IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
National Conference on Emerging Trends in Engineering and Technology (NCETET'16)
Lourdes Matha College of Science & Technology, Thiruvananthapuram
Vol. 3, Special Issue 3, August 2016

Design and Analysis of DSP Accelerator Architecture Exploiting Modified Booth

Linu M Jiji[1], **Ragimol**[2]
M.Tech Student, Department of ECE, Sree Buddha College of Engineering, Alappuzha, Kerala, India[1]
Professor, Department of ECE, Sree Buddha College of Engineering, Alappuzha, Kerala, India[2]

Abstract: Embedded systems often use specialized hardware accelerators to improve performance and reduce energy consumption especially in areas such as signal processing and video processing, communications, and computer vision. Hardware acceleration has been proved as an extremely promising implementation strategy for the digital signal processing (DSP) domain. An accelerator is a hardware module that can be attached to a processor core. It enhances the performance or functionality by executing certain function in the accelerator instead of executing in the processor core. The accelerator module mainly consists of flexible computational units(FCUs). The structure of the flexible computational unit is designed to enable high performance flexible operation chaining based on a set of operation templates found in DSP kernels. The number of flexible computational units is determined at the design time based on the instruction level parallelism and area constraints imposed by the designer. In this work a high performance architectural scheme is designed by combining both the architectural and arithmetic levels of abstraction. The proposed solution forms an efficient design tradeoff of 46% delivering optimized latency/area and energy implementations. It also provides high computing performance, real time processing and power efficiency to variety of applications ranging from sensors to servers. The accelerator module find wide applications in areas such as video encoding and decoding and in several image processing applications where high performance computation is needed.

Keywords: DSP Accelerator, Flexible computational unit, Digital signal processor, Modified Booth, Multiplier.

I. INTRODUCTION

Digital Signal Processors (DSP) are special kind of microprocessors with its architecture optimized for the digital signal processing application. Inorder to accelerate the performance of digital signal processors, DSP accelerators can be used. This kind of accelerators are used by embedded system widely in areas such as signal and video processing, communications, and computer vision to improve performance and reduce energy consumption. Algorithms can be applied for the implementation of accelerators that are always used independently in order to reduce area. This approach can create multioperational data paths for a given set of applications; however it alone does not provide greater flexibility as the user of such system wish to create additional applications. Flexibility can be achieved with the use of different types of operation templates.

The proposed work mainly focus on an efficient implementation of optimized accelerator architecture for digital signal processors to enhance the performance. Most of the operations performed with the help of templates can be executed within the accelerator module without interfering the processor. A template may be defined as a specialized hardware unit or a group of chained unit. A data-flow graph (DFG) is a graph which represents a data dependancies between a number of operations. Any algorithm consists of a number of ordered operations. Inorder to cover a part of data flow graph (DFG) using a given set of templates, the subset of templates that match this part must be identified called as template matching and then the most efficient template must be selected. Template selection has major impact on the performance, and many optimization techniques are used to address the problem.

Multiplier architecture mainly comprises of two architectures, i.e., Modified Booth and Wallace tree. Based on the study of various multiplier architectures, the Modified Booth increases the speed because it reduces partial products to half.

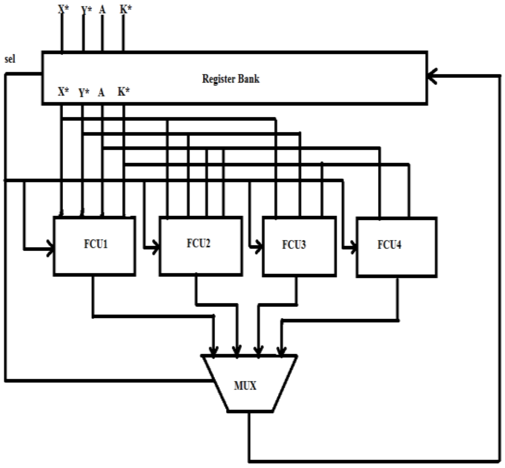


Figure 1 Proposed Optimized Accelerator Architecture for DSP

Further, the delay in multiplier can be reduced by using Wallace tree. Power consumption of Wallace tree multiplier is also less as compared to booth and array. Features of both multipliers can be combined to produce high speed and low power multiplier.

II. FLEXIBLE ACCELERATOR ARCHITECTURE

An optimized accelerator architecture for DSP using modified booth is shown in figure1.The architecture mainly consists of flexible computational units (FCUs). Each FCU operates directly on CS operands and produces data in the same form for the direct use of intermediate result. The number of computational unit is determined at the design time based on instruction level parallelism and area constraints imposed by the designer. Each FCU can be configured to operate based on a set of operation templates. The most suitable FCU is selected with the help of multiplexer. Register bank is mainly used to store the intermediate results and sharing values of operands among the FCUs.

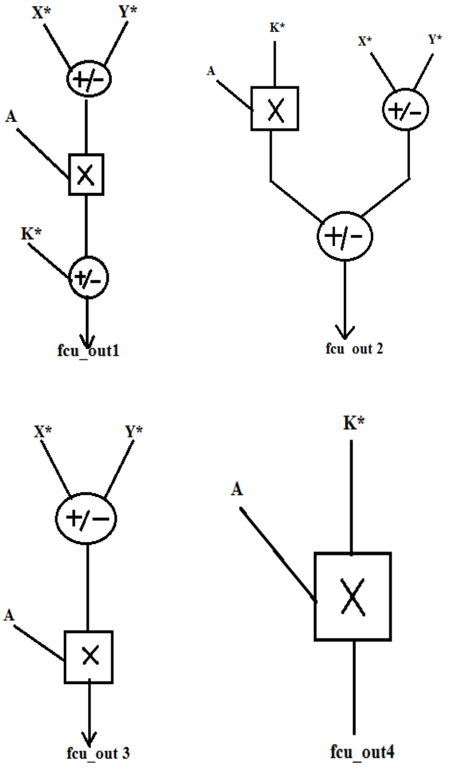


Figure 2 Templates used in the flexible computational unit

The different types of templates are shown in figure2. Each template may consist of an adder/subtractor module and a multiplier section. The multiplier here used is a modified booth multiplier. Booth algorithm for multiplication is a simple method in which multiplication is carried out with repeated addition operation. To overcome the main limitations of booth algorithm modified booth algorithm is used.

III.MODIFIED BOOTH ENCODED MULTIPLIER

Large numbers of arithmetic operation are used in many digital signal processing (DSP) platform. The multiplier reduces within power and area and plays significant role in high performance of any digital indication processing system. So whenever an addition operation followed by multiplication operation is required to be performed fused Add-Multiply (FAM) operator is used

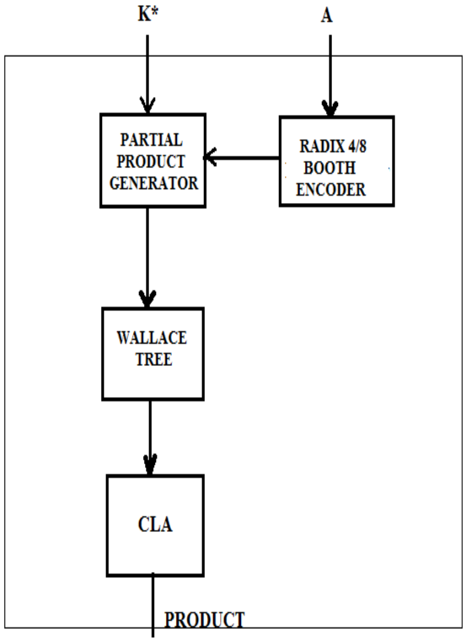


Figure 3 Modified Booth Encoded Multiplier

This implements a new technique by straight recoding of sum two numbers in Modified Booth (MB) form. The new technique is implemented by three new dissimilar schemes by integrating them within the existing FAM plans. The performance of the proposed schemes gives reduction in conditions of critical delay, hardware complication and power utilization while comparing with the existing AM design.

Modified booth multiplier is used to perform high speed multiplication using modified booth algorithm. The main advantage is that we can reduce the number of partial products to half. The detailed diagram of multiplier is shown in figure 3.The Radix8 booth encoder within the multiplier performs the process of encoding the multiplicand based on the multiplier bits.

It will compare three bits at a time based on the overlapping technique. For the purpose of generation of the partial products partial product generator is used. For large multipliers the performance of the modified booth algorithm is limited. For that purpose booth encoding together with the Wallace tree structure can be used.

Wallace tree adders are used in high speed designs to produce two rows of partial products that can be added in the final stage. Critical path and the number of adders are reduced as compared to the parallel adders. The speed, area and the power consumption of the multipliers are directly proportional to the efficiency of the compressors.

A carry save adder is a type of digital adder used in the computer architecture to compute the sum of three or more n-bit numbers in binary. It differs from other digital adders in the sense that it outputs two numbers of the same dimension as that of the inputs, one which is a sequence of the sum bit and the other which is a sequence of carry bits.

This kind of carry save adder trees are used for high speed implementation of multiple operand addition. The adder/ subtractor module is mainly designed with the help of carry select adders instead of using the ripple carry adders. Thus the carry propagation in the ripple carry adder can be completely overcome by using the carry select adder. This will significantly reduces the delay and enhances the performance.

IV. EXPERIMENTAL RESULTS

The model is simulated using Xilinx ISE Design Suite 13.2. Figure 5 shows the simulation result for multiplier. Figure 6 shows the simulation result for 16bit carry look ahead adder. Figure 7,8,9,10 shows the simulation result for different flexible computational units.

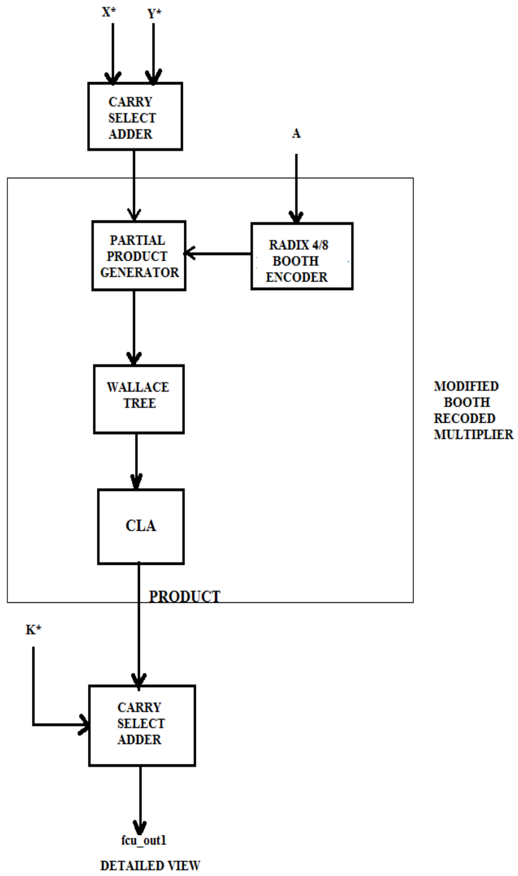
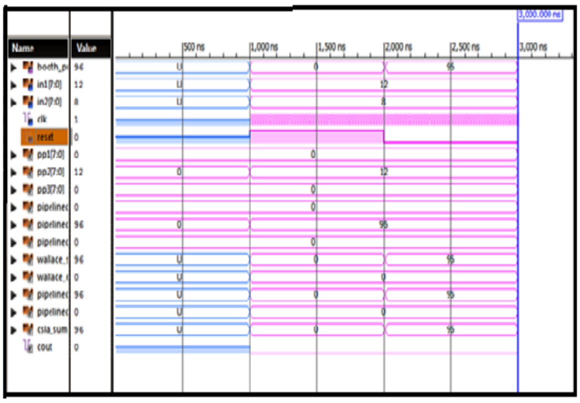


Figure 4 Detailed view of template1

MODIFIED BOOTH ENCODED MULTIPLIER



Name	Value
booth_p	96
in1[7:0]	12
in2[7:0]	8
clk	1
reset	0
pp1[7:0]	0
pp2[7:0]	12
pp3[7:0]	0
pipelined	0
pipelined	96
pipelined	0
wallace_s	96
wallace_c	0
pipelined	96
pipelined	0
csla_sum	96
cout	0

Figure 5 Simulation result for modified boothencoded multiplier

CARRY SELECT ADDER

Name	Value
in1[7:0]	11111111
in2[7:0]	11111111
sum[7:0]	11111110
cout	1
c0	1
c1	1
c2	1
rc_out2[1:0]	10
rc_out4[3:0]	1110
bec_in3[2:0]	110
bec_out3[2:0]	111
bec_in5[4:0]	11110
bec_out5[4:0]	11111
mux_out3[2:0]	111
mux_out5[4:0]	11111

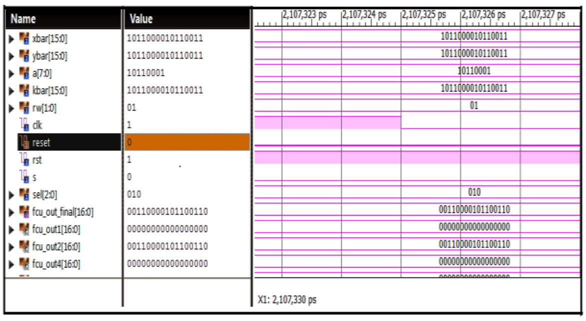
Figure 6 Simulation result for carry select adder

FLEXIBLE COMPUTATIONAL UNIT 1

Name	Value
xbar[15:0]	0000000000001010
ybar[15:0]	0000000000001010
a[7:0]	00000101
kbar[15:0]	0000000000001010
rw[1:0]	10
clk	0
reset	0
rst	0
s	0
sel[2:0]	001
fcu_out_final[16:0]	00000000001101110
fcu_out1[16:0]	00000000001101110
fcu_out2[16:0]	00000000000000000
fcu_out4[16:0]	00000000000000000

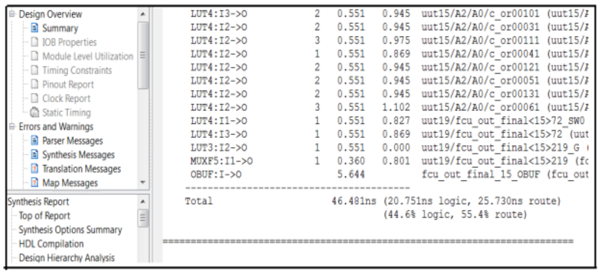
Figure 7 Simulation result for flexible computational unit 1

FLEXIBLE COMPUTATIONAL UNIT 2



Name	Value
xbar[15:0]	1011000010110011
ybar[15:0]	1011000010110011
a[7:0]	10110001
kbar[15:0]	1011000010110011
rw[1:0]	01
clk	1
reset	0
rst	1
s	0
sel[2:0]	010
fcu_out_final[16:0]	00110000101100110
fcu_out1[16:0]	00000000000000000
fcu_out2[16:0]	00110000101100110
fcu_out4[16:0]	00000000000000000

Figure 8 Simulation result for flexible computational unit 2



```
LUT4:I3->O           2   0.551   0.945  uut15/A2/A0/c_or00101 (uut15/
LUT4:I2->O           2   0.551   0.945  uut15/A2/A0/c_or00031 (uut15/
LUT4:I2->O           3   0.551   0.975  uut15/A2/A0/c_or00111 (uut15/
LUT4:I2->O           1   0.551   0.869  uut15/A2/A0/c_or00041 (uut15/
LUT4:I2->O           2   0.551   0.945  uut15/A2/A0/c_or00121 (uut15/
LUT4:I2->O           2   0.551   0.945  uut15/A2/A0/c_or00051 (uut15/
LUT4:I2->O           2   0.551   0.945  uut15/A2/A0/c_or00131 (uut15/
LUT4:I2->O           3   0.551   1.102  uut15/A2/A0/c_or00061 (uut15/
LUT4:I1->O           1   0.551   0.827  uut19/fcu_out_final<15>72_SW0
LUT4:I3->O           1   0.551   0.869  uut19/fcu_out_final<15>72 (uut
LUT3:I2->O           1   0.551   0.000  uut19/fcu_out_final<15>219_G (
MUXF5:I1->O          1   0.360   0.801  uut19/fcu_out_final<15>219 (fc
OBUF:I->O                5.644          fcu_out_final_15_OBUF (fcu_out
----------------------------------------
Total                   46.481ns (20.751ns logic, 25.730ns route)
                                 (44.6% logic, 55.4% route)


===================================================================
```

Figure 11 Total delay required for implementation.

V. CONCLUSION

In brief, an optimized accelerator architecture for DSP that exploits the merits of carry save arithmetic in case of multiplication and fast addition operations to enable fast chaining of additive and multiplicative operations is introduced. From the result analysis it is clear that the delay can be reduced by optimizing the architecture and incorporating the modified booth. The proposed flexible accelerator architecture is able to operate on both conventional two's complement and CS-formatted data operands, thus enabling high degrees of computational density to be achieved. The accelerator architecture implementation makes the operation of digital signal processors much faster. Experimental analyses have shown that the proposed solution forms an efficient design tradeoff point delivering optimized latency/area and energy implementations.

ACKNOWLEDGMENT

This journey of self actualization would not have been fulfilled without the guidance and support of several individuals. My deep appreciation is extended to my Thesis Guide Prof. Ragimol, Department of Electronics and communication for her guidance, encouragement and appreciation.

REFERENCES

[1] Kostas Tsoumanis, Sotirios Xydis, Georgios Zervakis, and Kiamal Pekmestzi, "Flexible DSP Accelerator Architecture Exploiting Carry-Save Arithmetic," IEEE Trans.Very Large Scale Integr. (VLSI) Syst., 2015
[2] 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
[3] Andrea Lodi, Mario Toma, Fabio Campi, Andrea Cappelli, Roberto Canegallo, and Roberto Guerrieri, "A VLIW Processor With Reconfigurable Instruction Set for Embedded Applications," Ieee Journal Of Solid-state Circuits, Vol. 38, No. 11, November 2003
[4] Hadi Parandeh-Afshar, Ajay Kumar Verma, Philip Brisk, and Paolo Ienne, "Improving FPGA Performance for Carry-Save Arithmetic," IEEE Trans.Very Large Scale Integr. (VLSI) Syst., Vol. 18, No. 4, April 2010
[5] Peter A. Milder, Franz Franchetti,Formal Datapath Representation and Manipulation for Implementing DSP Transforms June 2008
[6] Giovanni Ansaloni, Paolo Bonzini and Laura Pozzi, Member, IEEE, "EGRA: A Coarse Grained Reconfigurable Architectural Template" IEEE Trans. Very Large Scale Integr(VLSI) Systems, Vol. 19, No. 6, June 2011.