International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 9, November- 2012

FPGA Implementation Of Distributed Arithmetic For FIR Filter

M. Keerthi [1], Vasujadevi Midasala[2], S Nagakishore Bhavanam[3], Jeevan Reddy K[4]
[2]Assistant Professor, Associate Professor[4] ,
[1,2,4] Dept. of ECE, TeegalaKrishnaReddy Engineering College, Andhra Pradesh, India
[3] Assistant Professor, Dept. of ECE, University College of Engineering & Technology, ANU, Guntur,

Abstract - Digital filters are the essential units for digital signal processing systems. Traditionally, digital filters are achieved in Digital Signal Processor (DSP), but DSP-based solution cannot meet the high speed requirements in some applications for its sequential structure. Nowadays, Field Programmable Gate Array (FPGA) technology is widely used in digital signal processing area because FPGA-based solution can achieve high speed due to its parallel structure and configurable logic, which provides great flexibility and high reliability in the course of design and later maintenance.

In general, Digital filters are divided into two categories, including Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). And FIR filters are widely applied to a variety of digital signal processing areas for the virtues of providing linear phase and system stability.

The FPGA-based FIR filters using traditional direct arithmetic costs considerable multiply-and-accumulate (MAC) blocks with the augment of the filter order. A new design and implementation of FIR filters using Distributed Arithmetic is provided in this project to solve this problem.

Distributed Arithmetic structure is used to increase the resource usage while pipeline structure is also used to increase the system speed. In addition, the divided LUT method is also used to decrease the required memory units. However, according to Distributed Arithmetic, we can make a Look-Up-Table (LUT) to conserve the MAC values and callout the values according to the input data if necessary. Therefore, LUT can be created to take the place of MAC units so as to save the hardware resources.

This project provide the principles of Distributed Arithmetic, and introduce it into the FIR filters design, and then presents a 31-order FIR low-pass filter using Distributed Arithmetic, which save considerable MAC blocks to decrease the circuit scale, meanwhile, divided LUT method is used to decrease the required memory units and pipeline structure is also used to increase the system speed.

Keywords: Digital Filters, DSP, FPGA, FIR, IIR, MAC, LUT.

I. Introduction

In the recent years, there has been a growing trend to implement digital signal processing functions in Field Programmable Gate Array (FPGA). In this sense, we need to put great effort in designing efficient architectures for digital signal processing functions such as FIR filters, which are widely used in video and audio signal processing, telecommunications and etc.

Traditionally, direct implementation of a K-tap FIR filter requires K multiply-and-accumulate (MAC) blocks, which are expensive to implement in FPGA due to logic complexity and resource usage. To resolve this issue, we first present DA, which is a multiplier-less architecture.

Implementing multipliers using the logic fabric of the FPGA is costly due to logic complexity and area usage, especially when the filter size is large. Modern FPGAs have dedicated DSP blocks that alleviate this problem, however for very large filter sizes the challenge of reducing area and complexity still remains.

An alternative to computing the multiplication is to decompose the MAC operations into a series of lookup table (LUT) accesses and summations. This approach is termed distributed arithmetic (DA), a bit serial method of computing the inner product of two vectors with a fixed number of cycles.

The original DA architecture stores all the possible binary combinations of the coefficients w[k] of equation (1) in a memory or lookup table. It is evident that for large values of L, the size of the memory containing the pre computed terms grows exponentially too large to be practical. The memory size can be reduced by dividing the single large memory (2Lwords) into m multiple smaller sized memories each of size 2k where L = m × k. The memory size can be further reduced to 2L−1 and 2L−2 by applying offset binary coding and exploiting resultant symmetries found in the contents of the memories.

This technique is based on using 2's complement binary representation of data, and the data can be pre-computed and stored in LUT. As DA is a very efficient solution especially suited for LUT-based FPGA architectures, many researchers put great effort in using DA to implement FIR filters in FPGA.

Patrick Longa introduced the structure of the FIR filter using DA algorithm and the functions of each part. Sangyun Hwang analyzed the power consumption of the filter using DA algorithm. Heejong Yoo proposed a modified DA architecture that gradually replaces LUT requirements with multiplexer/adder pairs. But the main problem of DA is that the requirement of LUT capacity increases exponentially with the order of the filter, given that DA implementations need 2Kwords (K is the number of taps of the filter). And if K is a prime, the hardware resource consumption will cost even higher. To overcome these problems, this paper presents a hardware-efficient DA architecture.

This method not only reduces the LUT size, but also modifies the structure of the filter to achieve high speed performance. The proposed filter has been designed and synthesized with ISE 7.1, and implemented with a 4VLX40FF668 FPGA device. Our results show that the proposed DA architecture can implement FIR filters with high speed and smaller resource usage in comparison to the previous DA architecture.

1.1 Objective and goal

Traditionally, direct implementation of a K-tap FIR filter requires K multiply-and-accumulate (MAC) blocks, which are expensive to implement in FPGA due to logic complexity and resource usage. To resolve this issue, we first present DA, which is a multiplier-less architecture.

An alternative to computing the multiplication is to decompose the MAC operations into a series of lookup table (LUT) accesses and summations. This approach is termed distributed arithmetic (DA), a bit serial method of computing the inner product of two vectors with a fixed number of cycles. The original DA architecture stores all the possible binary combinations of the coefficients w[k] of equation (1) in a memory or lookup table.

It is evident that for large values of L, the size of the memory containing the pre computed terms grows exponentially too large to be practical. This technique is based on using 2's complement binary representation of data, and the data can be pre-computed and stored in LUT. As DA is a very efficient solution especially suited for LUT-based FPGA architectures, many researchers put great effort in using DA to implement FIR filters in FPGA.

But the main problem of DA is that the requirement of LUT capacity increases exponentially with the order of the filter, given that DA implementations need 2Kwords (K is the number of taps of the filter). And if K is a prime, the hardware resource consumption will cost even higher.

II. Literature Survey

2.1 Existing System

Existing systems for implementing FIR filter are the multiplier based designs and Distributed arithmetic based Designs. For multiplication based Designs number of multiplications required is large and the hardware utilization will be high in FPGA solutions. In Distributed Arithmetic concept FIR filter is implemented using LUT, add and shift hardware, in which no multiplications will be there. LUT stores all the possible combinations of the sums of the coefficients. Add and shift hardware is to implement the Filter functionality by using LUT contents. This method is explained in detail in the chapter FIR filter design using Distributed arithmetic in this thesis.

2.2 Implemented System

Here we present four different architectures analyzing their advantages and disadvantages. The implemented system starts from the original DA architecture. We then reduce the hardware by implementing lut-less and 4-input look up table architectures. Finally we implement a fully parallel architecture for high speed fir filters.

We implement this architecture for a 70 tap filter and compare the results with the original DA architecture. We apply pipelining to achieve high speed.

III. Distributed Arithmetic

3.1 Background

Traditional implementations of the finite impulse response (FIR) filter equation is

$$y[n] = \sum_{k=0}^{L-1} w[k]x[n-k]$$

Typically employ L multiply-accumulate (MAC) units. Implementing multipliers using the logic fabric of the FPGA is costly due to logic complexity and area usage, especially when the filter size is large. Modern FPGAs have dedicated DSP blocks that alleviate this problem, however for very large filter sizes the challenge of reducing area and complexity still remains. An alternative to computing the multiplication is to decompose the MAC operations into a series of lookup table (LUT) accesses and summations. This approach is termed distributed arithmetic (DA).

3.2 Theory

Distributed arithmetic (DA) is a bit serial method of computing the inner product of two vectors with a fixed number of cycles. The original DA architecture stores all the possible binary combinations of the coefficients w[k] of (1) in a memory or lookup table. It is evident that for large values of L, the size of the memory containing the pre computed terms grows exponentially too large to be practical. The memory size can be reduced by dividing the single large memory (2L words) into m multiple smaller sized memories each of size 2k where L = m × k. The memory size can be further reduced to 2L−1 and 2L−2 by applying offset binary coding and exploiting resultant symmetries found in the contents of the memories. However, for very large values of L, the listed approaches still succumb to the limitations of storing the coefficient combinations in memory due to the exponential dependence of memory size on filter length.

A simplified view of a DA FIR is shown in Figure 3.2.1. In its most obvious and direct form, DA based computations are bit-serial in nature which means serial distributed arithmetic (SDA) FIR. Extensions to the basic algorithm remove this potential throughput limitation. The advantage of a distributed arithmetic approach is its efficiency of mechanization. The basic operations required are a sequence of table look-ups, additions, subtractions and shifts of the input data sequence. All of these functions efficiently map to FPGAs. Input samples are presented to the input parallel-to serial shift register (PSC) at the input signal sample rate.

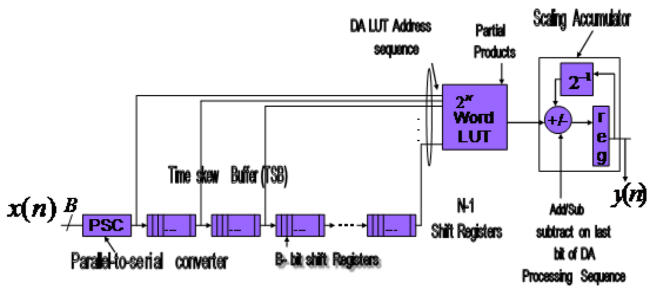


Figure 3.1: Serial distributed arithmetic FIR filter.

As the new sample is serialized, the bit wide output is presented to a bit-serial shift register or time-skew buffer (TSB). The TSB stores the input sample history in a bit-serial format and is used in forming the required inner-product computation. The TSB is itself constructed using a cascade of shorter bit–serial shift registers. The nodes in the cascade connection of TSB's are used as address inputs to a look-up table. This LUT stores all possible partial products over the filter coefficient space.

Several observations provide valuable insight into the operation of a DA FIR filter. In a conventional multiply-accumulate (MAC) based FIR realization, the sample throughput is coupled to the filter length. With DA architecture the system sample rate is related to the bit precision of the input data samples. Each bit of an input sample must be indexed and processed in turn before a new output sample is available. For B-bit precision input samples, B clock cycles are required to form a new output sample for a non-symmetrical filter, and B+1 clock cycles are needed for a symmetrical filter.The rate at which data bits are indexed occurs at the bit-clock rate. The bit-clock frequency is greater than the filter sample rate f_s and is equal to B f_s for a non-symmetrical filter and (B+1) f_s for a symmetrical filter.

In a conventional instruction-set (processor) approach to the problem, the required number of multiply-accumulate operations are implemented using a time-shared or scheduled MAC unit. The filter sample throughput is inversely proportional to the number of filter taps. As the filter length is increased the system sample rate is proportionately decreased. This is not the case with DA based architectures. The filter sample rate is de-coupled from the filter length. The trade off introduced here is one of silicon area (FPGA logic resources) for time. As the filter length is increased in a DA FIR filter, more logic resources are consumed, but throughput is maintained.

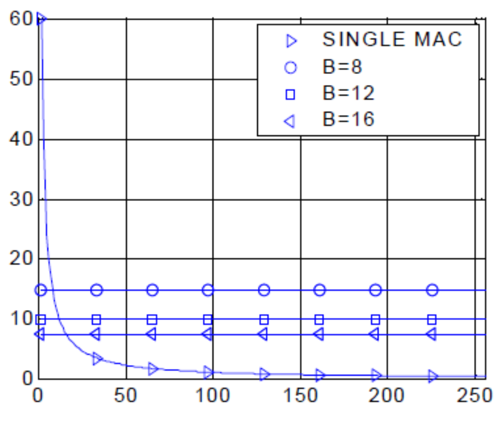


Figure 3.2 Throughput (sample rate) comparison of single-MAC based FIR and DA FIR as a function of filter length. B is the DA FIR input sample precision. The clock rate is 120 MHz.

Figure 3.2 provides a comparison between a DA FIR architecture and a conventional scheduled MAC-based approach. The clock rate is assumed to be 120 MHz for both filter architectures.

Several values of input sample precision for the DA FIR are presented. The dependency of the DA filter throughput on the sample precision is apparent from the plots.

For 8-bit precision input samples, the DA FIR maintains a higher throughput for filter lengths greater than 8 taps. When the sample precision is increased to 16 bits, the crossover point is 16 taps.

IMPLEMENTATION OF FIR FILTER USING DISTRIBUTED ARITHMETIC

3.4 Distributed Arithmetic FIR filter architecture

Distributed Arithmetic is one of the most well-known methods of implementing FIR filters. The DA solves the computation of the inner product equation when the coefficients are pre knowledge, as happens in FIR filters.

An FIR filter of length K is described as:

$$y[n] = \sum_{k=0}^{K-1} h[k]x[n-k] \qquad \text{..............................(1)}$$

Where h[k] is the filter coefficient and x[k] is the input data. For the convenience of analysis, x'[k] =x [n - k] is used for modifying the equation (1) and we have:

$$y = \sum_{k=0}^{K-1} h[k].x^{'}[k] \qquad \text{... (2)}$$

Then we use B-bit two's complement binary numbers to represent the input data:

$$x^{'}[k] = -2^{B}\cdot x_B[k] + \sum_{b=0}^{B-1} x_b[k]\cdot 2^{b} \qquad \text{..................(3)}$$

where $x_b[k]$ denotes the b'th bit of x[k], $x_b[k] \in$ {0, 1}.

Substitution of (3) into (2) yields:

$$y = \sum_{k=0}^{K-1} h[k]\cdot(-2^{B}\cdot x_B[k] + \sum_{b=0}^{B-1} x_b[k]\cdot 2^{b})$$

$$= -2^{B}\cdot \sum_{k=0}^{K-1} h[k]\cdot x_B[k] + \sum_{b=0}^{B-1} 2^{b}\cdot \sum_{k=0}^{K-1} h[k]\cdot x_b[k]$$

$$= -2^{B}\cdot f(h[k], x_B[k]) + \sum_{b=0}^{B-1} 2^{b}\cdot f(h[k], x_b[k]) \qquad \text{............ (4)}$$

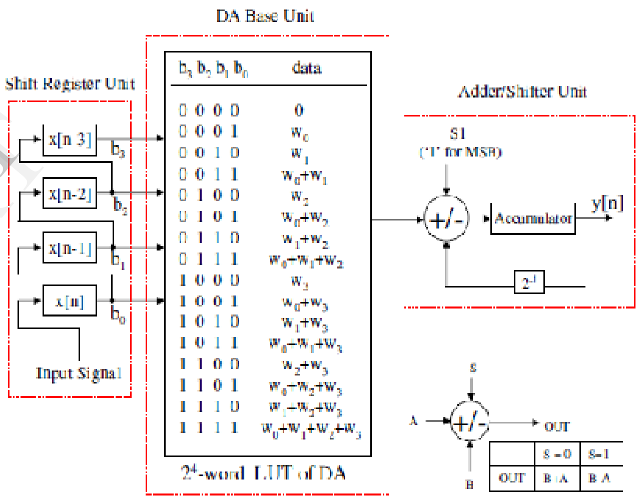
We have

$$f(h[k], x_b[k]) = \sum_{k=0}^{K-1} h[k]\cdot x_b[k] \qquad \text{......(5)}$$

In equation (4), we observe that the filter coefficients can be pre-stored in LUT, and addressed by xb = [$x_b[0]$, $x_b[1]$,... $x_b[K-1]$]. This way, the MAC blocks of FIR filters are reduced to access and summation with LUT.

The implementation of digital filters using this arithmetic is done by using registers, memory resources and a scaling accumulator.

Original LUT-based DA implementation of a 4-tap (K=4) FIR filter is shown in Figure 3.3 The DA architecture includes three units: the shift register unit, the DA-LUT unit, and the adder/shifter unit.



b_3	b_2	b_1	b_0	data
0	0	0	0	0
0	0	0	1	W_0
0	0	1	0	W_1
0	0	1	1	W_0+W_1
0	1	0	0	W_2
0	1	0	1	W_0+W_2
0	1	1	0	W_1+W_2
0	1	1	1	$W_0+W_1+W_2$
1	0	0	0	W_3
1	0	0	1	W_0+W_3
1	0	1	0	W_1+W_3
1	0	1	1	$W_0+W_1+W_3$
1	1	0	0	W_2+W_3
1	1	0	1	$W_0+W_2+W_3$
1	1	1	0	$W_1+W_2+W_3$
1	1	1	1	$W_0+W_1+W_2+W_3$

	S=0	S=1
OUT	B+A	B-A

Figure 3.3 Original LUT-based DA implementation of a 4-tap filter

IV. LOOK UP TABLE LESS DISTRIBUTED ARITHMETIC ARCHITECTURE

4.1 Motivation for LUT less Architecture

As the filter order increases, the memory size also increases. This in turn increases the look up table (LUT) size. So we use combinational logic in place of look up table for better performance. The proposed DA-LUT unit dramatically reduces the memory usage, since all the LUT units can be replaced by multiplexers and full adders

4.2 Proposed DA-LUT unit

In Fig.3.3, we can see that the lower half of LUT (locations where b_3=1) is the same with the sum of the upper half of LUT (locations where b_3=0) and h [3]. Hence, LUT size can be reduced 1/2 with an additional 2x1 multiplexer and a full adder, as shown in Figure 3.4.

By the same LUT reduction procedure, we can have the final LUT-less DA architectures, as shown in Figure .3.5 On other side, for the use of combination logic circuit, the filter performance will be affected. But when the taps of the filter is a prime, we can use 4-input LUT units with additional multiplexers and full adders to get the trade off between filter performance and small resource usage.

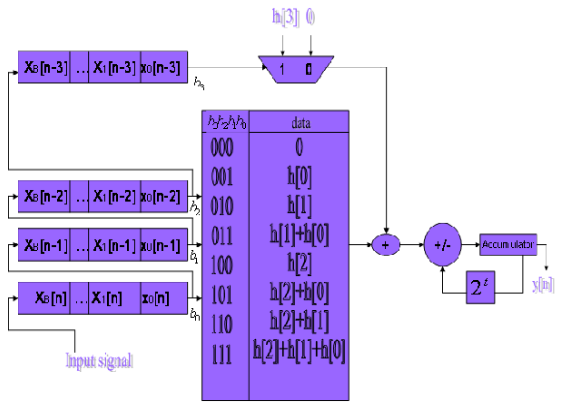


Figure 4.4: Proposed DA architecture for a 4-tap filter (2^3 word LUT implementation of DA)

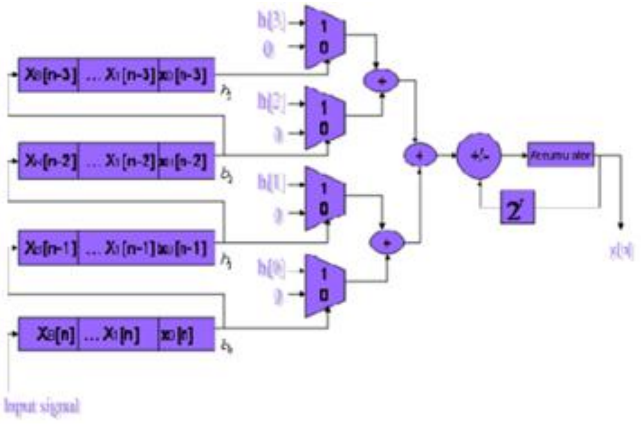

h[3]
0
h[2]
0
h[1]
0
h[0]
0
1
0
X8[n-3] ... X1[n-3] x0[n-3]
X8[n-2] ... X1[n-2] x0[n-2]
X8[n-1] ... X1[n-1] x0[n-1]
X8[n] ... X1[n] x0[n]
+
+/-
Accumulator
2^t
y[n]
Input signal

Figure 4.5: LUT-less DA architectures for a 4-tap FIR filter

V. Results

5.1 Implementation

To evaluate the performance of the proposed scheme, a 70-tap low-pass FIR filter is implemented. The sampling frequency was defined at 40 MHz, the bandwidth of pass band equaled 2 MHz, and the precision for inputs and filter coefficients were 13 and 12 respectively.

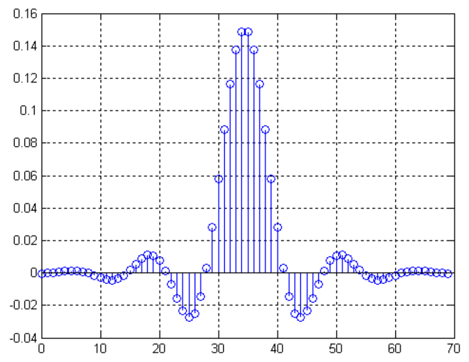


Figure 5.1(a): Impulse response

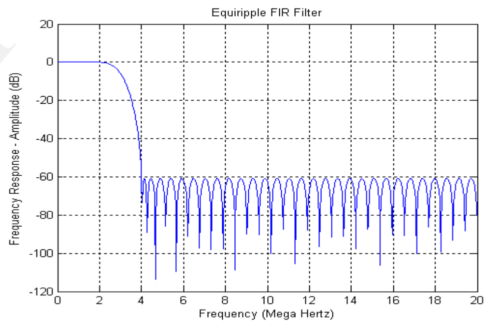


Figure 5.1.(b): Frequency response

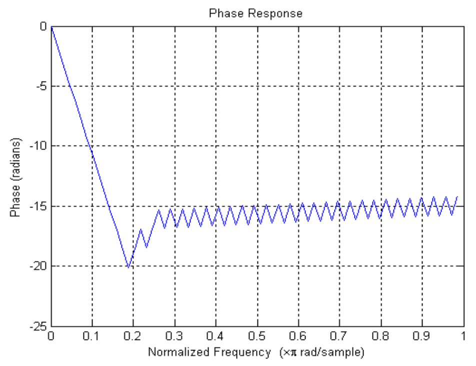


Figure 5.1.(c): Phase response

Firstly, the prototype low pass FIR filter was designed using the McClellan-Parks design algorithm. The impulse response and the frequency response are shown in Figures 5.1.(a). and 5.1(b).

The 70-tap FIR filter had symmetrical structure, so we could reduce it to 35-tap.Then we divided the 35-tap filter into 7 smaller filters each having 5-tap DA-LUT units. The 5-tap DA-LUT unit could be implemented by a 4-input LUT with an additional 2x1 multiplexer and a full adder as shown in figure 9.2.1. To validate the correct functionality of the full-parallel DA architecture, we copied each 5-tap DA-LUT unit 13 times to test whether we could get the final result every clock cycle or not.

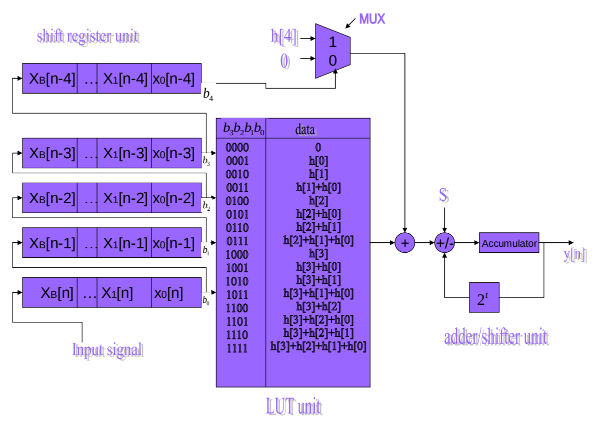


Figure 5.2: 5-Tap Filter

Finally, our implementation was synthesized using ISE7.1 on a 4VLX40FF668 FPGA device. The sampling frequency fs of the input signal was 40 MHz, it had a carrier frequency f_o of 9 MHz and a bandwidth of 2 MHz. As show in Figure 5.3 after mixed with $\cos(2\pi n f_0 / f_s)$ the signal got into the filter.

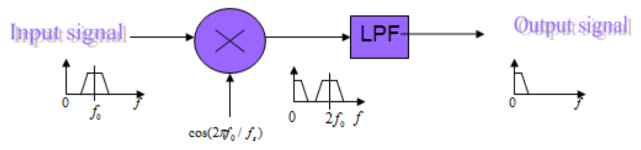
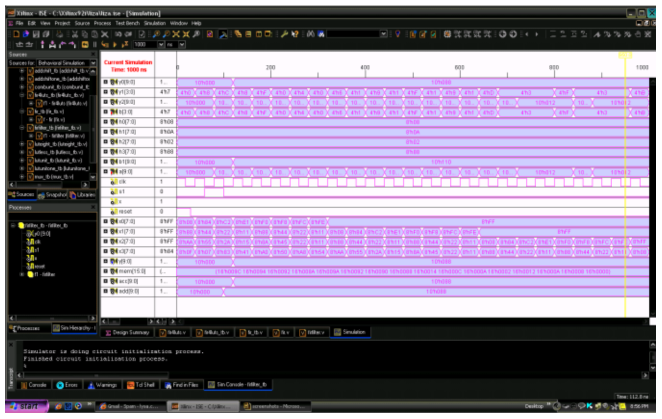


Figure 5.3: Simulation system

We validated the result with a Mat lab code. To illustrate the merits of the proposed DA architecture, the full-parallel version of the original DA architecture was also implemented on a 4VLK40FF668 FPGA device.

5.1- results and Analysis

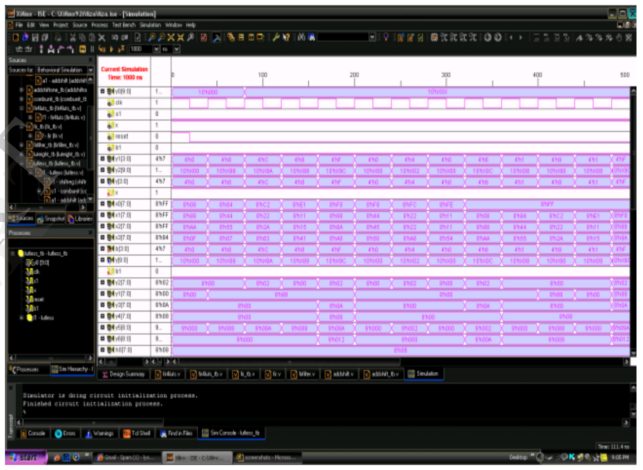
Figure 1 Output



```
Simulator is doing circuit initialization process.
Finished circuit initialization process.
```

Original LUT-based DA implementation of a 4-tap filter

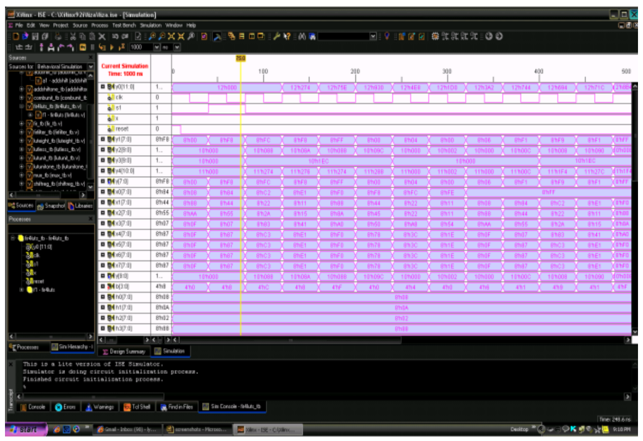
Figure 2 Output



```
Simulator is doing circuit initialization process.
Finished circuit initialization process.
```

LUT-less DA architectures for a 4-tap FIR filter

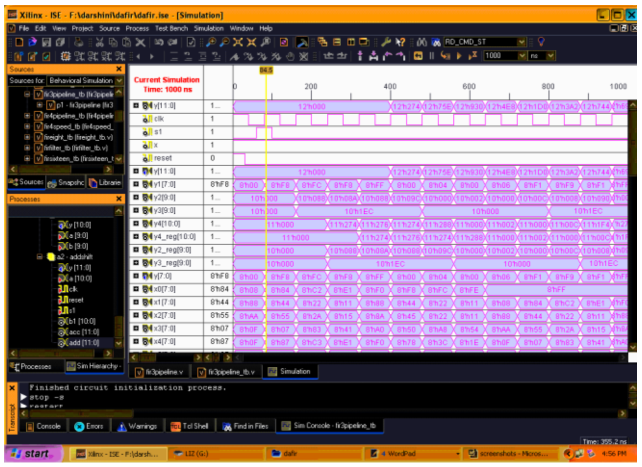
Figure 3 without pipeline registers Output



```
This is a Lite version of ISE Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
```

4-input look up table architecture for high order filters
Without pipelining

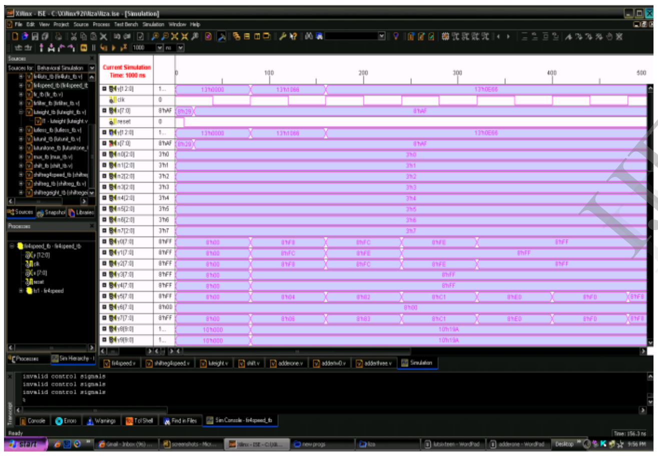
Figure 3 with pipeline registers Output



```
Finished circuit initialization process.
stop -s
restart
```

4-input look up table architecture with pipelining

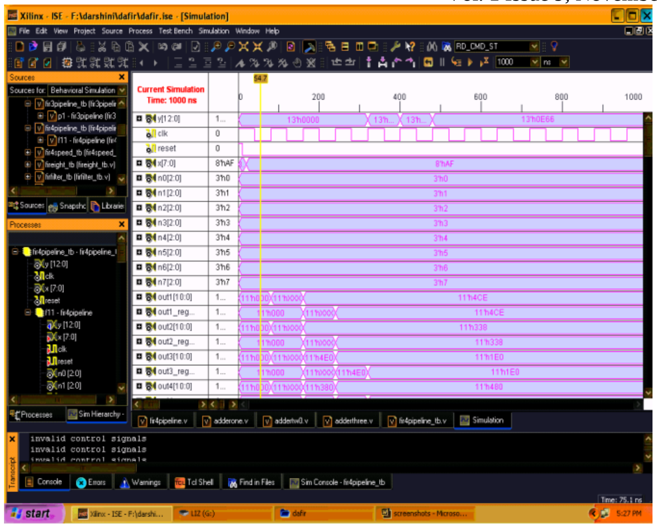
Figure 4 without pipeline registers Output



```
invalid control signals
invalid control signals
invalid control signals
```

Fully parallel DA architecture FIR filter without pipelining

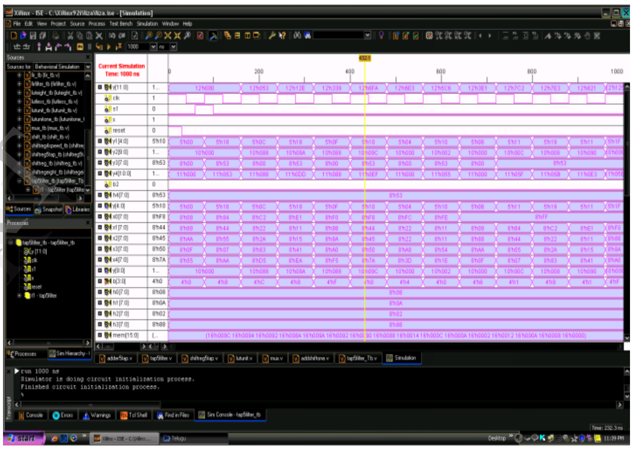
Figure 4 with pipeline registers Output



```
invalid control signals
invalid control signals
invalid control signals
```

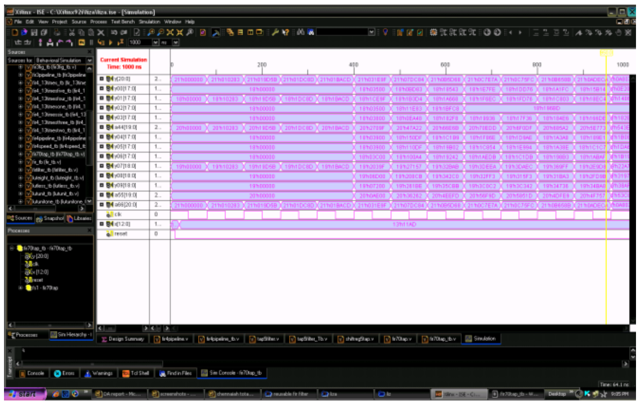
Fully parallel DA architecture FIR filter with pipelining

5-tap filter Output



```
run 1000 ns
Simulator is doing circuit initialization process.
Finished circuit initialization process.
```

5-Tap Filter output



70 tap filter(35 tap) Output

VI. CONCLUSIONS

6.1 Conclusion

This paper presents the proposed DA architectures for high-order filter. The architectures reduce the memory usage by half at every iteration of LUT reduction at the cost of the limited decrease of the system frequency. We also divide the high-order filters into several groups of small filters; hence we can reduce the LUT size also. As to get the high speed implementation of FIR filters, a full-parallel version of the DA architecture is adopted.

We have successfully implemented a high-efficient 70-tap full-parallel DA filter, using both an original DA architecture and a modified DA architecture on a 4VLX40FF668 FPGA device. It shows that the proposed DA architectures are hardware efficient for FPGA implementation.

References

1. Partrick Longa, Ali Miri, "Area-Efficient Fir Filter Design on FPGAs using Distributed Arithmetic" IEEE International Symposium on Signal Processing and Information Technology, pp:248-252,2006

2. Sangyun Hwang, Gunhee Han,Sungho Kang, Jaeseok Kim, "New Distributed Arithmetic Algorithm for Low-Power FIR Filter Implementation", IEEE Signal Processing Letters, Vol.11, No5, pp:463-466,May, 2004

3. Heejong Yoo, David V.Anderson, "Hardware-Efficient Distributed Arithmetic Architecture For High-order Digital Filters", IEEE International Conference on Acoustics, Speech and Signal Processing, Vol.5,pp: 125-128,March,2005

4. Wangdian, Xingwang Zhuo "Digital Systems Applications and Design Based On Verilog HDL", Beijin: National Defence Industry press, 2006.

5. McClellan , J.H. Parks, T.W. Rabiner, L.R. "A computer program for designing optimum FIR linear phase digital filters":. IEEE Trans. Audio Electroacoust. Vol. 21, No.6, pp:506-526, 1973.

6. Httl://cse.stanford.edu/class/sophomore-college/projects-00/risc/pipelining/index.html