INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 6, Issue 1 JAN 2017

Flexible Computational Unit using Carry Save Arithmetic and Dadda Multiplier

Mr. R. John Marion[1] Dr. G. Shanmuga Priya[2] Mrs. S.Sri Bindu[3] Dr. M. Gurunadha Babu[4]
rjohnmarion@yahoo.com[1], spriyagsn@yahoo.com[2] sribindu.sattu@gmail.com[3]
[1]PG Scholar, Dept of ECE, CMR Institute of Technology, Kandlakoya, Medchal, Ranga Reddy, Telangana.
[2]Professor, Dept of ECE, CMRIT, Kandlakoya, Medchal, Ranga Reddy, Telangana.
[3]Associate Professor, Dept of ECE, CMRIT, Kandlakoya, Medchal, Ranga Reddy,Telangana.
[4]Professor, HOD, Dept of ECE, CMRIT, Kandlakoya, Medchal, Ranga Reddy, Telangana.

Abstract: This paper presents a novel approach to design Flexible computational unit (FCU) exploiting carry save arithmetic and dadda multiplier. Earlier projects emphasized on designing systems with low power and fast operations resulting in increased area. The goal of this project is to design DSP system to make the design more efficient by proposing Dadda tree multiplier. Utilizing Dadda tree multiplier reduces the area required for the advisement. As in these modern times, most of the electronic systems are adapted to perform digital signal processing; the DSP integrated systems have wide range of applications. Kernel which is the core of the DSP system consists of data flow graph that features the corresponding arithmetic operation performed by the system. These DFGs basically consist of arithmetic units such as adders, multipliers etc that are mapped onto the proposed flexible control units which are carry save formatted data. As we know that the basic intension of designing integrated circuits is to minimize the dimensions of a given circuit. This objective is achieved by incorporating dadda multiplier as multiplier unit of the DSP data flow graph template. This implementation shows considerable difference in terms of area which further leads to other advantages such as to fabricate more number of components onto the resulted area.

Index Terms: Digital signal processing (DSP), Data flow graph (DFG), Register Transfer logic (RTL), Template(T), Spurious power suppression technique (SPST), Carry save arithmetic (CSA).

I. Introduction

Digital signal processing (DSP) play a very vital role in our daily life.Digital signal processing techniques are increasingly replacing conventional analog signal processing methods in many fields. DSP systems have wide range of applications such as Radar, Multi-rate processing and non-linear DSP, Analogue emulation systems, Networking, Audio/video/multimedia, Noise cancellation systems, Biomedical, Non-destructive testing, Pattern recognition and matching, Control systems engineering, Digital waveform synthesis, remote sensing, Earth-based telecommunications, Robotics, Image processing, Satellite telemetry, Image processing in all its representations, Seismology, Industrial signal conditioning, Speech recognition/ synthesis, Mechatronics, Scientific instrumentation, Signal analysis, Military/surveillance, Multiplexing etc. DSP system is the use of digital processing, such as by computers, to perform a wide variety of signal processing operations. The signals processed in this manner are a sequence of numbers that represent samples of a continuous variable in a domain such as time, space, or frequency. A Digital signal processing (DSP) is a specialized microprocessor (or a SIP block), with its architecture optimized for the

operational needs of Digital signal processing (DSP). This project projects the evolution of current project from preceding projects that are existing and proposed project. The Existing project mainly dealt with implementation of SPST adders in adder unit of the DFG of the given DSP data flow graph resulting in speed of the system. The proposed system incorporated carry save adder in the data flow graph instead of the SPST adder resulting in further increase in the overall speed of the system but compromising for the area i.e. increase in the dimensions of the system. Our project which is its extension incorporates dadda multiplier reduction in area which is remarkably advantageous especially for integrated circuits.

II. Flexible Computational Unit

Adaptable data paths have been proposed to efficiently scale chained operations found in the initial data-flow graph (DFG) of a kernel of the DSP system. The templates of complex chained operations are either extracted directly from the kernel's DFG or specified in a ready-made behavioral template library. The stated architecture comprises flexible computational units (FCUs), which enable the execution of a large set of operation templates found in DSP kernels. The proposed accelerator architecture delivers average gains in area-delay product and in energy consumption compared to state-of-art flexible data paths, sustaining efficiency toward scaled technologies.

The proposed flexible accelerator architecture is shown in Fig. 1. Each FCU operates directly on operands and produces data in the same form for direct reuse of intermediate results. Each FCU operates on 16-bit operands. Such a bit-length is adequate for the most DSP datapaths, but the architectural concept of the FCU can be straightforwardly adapted for smaller or larger bit-lengths. The number of FCUs is determined at design time based on the instruction-level parallelism (ILP) and area constraints imposed by the designer. The register bank consists of scratch registers and is used for storing intermediate results and sharing operands among the FCUs. Different DSP kernels (i.e., different register allocation and data communication patterns per kernel) can be mapped onto the proposed architecture using post-RTL datapath interconnection sharing techniques.

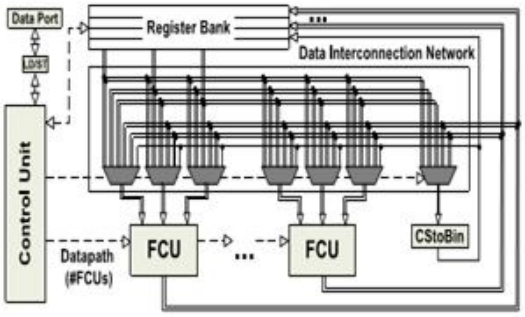


Fig.1: Flexible accelerator architecture.

The control unit drives the entire accelerator architecture (i.e., communication between the data port and the register bank, configuration words of the FCUs and selection signals for the multiplexers) in each clock cycle.

Data Path

The structure of the FCU (Fig. 2) has been designed to enable high-performance flexible operation chaining based on a library of operation templates. Each FCU can be configured to any of the T1–T5 operation templates in template library. The proposed FCU enables intra-template operation chaining by fusing the additions performed before/after the multiplication &performs any partial operation template of the following complex operations:

$$W = A \times (X + Y) + K \quad (1)$$

$$W = A \times K + (X + Y) \quad (2)$$

The above specified equations can be implemented simultaneously in a operational template found in the FCU as shown in figure 2.

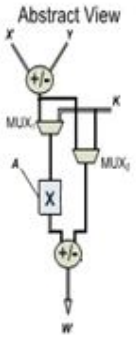


Fig.2: shows a segment of the internal structure of the FCU i.e. operational template of the data path specified in equation (1) and (2)

III. Existing Project

The existing project implemented adders using spurious power suppression technique as substitute for adders and Modified Booth Encoder for multipliers found in the datapath of the FCUs in the system kernels. Figure 5 shows implementation of SPSTadders in the internal structure of FCU. But the serious drawback of this project was that there was tradeoff between power and speed resulting in time consumption.

FCU with Spurious Power Suppression Technique (SPST) Implementation

Minimized scaling integrated circuits and their related researchers are still finding solutions for further minimization. Thereduced power, thermal dissipation is the goal for the VLSI researchers. There are different methods used for the minimized sources with efficient performance. Here a method is discussed for the reduction of the power. The power dissipation in the integrated circuits is more because of the dynamic power dissipation. This is mainly because of the charging and discharging of the capacitance in the circuit.

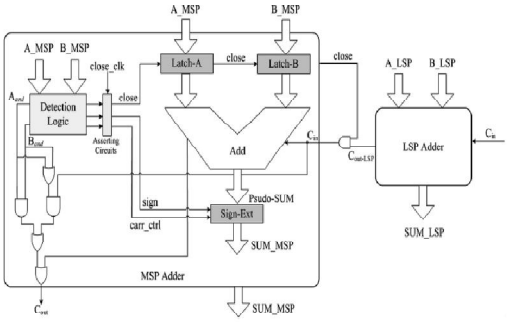


Fig.3: Block diagram of adder implementing spurious power suppression technique.

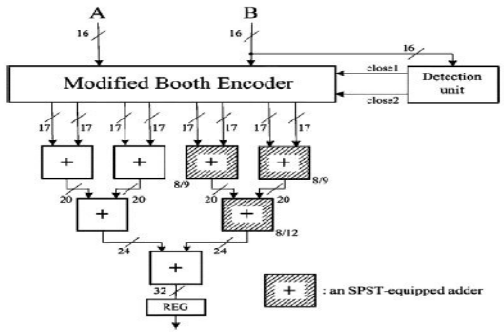


Fig. 4: Block diagram of SPST multiplier

By reducing the capacitance, clock frequency, power supply, switching activity we can reduce the power consumption of the integrated circuits. Here by reducing the partial products generated in the repeated addition of the multipliers we can reduce half of the power consumption when compared to the other multipliers. Multipliers with high speed are essential for digital applications. The multiplier is basically added repeatedly for the value of the multiplicand times. This is a long time process. This method follows avoiding accumulation of partial products generated in the multiplication process. Hence this method is known as spurious power suppression technique. Figure 5 shows implementation of SPST adders in the internal structure of the FCU.

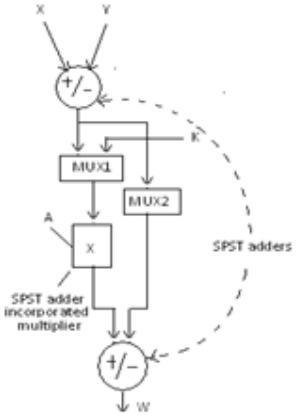
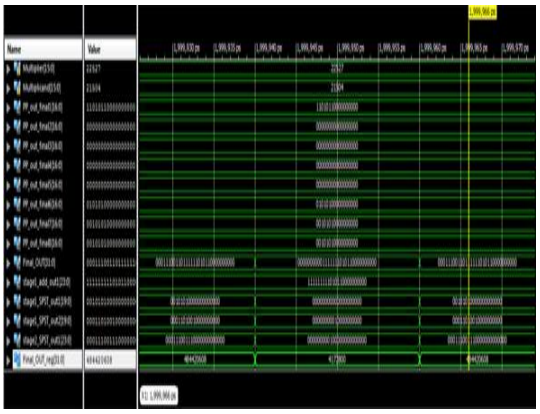


Fig. 5: Implementation of SPST adders in the internal structure of FCU.

Simulation Results:

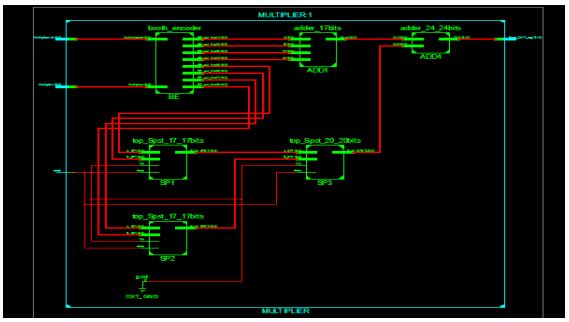
The written Verilog HDL code is simulated by using ISIM simulator and results are shown in below fig.



Synthesis Results:

The written Verilog HDL code is synthesized by using Xilinx ISE and results are shown in below fig.

RTL Schematic:



Design Summary:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	591	4656	12%
Number of 4 input LUTs	1091	9312	11%
Number of bonded IOBs	65	232	28%

Timing Report:

```
MUXCY:CI->O          1   0.051   0.000  SP3/ADD_MSP1/n5/Madd_AUX_3_addsub0000(
MUXCY:CI->O          1   0.051   0.000  SP3/ADD_MSP1/n5/Madd_AUX_3_addsub0000(
MUXCY:CI->O          1   0.051   0.000  SP3/ADD_MSP1/n5/Madd_AUX_3_addsub0000(
XORCY:CI->O          1   0.699   0.426  SP3/ADD_MSP1/n5/Madd_AUX_3_addsub0000(
LUT4:I1->O           1   0.612   0.426  SP3/ADD_MSP1/n6/sum_MSP_6_mux00001 (s
LUT2:I1->O           1   0.612   0.000  ADD4/Madd_OUT_lut<30> (ADD4/Madd_OUT_
MUXCY:S->O           0   0.404   0.000  ADD4/Madd_OUT_cy<30> (ADD4/Madd_OUT_c
XORCY:CI->O          1   0.699   0.357  ADD4/Madd_OUT_xor<31> (Final_OUT_reg_
OBUF:I->O                3.169          Final_OUT_reg_31_OBUF (Final_OUT_reg<
----------------------------------------
Total                   35.961ns (24.788ns logic, 11.173ns route)
                                 (68.9% logic, 31.1% route)
```

IV. Proposed Project

In the Proposed project, the main aim was to exploit the time saving characteristic of the carry save technique i.e. by avoiding long carry propagation. The drawback in speed in the existing project led to the incorporation of carry save adder instead of SPST adder. This fast propagation in carry save technique results in fast operation. But this turns out to with increase area of the template.

FCU With Carry Save Arithmetic Implementation

CS representation has been widely used to design fast arithmetic circuits due to its

inherent advantage of eliminating the large carry-propagation chains. CS arithmetic optimizations rearrange the application's DFG and reveal multiple input additive operations (i.e., chained additions in the initial DFG), which can be mapped onto CScompressors. The goal is to maximize the range that a CS computation is performed within the DFG. However, whenever a multiplication node is interleaved in the DFG, either a CS to binary conversion is invoked or the DFG is transformed using the distributive property . Thus, the aforementioned CS optimization approaches have limited impact on DFGs dominated by multiplications, e.g., filtering DSP applications.

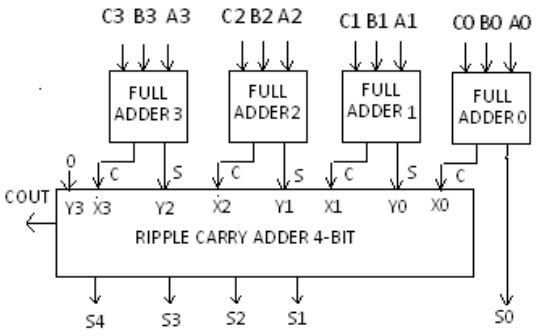


Fig.6 : A 4-bit carry save adder using four full adders and a 4-bit ripple carry adder.

In this brief, we tackle the aforementioned limitation by exploiting the CS to modified Booth (MB) recoding each time a multiplication needs to be performed within a CS-optimized datapath. Thus, the computations throughout the multiplications are processed using CS arithmetic and the operations in the targeted datapath are carried out without using any intermediate carry-propagate adder for CS to binary conversion, thus improving performance.

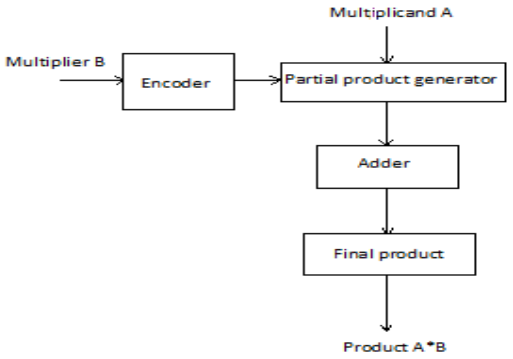


Fig.7: Block diagram of Modified booth multiplier

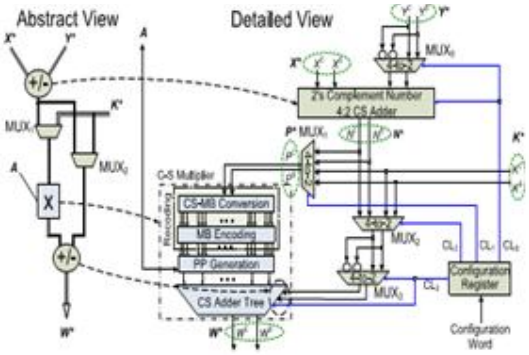
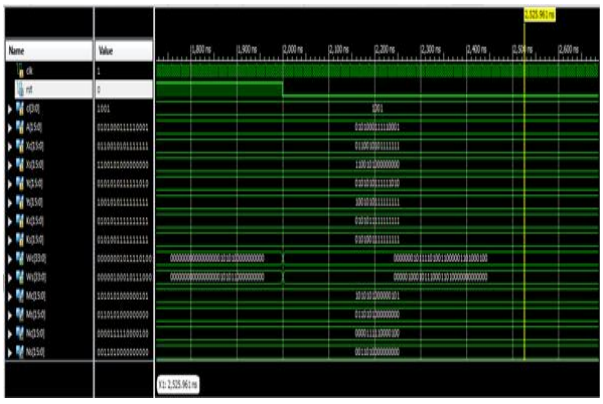


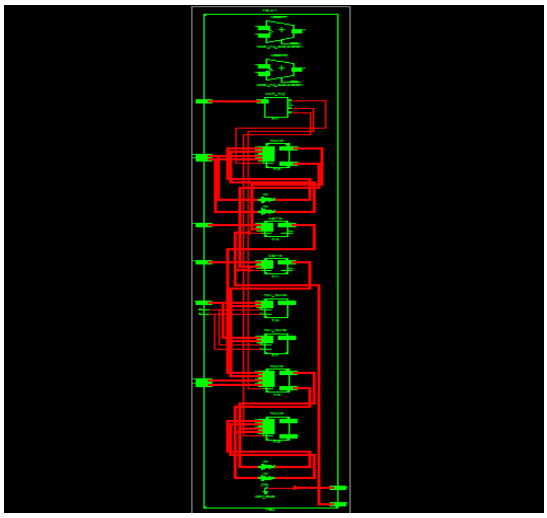
Fig. 8: Implementation of carry-save technique for adders and modified booth technique for multiplier in FCU.

Simulation Results:



Synthesis Results:

RTL Schematic:



Design Summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	824	4656	17%
Number of Slice Flip Flops	128	9312	1%
Number of 4 input LUTs	1472	9312	15%
Number of bonded IOBs	185	232	79%
Number of GCLKs	1	24	4%

Timing Report:

```
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<24> (Madd_Ws_
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<25> (Madd_Ws_
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<26> (Madd_Ws_
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<27> (Madd_Ws_
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<28> (Madd_Ws_
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<29> (Madd_Ws_
    MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<30> (Madd_Ws_
    XORCY:CI->O           1   0.699   0.357  Madd_Ws_addsub0000_xor<31> (Ws_31_OB
    OBUF:I->O                 3.169          Ws_31_OBUF (Ws<31>)
    ----------------------------------------
    Total                    18.934ns (12.483ns logic, 6.451ns route)
                                      (65.9% logic, 34.1% route)
```

V. Extension

Since area parameter is a major concern in the integrated circuits, so our extended project exploits dadda multiplier for multiplier in FCU. The internal operation in the dadda multiplier includes adders exploiting carry save arithmetic. In our project as stated earlier, the dadda multiplier is implemented in the multiplier unit of the operational template found in the data flow graph of the kernel of the DSP system.

FCU with Carry save Arithmetic and Dadda Multiplier Implementation

The Dadda multiplier is a hardware multiplier design invented by computer scientist Luigi Dadda in 1965. It issimilar to the Wallace multiplier, but it is slightly faster (for alloperand sizes) and requires fewer gates (for all but the smallest operand sizes). In fact, Dadda and Wallace multipliers have the same 3 steps: Firstly, multiply (logical AND) each bit of one of the arguments, by each bit of the other, yielding (n*n) results. Depending on the position of the multiplied bits, the wires carry different weights. Secondly, reduce the number of partial producted to two by layers of full and half adders. Thirdly, group the wires in two numbers and add them with a conventional adder. Fig 9 demonstrates the reduction of partial products in a dadda multiplier.

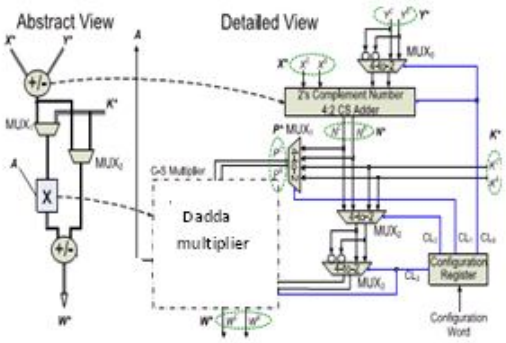


Fig. 2. FCU.

Fig.9: Implementation of carry-save technique for adders and dadda technique for multiplier in FCU.

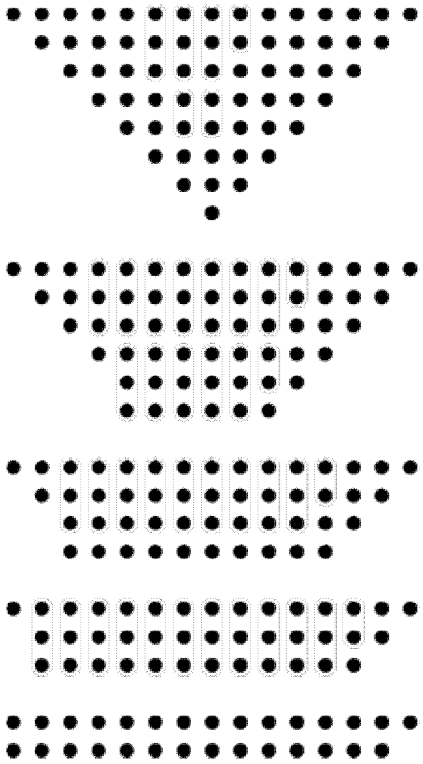
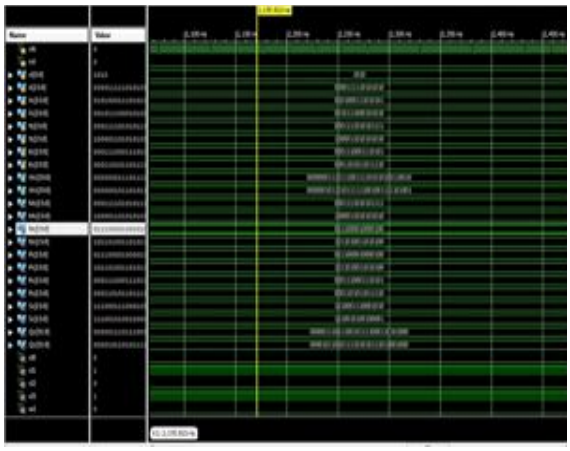


Fig.10: Example of Dadda reduction on 8x8 multiplier. Bits with lower weight are rightmost.

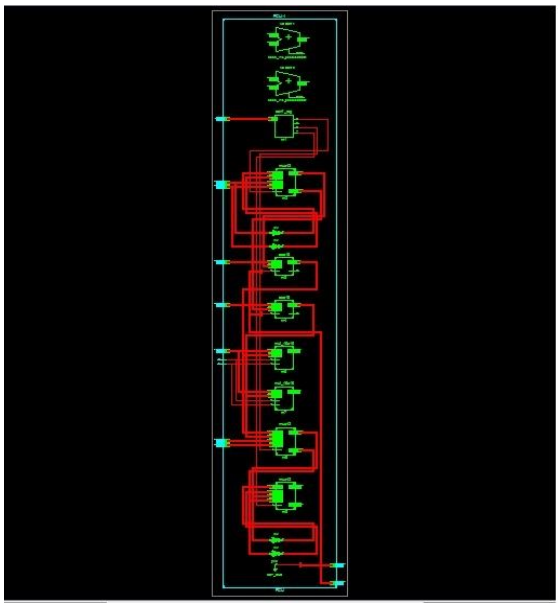
VI. Results

Simulation Results:



Synthesis Results:

RTL Schematic:



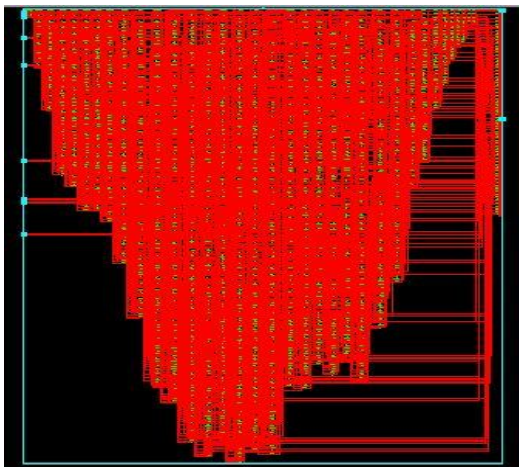
Design Summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	824	8672	9%
Number of Slice Flip Flops	128	17344	0%
Number of 4 input LUTs	1472	17344	8%
Number of bonded IOBs	185	190	97%
Number of GCLKs	1	24	4%

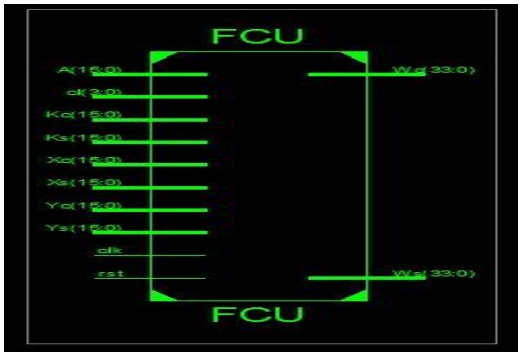
Timing Report:

```
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<18
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<19
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<20
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<21
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<22
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<23
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<24
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<25
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<26
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<27
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<28
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<29
MUXCY:CI->O           1   0.051   0.000  Madd_Ws_addsub0000_cy<30
XORCY:CI->O           1   0.699   0.357  Madd_Ws_addsub0000_xor<3
OBUF:I->O                 3.169          Ws_31_OBUF (Ws<31>)
----------------------------------------
Total                    18.934ns (12.483ns logic, 6.451ns route)
                                  (65.9% logic, 34.1% route)
```

Technology schematic



Top schematic



VII. Conclusion

Concisely, we introduced an FCU architecture that exploits the incorporation of CS arithmetic and dadda algorithmic optimizations to enable fast chaining of additive and multiplicative operations. This flexible accelerator architecture allow to operate on both conventional two's complement and CS-formatted data operands, thus enabling high degrees of computational density to be achieved. Theoretical and experimental analyses have shown that the extended solution forms an efficient design delivering considerable amount in terms of area.

VIII. Future Aspect

Flexible accelerator architecture is able to operate on both conventional two's complement and CS-formatted data. Flexible accelerator architecture that exploits the incorporation of CS arithmetic optimizations enable fast chaining of additive and multiplicative operations. These operations are performed on less number bit length inputs. These can further increase to more number of bits i.e, upto 64-bits.

REFERENCES

[1] T. Kim and J. Um, "A practical approach to the synthesis of arithmetic circuits using carry-save-adders," IEEE Trans. Comput.- Aided Design Integr. Circuits Syst., vol. 19, no. 5, pp. 615–624, May 2000.

[2] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in Proc. 13th Int. Conf. Field Program. Logic Appl., vol. 2778. 2003, pp. 61–70.

[3] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems," J. Supercomput., vol. 26, no. 3, pp. 283–308, 2003.

[4] A. Hosangadi, F. Fallah, and R. Kastner, "Optimizing high speed arithmetic circuits using three-term extraction," in Proc. Design, Autom. Test Eur. (DATE), vol. 1. Mar. 2006, pp. 1–6.

[5] M. D. Galanis, G. Theodoridis, S. Tragoudas, and C. E. Goutis, "A high-performance data path for synthesizing DSP kernels," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 6, pp. 1154–1162, Jun. 2006.

[6] P. Ienne and R. Leupers, Customizable Embedded Processors:Design Technologies and Applications. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[7] K. Compton and S. Hauck, "Automatic design of reconfigurable domainspecific flexible cores," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 5, pp. 493–503, May 2008.

[8] A. K. Verma, P. Brisk, and P. Ienne, "Data-flow transformations to maximize the use of carry-save representation in arithmetic circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 27, no. 10, pp. 1761–1774, Oct. 2008.
[9] S. Xydis, G. Economakos, and K. Pekmestzi, "Designing coarse-grain reconfigurable architectures by inlining flexibility into custom arithmetic data-paths,"Integr., VLSI J., vol. 42, no. 4, pp. 486–503, Sep. 2009.
[10] G. Ansaloni, P. Bonzini, and L. Pozzi, "EGRA: A coarse grained reconfigurable architectural template," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 6, pp. 1062–1074, Jun. 2011.
[11] S. Xydis, G. Economakos, D. Soudris, and K. Pekmestzi, "High performance and area efficient flexible DSP datapath synthesis," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 3, pp. 429–442, Mar. 2011.
[12] M. Stojilovic, D. Novo, L. Saranovac, P. Brisk, and P. Ienne, "Selective flexibility: Creating domain-specific reconfigurable arrays," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 5, pp. 681–694, May 2013.