*International Journal of Latest Engineering and Management Research (IJLEMR)*
*ISSN: 2455-4847*
*www.ijlemr.com ǁ Volume 02 - Issue 01 ǁ January 2017 ǁ PP. 06-14*

# Implementation of Parallel Multiplier-Accumulator using Radix-2 Modified Booth Algorithm and SPST

Ms. Deepali P. Sukhdeve
*Assistant Professor*
*Department of Electronics Engineering*
*Priyadarshini College of Engineering*
*Nagpur, India*

**Abstract:** In this paper we analyze a multiplier-and-accumulator unit (MAC) for high speed and less power consumption. Radix-2 Modified Booth Algorithm is used which reduces the partial products and improves speed. Carry Save Adder adds the 8 partial products and generates final three intermediate operands. Two of them are given as inputs for final addition that produces output of MAC unit from MSB 16-bit Result. Pipelining scheme is used which helps to improve the overall performance. Thus improving the speed. For reducing the power consumption, we need to reduce the glitches and spikes. The adder designed using Spurious Power Suppression Technique is used to remove the glitches and spikes. This SPST adder is applied to the modified Booth Encoder and the performance of multiplier-and-accumulator unit designed using CSA is compared with the multiplier-and accumulator unit designed using Spurious Power Suppression Technique (SPST).Both the architectures are synthesized using Xylinx ISE. The SPST MAC gives the superior results as compared to the MAC designed using CSA in terms of area, delay and speed and hence improves the overall performance.

**Index Terms:** Carry Save Adder (CSA), multiplier-and-accumulator(MAC), digital signal processing (DSP), spurious power suppression technique (SPST), Radix-2 Modified Booth Algorithm.

## I. Introduction

In computing, especially digital signal processing, the multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC, or MAC unit); the operation itself is also often called a MAC or a MAC operation. The MAC operation modifies an accumulator *a*:

$$a \leftarrow a + (b \times c) \qquad (1)$$

When done with floating point numbers, it might be performed with two rounding's (typical in many DSPs), or with a single rounding. When performed with a single rounding, it is called a fused multiply–add or fused multiply–accumulate.Modern computers may contain a dedicated MAC, consisting of a multiplier implemented in combinational logic followed by an adder and an accumulator register that stores the result. The output of the register is fed back to one input of the adder, so that on each clock cycle, the output of the multiplier is added to the register. Combinational multipliers require a large amount of logic, but can compute a product much more quickly than the method of shifting and adding typical of earlier computers. The first processors to be equipped with MAC units were digital signal processors [1], but the technique is now also common in general-purpose processors. Some digital signal processors also use discrete cosine transform (DCT) or discrete wavelet transform.

A multiplier commonly uses an array of Full Adders and Booth's algorithm [2].It also uses Wallace Tree[4] instead of Array of full Adders. Such multipliers consists of Booth Encoder, Wallace tree and Final Adder[2],[4].A new architecture of MAC for high speed arithmetic is been proposed by Young-Ho Seo and Dong-Wook Kim. In this architecture multiplication is combined with accumulation and CSA is devised to improve the performance. The architecture proposed in [2] had a better performance compared to the previous MAC architectures. This architecture proposed in [2] can be adapted to various fields requiring high performance such as the signal processing areas. There is a need to reduce the power consumption due to the unwanted glitches and spikes.

In this paper, a multiplier-and-accumulate (MAC) unit will be designed for high speed and low power consumption. A 16 bit CSA tree is proposed to improve the output rate. The CSA tree uses Radix-2 Modified Booth Algorithm. A Carry Look Ahead Adder can be used to improve speed by reducing the amount of time required to determine carry bits.

In general a multiplier can be divided into three operational steps. The first step includes generation of partial products. The second step includes compression of partial products and the last is the final addition in which the result is produced by adding the sum and the carry. If we need to accumulate the results the operation consists of four steps.

## II. MAC Architecture

**1) MAC Basics**

The Hardware architecture of MAC executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y. This is added to the previous multiplication result Z as the accumulation step.

If N-bit data are multiplied, the number of the generated partial products is proportional to N. In order to add them serially, the execution time is also proportional to N. The architecture of a multiplier, which is the fastest, uses radix-2 Booth encoding that generates partial products and a Wallace tree based on CSA as the adder array to add the partial products [4]. The n–bit MAC inputs X and Y, are converted into an (n+1)-bit partial product by passing through the Booth encoder. In the CSA and accumulator, accumulation is carried out along with the addition of the partial products. As a result, n-bit S,C and Z(the result from adding the lower bits of the sum and carry) are generated. These three values are fed back and used for the next accumulation. If the final result for the MAC is needed, is generated by adding S and C in the final adder and combined with P[n-1:0].

The proposed MAC takes in two 16-bit operands: the multiplier (MR) and the multiplicand (MD), and produces the 32-bit MAC result of the two at its output.

**2) Equation for MAC:**

The expression of MAC will be expressed as follows.

$$P = \left(d_0 2Y + \sum_{i=0}^{N-1} z_i 2^i\right) + \left(\sum_{i=1}^{N/2-1} d_i 2^{2i} Y + \sum_{i=0}^{N-2} c_i 2^i 2^N\right) + \left(d_{N/2-1} 2^{N-2} Y + \sum_{i=0}^{N-2} s_i 2^i 2^N\right). \quad (2)$$

The first parenthesis corresponds to the operation to accumulate the first partial product with the added result of the sum and the carry. The second parenthesis corresponds to the operation to accumulate the middle partial products with the sum of the CSA that was fed back and the third parenthesis expresses the operation to accumulate the last partial product with carry of CSA.

**3) Proposed Architecture**

**(A) MAC UNIT USING CSA**

The architecture of the multiplier primarily consists of five major modules as shown in figure 1. They are:

- 2's Complement Generator,
- Booth Encoder,
- Partial Product Generator,
- Carry Save Adder and Carry Look-ahead Adder.
- Final adder

The multiplier can be constructed in its simplest conceptual form. Original Booth's Algorithm (Radix 2) can be used for encoding for the Booth Encoder. The 2's Complement Generator uses a Ripple Carry Adder constructed from Full-adder modules. The Partial Product Generator uses two control signals x and z produced by the Booth Encoder and uses these signals to choose from and extend signs of '0', MD or -MD for creating 8 partial products. The 8 partial products are supplied to Carry Save Adder and added appropriately. The Carry Save Adder uses both Full Adder and Half Adders and Carry Look Ahead to get the intermediate Result. The final 32 – bit result is given to 16- bit is Accumulator and 16 – bit of Sum and Carry Output of Carry Save Adder is given to input Final Adder which consist of only Half Adder And Full Adder to produce the MSB results to the Product.

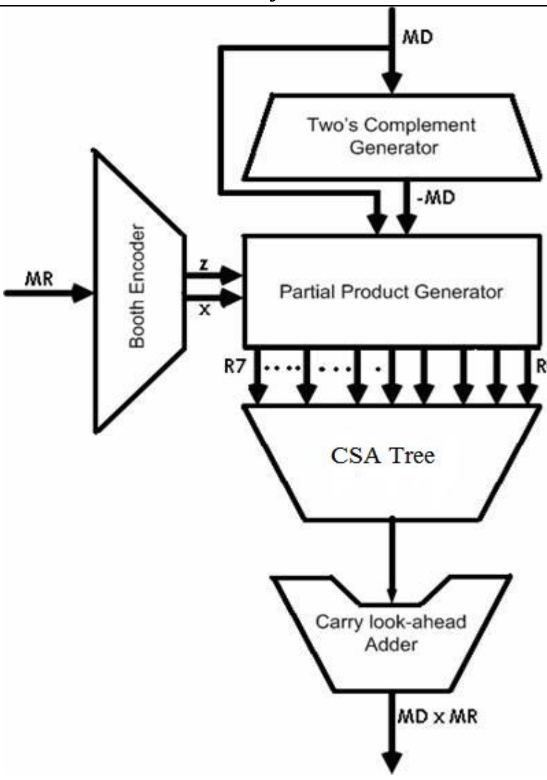

Fig1. Block Diagram of the  Booth Multiplier Architecture

The proposed CSA architecture requires 8 rows of FAs for 8 partial products. Totally 9 rows of full adders are necessary since one more level of rows are needed for accumulation. The critical path is determined by 2-bit CLA. Pipelining is used to increase the output rate [2].

**(B) IMPLEMENTATION OF MAC UNIT USING CSA**

Multiplier-and-Accumulator unit using CSA is implemented and analyzed and then compared with the previous researches. The implementation result will be compared with the standard designs.

**(a) Analysis of Hardware Resources:** The architecture is analyzed and hardware resources are calculated experimentally and then compared with the previous results. The results are shown in Table I.

Table I: Synthesis report

| **Device Parameters** | **MAC with CSA (Spartan 3E)** | **Spartan6** | **Vertex 5** |
|---|---|---|---|
| Number of Slices | 457 out of 4656 | 79 out of 30064 | 79 out of 12480 |
| Number of Slice Flip-Flops | 79 out of 9312 | 832 out of 15032 | 810 out of 12480 |
| Number of 4 input LUT's | 890 out of 9312 | 78 out of 833 | 69 out of 820 |
| Number of IO's | 66 | 66 | 66 |
| Number of BONDED IOB's | 66 out of 190 | 66 out of 190 | 66 out of 172 |
| Number of GCLK's | 1 out of 24 | 1 out of 16 | 1 out of 32 |

**(b) Delay Analysis:** The maximum combinational path delay is analyzed in various devices and families and the results are shown in table II. We compared the results with that of  [2]. The delay of ours was 37.959ns  while in [2] it was 57.2ns, which means ours improved the speed performance. This improvement is due to the final adder. Our architecture uses (n-2) bit final adder, which causes speed improvement.

Table II : Delay Time Analysis in various Devices and Families

| **Device** | **CSA MAC (Spartan 3E)** | **CSA MAC (Spartan 6)** | **CSA MAC (Vertex 5)** |
|---|---|---|---|
| Maximum Combinational Path Delay | 37.959ns | 21.144ns | 14.189ns |

**(c) Pipelining Stage Analysis:** Pipelining scheme is incorporated to increase the operation speed[2]. The pipelining stages were determined based on the delay modeling. In our architecture the accumulation is carried out by feeding back the final CSA outputs rather than the final adder results and shows the result in every clock cycle.

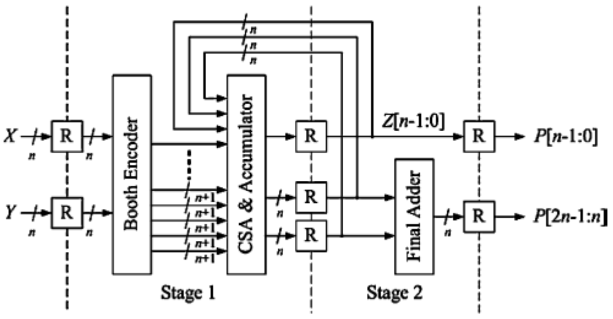

Fig2.Pipelined Hardware Structure [2]

**(d) Power Analysis:** After synthesizing in Xylinx the X-Power analysis has been carried out for our architecture in various devices and families and the results obtained are as shown in table III.

Table III : Power Analysis

| **Mutiplier Type** | **Multiplier-and-Accumulator using CSA** | | |
|---|---|---|---|
| **Vendor** | Xilinx | Xilinx | Xilinx |
| **Device and Family** | Spartan 3E (Xc3s500eft256-4) | Spartan6 (Xc6slx25TcsG324-4) | Vertex 5 (Xc5elx20Tff323-2). |
| **Power Consumption** | 0.081 W | 0.029 W | 0.322 W |

High speed and low power MAC units are required for applications of digital signal processing like Fast Fourier Transform, Finite Impulse Response filters, convolution etc. For improving the speed and reducing the dynamic power, there is a need to reduce the glitches and spikes. Adder designed using spurious power suppression technique (SPST) avoids the unwanted glitches and spikes, thus minimizing the switching power dissipation and hence the dynamic power. Radix -2 modified booth algorithm reduces the number of partial products to half by grouping of bits from the multiplier term, which improves the speed.

**(C) MAC UNIT USING SPST**

The low power multiplier with SPST consists of

1) Modified Booth Encoder,
2) Detection Unit, and
3) Register [5] as shown in figure 3.

Radix-2 modified booth MAC with SPST performs both multiplication and accumulation. Multiplication result is obtained by multiplying multiplicand and multiplier. This multiplication result is accumulated with previous result. When we consider a MAC unit there is a feedback connection between the result and adder where the result is cumulatively added with the new input.As the feedback is very short whatever result we get will enter into the adder due to that final adder result will alter.

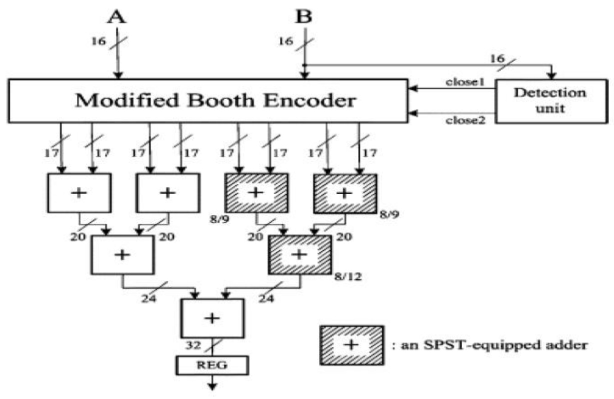

Figure3: Low Power multiplier equipped with SPST[5]

**(1) Modified Booth Encoder:** Booth multiplication is a technique that allows faster multiplication by grouping the multiplier bits. The grouping of multiplier bits and Radix-2 Booth encoding reduce the number of partial products to half. So we take every second column, and multiply by ±1, ±2, or 0, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0. To obtain the correct partial product each block is decoded. Table IV shows the encoding of the multiplier Y, using the modified Booth algorithm, generates the following five signed digits, -2, - 1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand X.

Table IV: Operations on Multiplicand

| Block | Re - coded digit | Operation on X |
|---|---|---|
| 000 | 0 | 0 X |
| 001 | +1 | +1 X |
| 010 | +1 | +1 X |
| 011 | +2 | +2 X |
| 100 | -2 | -2 X |
| 101 | -1 | -1 X |
| 110 | -1 | -1 X |
| 111 | 0 | 0 X |

(2) *Spurious Power Suppression Techniques and Registers*

Spurious transitions in combinational CMOS logic are a well known source of unnecessary power dissipation. Reducing glitch power is a highly desirable target because in the vast majority of digital CMOS circuits, only one signal transition per clock cycle is functionally meaningful. The procedure for glitch minimization is based on a known concept. Glitches are eliminated by adding some redundant logic that prevents spurious transitions. This can be done by inserting latches in a gate-level net list.

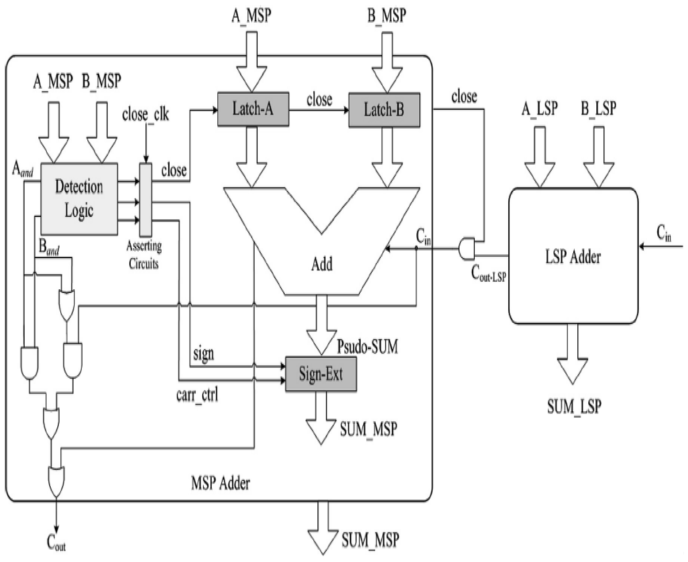

Fig4: Low Power Adder using SPST[5]

In the above figure 4 the 16-bit adder is divided into MSP (Most Significant Part) and LSP (Least Significant Part) between the 8th and 9th bits.Whenever we consider input, we divide into two equal parts where one is considered as MSB and other LSB. In LSB we do normal addition such that from the adder we get a carry signal. Depending upon that carry signal in LSB part obtained is transferred to MSB block. Depending on that carry value detection logic, sign extension we calculate MSB part sum. Propogation delay increases from the input to the output such that change in the input does not affect suddenly at the output.

**(3) Applying SPST to Modified Booth Encoder:**

Mainly SPST is used in MAC unit with respect to Modified Encoder Multiplier. By using detection logic it controls the execution of Booth Encoder calculation. It stops flow of inputs into the MUX's before addition. If all partial products are zero or LSB partial products are zero then it stops the booth encoding block. Due to this the power used to encode is reduced. The SPST equipped modified Booth encoder, which is controlled by a detection unit. One of the two operands as input to the detection unit, which decide whether the Booth encoder calculates redundant computations.

The SPST adder can be used in Booth Encoder multiplier to add non-zero partial products. Asserting circuits are used to assert when the latches should work and when not.

## (D)IMPLEMENTATION OF MAC USING SPST

### (a)Hardware Analysis of SPST MAC

The MAC architecture using is analyzed and hardware resources are calculated experimentally and then compared with the previous results. The results are shown in Table VI. When we are calculating the amount of hardware resources, the resources for Booth encoder is excluded.

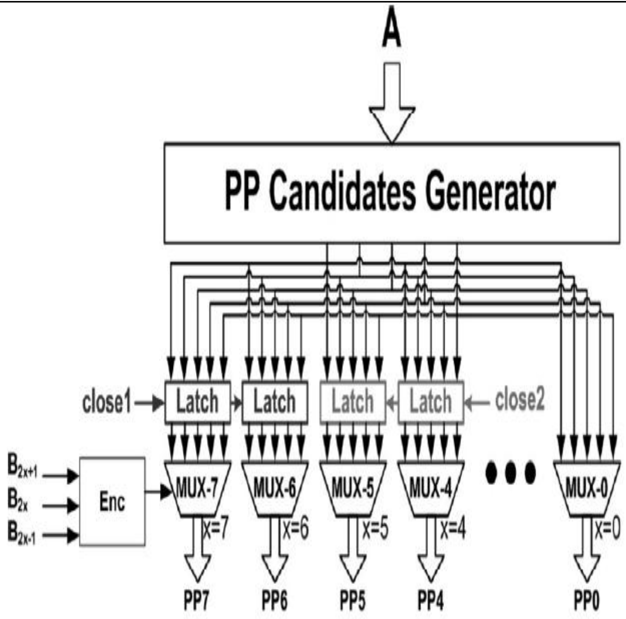

Fig5: Modified Booth Encoder with SPST[5]

Table V: Synthesis Report

| **Device Parameters** | **MAC with SPST(Spartan 3E)** | **Spartan6** | **Vertex 5** |
|---|---|---|---|
| Number of Slices | 435 out of 4656 | 50 out of 30064 | 42 out of 12480 |
| Number of Slice Flip-Flops | 42 out of 9312 | 550 out of 15032 | 516 out of 12480 |
| Number of 4 input LUT's | 790 out of 9312 | 41 out of 591 | 24 out of 534 |
| Number of IO's | 66 | 66 | 66 |
| Number of BONDED IOB's | 66 out of 190 | 66 out of 190 | 66 out of 172 |
| Number of GCLK's | 1 out of 24 | 1 out of 16 | 1 out of 32 |

**(b) Delay Time Analysis:** The maximum combinational path delay is analyzed in various devices and families and the results are shown in table VII.

Table VI: Delay Time Analysis Of SPST MAC

| **Device** | **SPST MAC (Spartan 3E)** | **SPST MAC (Spartan 6)** | **SPST MAC (Vertex 5)** |
|---|---|---|---|
| Maximum Combinational Path Delay | 4.450ns | 3.732ns | 2.844ns |

## III. RESUTS

The simulation results for 16-bit MAC using Radix-2 modified Booth algorithm and SPST MAC are shown in figures 6 and 7 respectively.

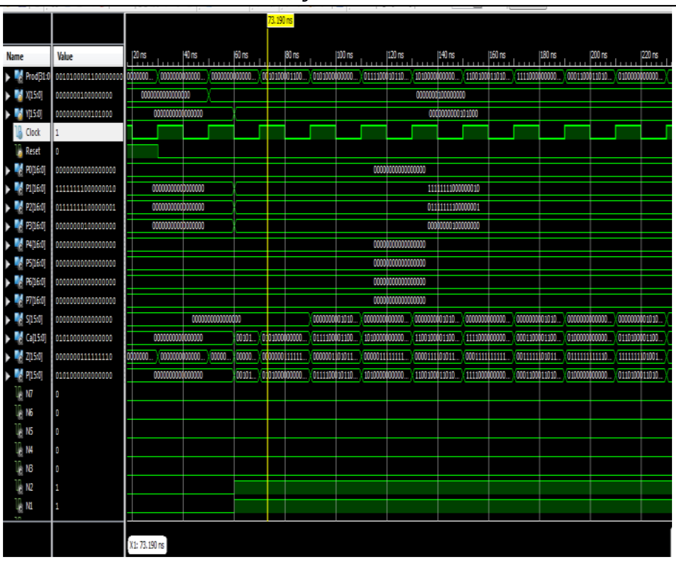

Fig 6.Simulation Results for 16-bit MAC using Radix-2 modified Booth algorithm and CSA

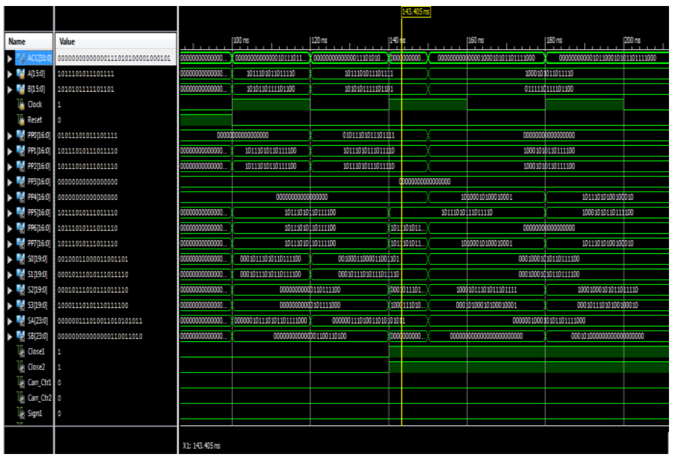

Fig 7. Simulation Results for 16-bit MAC with SPST Adder.

The RTL Schematics for 16-bit MAC using CSA and SPST MAC are shown in figures 8 and 9 respectively.

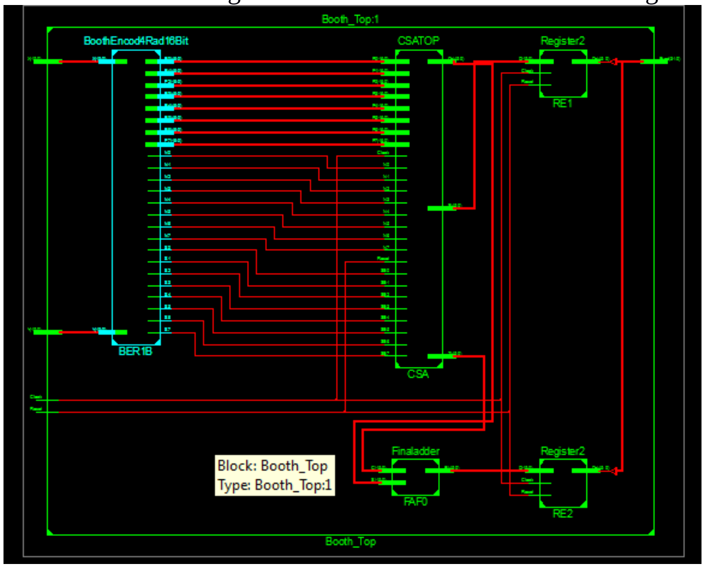

Fig 8. RTL Schematic for 16-bit MAC using CSA.

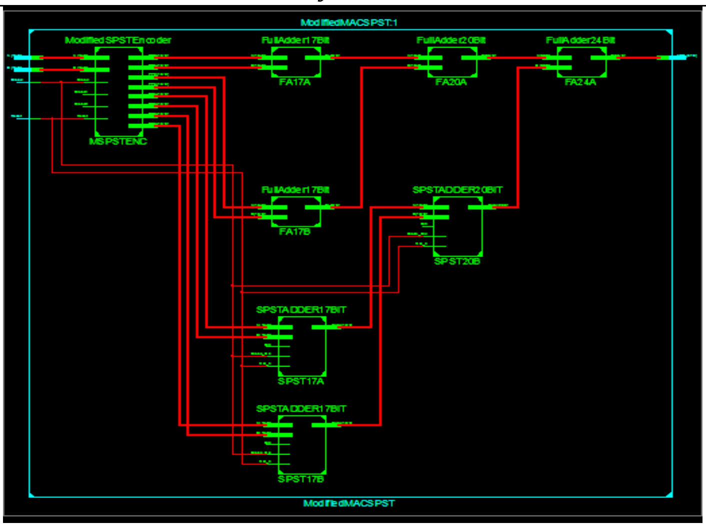

Fig 9. RTL Schematic for 16-bit MAC using SPST.

## IV. CONCLUSION

In this paper we have compared two MAC architectures using CSA and using SPST. The architectures were implemented and synthesized using XILINX. By removing the independent accumulation process and merging it to the partial product compression process, the overall MAC performance has been improved. As compared to the previous results the hardware resources used by SPST MAC were very less. The performance has been increased. This architecture can be used in the areas requiring high throughputs. The Radix-2 modified Booth Algorithm reduced the number of partial products to half and improved the speed. The implemented MAC with SPST gives less overall delay and power consumption.

## RFERENCES

[1]. O. L. Mac Sorley, "High speed arithmetic in binary computers," Proc. IRE, vol. 49, pp. 67–91, Jan. 1961.

[2]. A New VLSI Architecture of Parallel Multiplier–Accumulator Based on Radix-2 Modified Booth Algorithm by Young-Ho Seo, Member, IEEE, and Dong-Wook Kim,IEEE Transcations on very large scale integration (VLSI)Systems Vol. 18, No. 2, February 2010.

[3]. F. Elguibaly, "A fast parallel multiplier–accumulator using the modified Booth algorithm," IEEE Trans. Circuits Syst., vol. 27, no. 9, pp. 902–908, Sep. 2000.

[4]. C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[5]. Kuan-Hung Cheng and Yuan-Sun Chu,"A Low Power Multiplier with Spurious Power Suppression Technique",IEEE Transactions on VLSI System,Vol.15,No.7,July 2007.

[6]. A. D. Booth, "A signed binary multiplication technique," Quart. J. Math., vol. IV, pp. 236–240, 1952.

[7]. A. R. Cooper, "Parallel architecture modified Booth multiplier," Proc. Inst. Electr. Eng. G, vol. 135, pp. 125–128, 1988.

[8]. J. Fadavi-Ardekani, "MxN Booth encoded multiplier generator using optimizedWallace trees," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 1, no. 2, pp. 120–125, Jun. 1993

[9]. N. R. Shanbag and P. Juneja, "Parallel implementation of a 4x4-bit multiplier using modified Booth's algorithm," IEEE J. Solid-State Circuits, vol. 23, no. 4, pp. 1010–1013, Aug. 1988.

[10]. A. Tawfik, F. Elguibaly, and P. Agathoklis, "New realization and implementation of fixed-point IIR digital filters," J. Circuits, Syst., Comput., vol. 7, no. 3, pp. 191–209, 1997.