on$N$ as well as the word-length .For large values of L the depth of SA-tree becomes large and word-length grows as we go down on the SA-tree. The computation of adder module, therefore, becomes larger for larger values of . The adder module in that case could be implemented by a pipelined SA-tree to increase the throughput rate. Similarly, when N is large compared with , the bit-level accumulation-time in counter module becomes bigger than the adder module delay. For very large values of , it should be preferred to perform the accumulation in multiple pipelined stages of the proposed accumulators. In any case, the clock period of adder module, however, could be made an integer multiple of the clock period of counter module for efficient derivation of clock signals.

IV. Experimental result

Experimental results

For the evaluation of the suggested approach, in this work a parallel architecture based on tree structure is developed. The timing observation for the developed approach is as illustrated in below section.

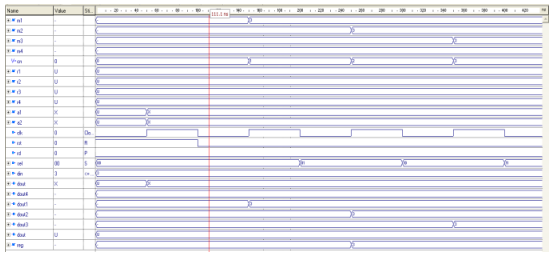


Figure 9: result illustrating the initial value for the multiplier unit design in parallel coding

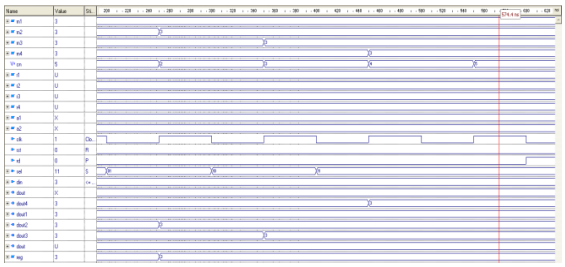


Figure 10: simulation result illustrating the temporary register buffering

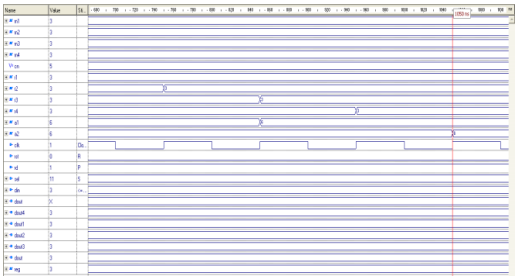


Figure 11: Simulation result showing the final result generation

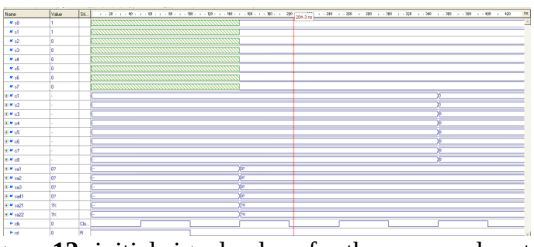


Figure 12: initial signal values for the proposed system

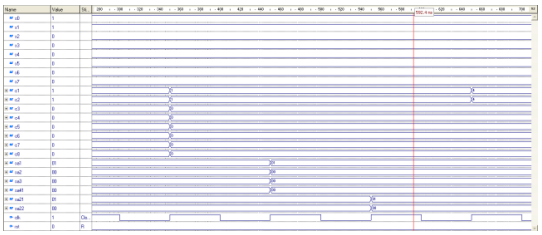


Figure 13: Simulation result showing result in temporary buffering

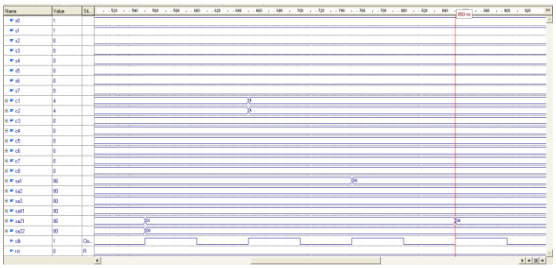


Figure 14: Simulation result illustrating the final result generation

The obtained multiplier product value of 6 is seen on the output line signal. The obtained result is derived at 850ns of system clock cycle. The same result for the conventional parallel computing is obtained at 1050ns. The simulation time period is hence observed to be conserved for 200ns. The reduction in time period is due to the reduction in temporary storage and intermediate operand buffering and computing of the two resulting operands. The resource overhead is observed to be minimized. The implementation realization of the developed approach targeting to the Xilinx FPGA device is as given below.

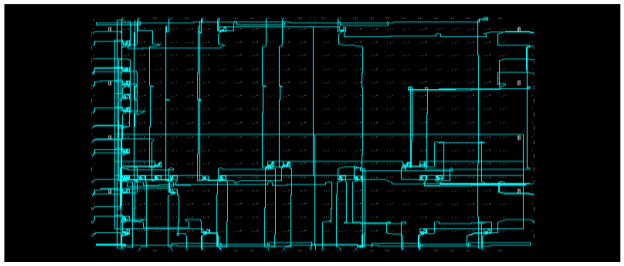


Figure 15: Logical routing of the implemented wavelet decomposing module targeting to Xc2s50e-ft256-7

Figure shows the logical routing of the implemented design targeting on to Xc2s50e-ft256-7 FPGA of virtex family. The routing is carried out on Xilinx FPGA editor. The result obtained shows the real time FPGA interconnection of logics connected inside the FPGA.

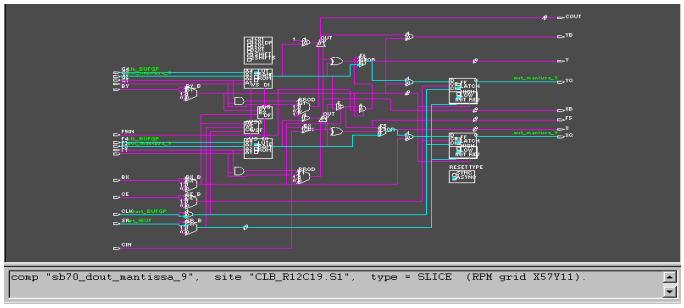


Figure 16: logical placement of the implemented design in the FPGA (xc2s50e-ft256-7)

Figure 16 shows the logical placement of the implemented design on to the targeted FPGA (Xc2s50e-ft256-7). The figure shows the logical resources used by the Logic implemented for the implemented design for one CLB. The basic elements used for the implementation were LUT and Buffer elements as shown in figure.

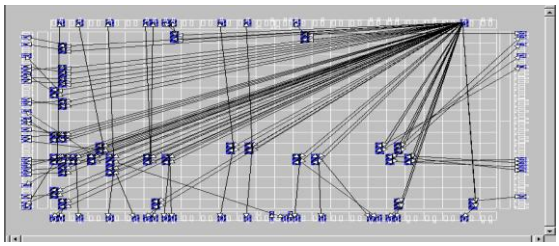


Fig 17 Floor planning of the implemented design in the targeted FPGA (Xc2s50e-ft256-7)

Figure 17 shows the floor planning of the implemented design on to the targeted FPGA (Xc2s50e-ft256-7). The figure shows the logical net connection between the two CLB and the IO buffer used. The interconnects obtained shows the path covered for the logical mapping of the resources for data transfer for the operation.

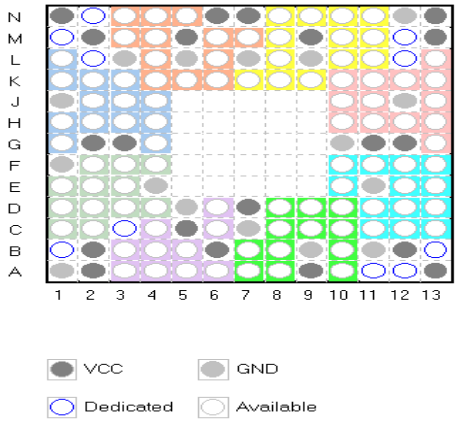


Fig 18 Bottom Package View for the Target FPGA(Xc2s50e-ft256-7)

Figure shows the bottom view of the targeted Virtex FPGA ((Xc2s50e-ft256-7) this package view is consisting of the power supply pins (VCC), the ground (GND) the dedicated lines used for data transfer and available pins for other applications.

A comparative analysis of the developed system over the conventional parallel multiplier structure is carried out as listed in table below.

Coding approach	Maximum frequency (MHz)	BELs	Power(mW)
Proposed coding	377.5	205	103
Conventional parallel coding	321.15	356	186

Table 1: Comparative analysis of the proposed method over conventional parallel coding approach.

V. Conclusion

This paper outlines a design methodology for multiplier design based on block counting at multiplier operation. An efficient algorithm of multiplier of multiple accumulations, where N input words of L-bit size are transformed either recursive codes which could be shift-accumulated to generate the desired accumulated sum. The proposed algorithm, offer low power consumption and less time complexity when compared to conventional methods. The proposed multiplier structure is found to involve less computation time. The proposed structure have potential advantages in many real time applications for implementation giving, adaptive operands and computation of real time signal. It offers a solution to multiplication processing in low resource overhead systems.

References

[1] S. K. Mitra, Digital Signal Processing: A Computer Based Approach. Boston, MA: McGraw-Hill, 2006.

[2] S. R. Vangal, Y. V. Hoskote, N. Y. Borkar, and A. Alvandpour, "A 6.2- GFLOPs floating-point multiply accumulator with conditional normalization," IEEE J. Solid-State Circuits, vol. 41, no. 10, pp. 2314–2323, Oct. 2006.

[3] L.-H. Chen, O. T. C. Chen, T.-Y.Wang, and Y.-C.Ma, "A multiplication- accumulation computation unit with optimized compressors and minimized switching activities," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05), May 2005, vol. 1, pp. 6118–6121.

[4] A. Fayed, W. Elgharbawy, and M. Bayoumi, "A data merging technique for high-speed low-power multiply accumulate units," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'04), May 2004, vol. 5, pp. 145–148.

[5] Liu, D.N.; Fitz, M.P., "Low complexity linear MMSE detector with recursive update algorithm for iterativedetection-decoding MIMO OFDM system,IEEE Conference Publications 2006
[6] T.A. Ramstad S.O. Aase J.H. Husoy "Multiplier Compression of Image:Principles and Examples" Elsevier Science B.V, 1995
[7] Ali N.Akansu, Mark J.T.Smith, "Multiplier wavelet transforms design and application", Kluwer academic publishers. 1996.
[8] Youn-Hong Kim, Chung-Hwa Kim, Kang-Hyeon Rhee "A Design of Discrete Wavelet Transform Encoder for Image Signal Processing", Processings of IEEK Fall Conference '98. pp 1101-1104, Nov 1998.
[9] RaguveerM.Rao, Ajit S. Bopatikar "Wavelet transforms introduction to theory and application, Addison-weswley, 2001.
[10] J.Bhaskar,"AVHDL primer" Pearson education,2004.
[11] A. Danilin, M. Bennebroek and S. Sawitzki, "A novel toolset for the development of FPGA-like reconfigurable logic", in Proc. FPL 2005, pp. 640-643, 2005.
[12] Valadon, C.G.F.; Tafazolli, R.; Evans, B.G. "Performance evaluation of concatenated codes with inner trellis codesandouterReed-Solomoncode" IEEE Transactions rolume: 49 , Issue: 4 2001.
[13] Qingchun Chen; Wai Ho Mow; Pingzhi Fan "Some New Results on Recursive Convolutional Codes and Their Applications " IEEE 2006.
[14] Liao, Xin; Ilow, Jacek; Al-Shaikhi, Al "Trellis Termination in Turbo Codes with Full Feedback RSCEncoders" 6th International ITG-Conference on Source and Channel Coding , IEEE 2006.