



THE EFFICIENT PARALLEL ARCHITECTURE FOR FIXED COEFFICIENT FIR FILTERS USING DISTRIBUTED ARITHMETIC

M.Karthikeyani¹, E.chitra²,

Department of Electronics and Communication Engineering,

SRM University, Kattankulathur,

Tamil Nadu, India

erkarthikams@gmail.com,

chitranithil@gmail.com

June 21, 2018

Abstract

The implementation of FIR filters on FPGA supported ancient technique prices tidy hardware resources, which matches against the decrease of circuit scale and therefore the increase of system speed. a brand new style and implementation of FIR filters exploitation Distributed Arithmetic is provided during this paper to resolve this drawback. Distributed Arithmetic structure is employed to extend the resource usage whereas pipeline structure is additionally wont to increase the system speed. Additionally, the divided LUT technique is additionally wont to decrease the desired memory units. The simulation results indicate that FIR filters exploitation Distributed Arithmetic will work stable with high speed and might save nearly fifty % hardware resources to decrease the circuit scale, and might be applied to a range of areas for its nice flexibility and high reliableness.

Key Words: Least Mean Square (LMS), the divided LUT method and Distributed Arithmetic.

1 INTRODUCTION

Adaptive filters square measure wide employed in many digital signal process applications. The tapped-delay line finite impulse response (FIR) filter whose weights square measure updated by the famed least mean sq. (LMS) algorithmic rule is that the most popularly used adaptive filter not solely because of its simplicity however additionally because of its satisfactory convergence performance. The direct kind configuration on the forward path of the FIR filter ends up in an extended vital path because of AN inner-product computation to get a filter output. Therefore, once the input features a high rate, it's necessary to cut back the vital path of the structure so the vital path couldn't exceed the sampling amount. In recent years, the multiplier-less distributed arithmetic (DA)-based technique has gained substantial quality for its high-throughput process capability and regularity, that lead to efficient and areatime economical computing structures. Hardware-efficient DA-based style of adaptive filter has been. Exploitation 2 separate search tables (LUTs) for filtering and weight update. The structures in don't support high rate since they involve many cycles for LUT updates for every new sample. in an exceedingly recent paper, we've projected associate degree economical design for high-speed DA-based adaptive filter with terribly low adaptation delay. This transient proposes a unique DA-based design for low power, low-area, and high-throughput pipelined implementation of adaptive filter with terribly low adaptation delay. The contributions of this transient area unit as follows

1. Throughput rate is considerably accrued by a parallel LUT update.
2. More sweetening of outturn is achieved by synchronous implementation of filtering and weight change.
3. Typical adder-based shift accumulation is replaced by a conditional carry-save accumulation of signed partial inner product to cut back the sampling amount.
4. Reduction of power consumption is achieved by employing a quick bit clock for carry-save accumulation however a way

slower clock for all alternative operations.

5. The prevailing styles need associate degree auxiliary management unit for address generation, that isn't needed within the projected structure.

2 PROPOSED WORKS

A. Parallel Distributed Arithmetic Architecture

A basic prosecuting attorney design, for a length N th sum-of-product computation, accepts one bit from every of N words. If 2 bits per word are accepted, then the procedure speed may be primarily improved. the utmost speed may be achieved with the totally pipelined word-parallel design as shown in Fig 1, for max speed, a separate storage (with identical content) for every bit vector $x_b[n]$ ought to be provided.

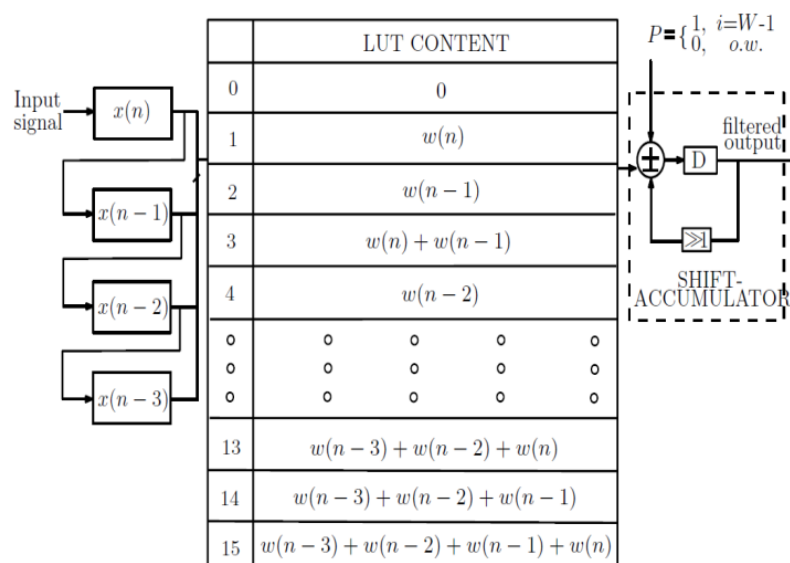


Figure 1 Contents of -LUT for 4-th order FIR filter.

B. Implementation

To evaluate the performance of the Distributed Arithmetic serial and parallel theme for regular FIR filters area unit enforced and

synthesized victimisation Xilinx ISE fourteen.5 Target as a Spartan half dozen FPGA device and therefore the results area unit compared to traditional FIR filter. ISE style software package offers a whole style suit based mostly programmable logic devices on Xilinx ISE. the look is simulated and synthesized within the type of schematic or high-density lipoprotein entry on Xilinx ISE platform. Spartan half dozen FPGA is programming directly from Xilinx ISE in configuration logic blocks interconnected with shift matrix. DSP style is enforced for fewer resources, high speed and low power. The designed FIR filter is programmed in Verilog high-density lipoprotein language. The projected style is enforced for tiny memory location LUT and additionally for big memory location LUT to re-search the performance of the projected style for speed and space parameters. within the gift work, the projected style is analyzed through 3-tap and sixteen tap public prosecutor FIR filters.

C. Design Procedure:

Step1: Derive the filter constant in line with specification of filter.

Step2: Store the inputs worth in input register.

Step3: style the LUTs as shown in Fig.4, that represents all the doable add combination of filter coefficients.

Step4: Accumulate and shift the worth in line with partial term starting with LSB of the input and shift it to the correct to feature it to consecutive partial result.

Step5: initial worth should be subtracted, because of negative little bit of mutual savings bank.

Step6: Analyze the output of filter as per specifications, otherwise attend step one.

Step7: a similar procedure is applied for Parallel prosecutor FIR from step one to step six except in Step3 wherever bit address worth is being needed 2-bits at single time, in order that two occasions LUT is needed compared to the serial prosecutor.

However, with the rise of filter order, the size of LUT can increase dramatically, which can price longer to appear up the table and a lot of memory to store the values. Therefore, we will 255 divide the LUT unit into four little LUT units to resolve this downside.

TABLE I Coefficient values of LUT

$b_3b_2b_1b_0$	Data
0000	0
0001	$h[0]$
0010	$h[1]$
0011	$h[0] + h[1]$
0100	$h[2]$
0101	$h[0] + h[2]$
0110	$h[1] + h[2]$
0111	$h[0] + h[1] + h[2]$
1000	$h[3]$
1001	$h[0] + h[3]$
1010	$h[1] + h[3]$
1011	$h[0] + h[1] + h[3]$
1100	$h[2] + h[3]$
1101	$h[0] + h[2] + h[3]$
1110	$h[1] + h[2] + h[3]$
1111	$h[0] + h[1] + h[2] + h[3]$

Then the values of the four divided LUT units are added as the final value. Coefficient values of small LUT is given in Table-1.

D. Folding Architecture In Addition:

Folding may be a transformation technique mistreatment in DSP design implementation for minimizing the quantity of purposeful blocks in synthesizing DSP design. The folding transformation is applied to fold a bit-parallel design to a digit-serial or bit-serial one. It reduces the semiconductor space by time multiplexing several rule operations into single purposeful units.

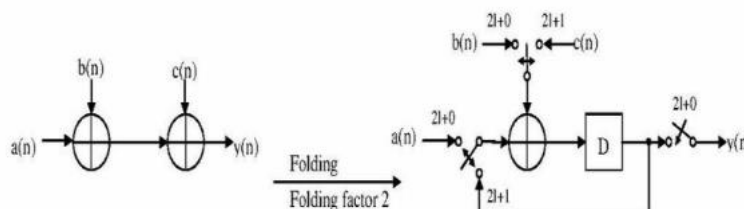


Figure 2 Folded Adder.

The resource utilized in original system is two adders, and therefore the resources utilized in reworked system square measure one adder, one register and three electronic device. The practical block and adder, is so reduced. within the original system the output $y(n)$ is obtained by the addition of 3 inputs ($a(n)$, $b(n)$ and $c(n)$) victimisation 2 operators. during this folding technique we tend to uses 3 electronic device as switch logic, initially the 2 input worths ($a(n)$ and $b(n)$) square measure get added and therefore the result's hold on during a register then the third worth ($c(n)$) is added with the hold on value and at last the output $y(n)$ is obtained.

E. Folded Architecture In Fir Filters:

Folding architectures is planned for reducing the hardware and also the space with low power consumption only the output needed by the applying is lesser than the operative output. the final block of Distributed Arithmetic (DA) and folded-up adder based mostly finite impulse response (FIR) filter is shown in below figure.

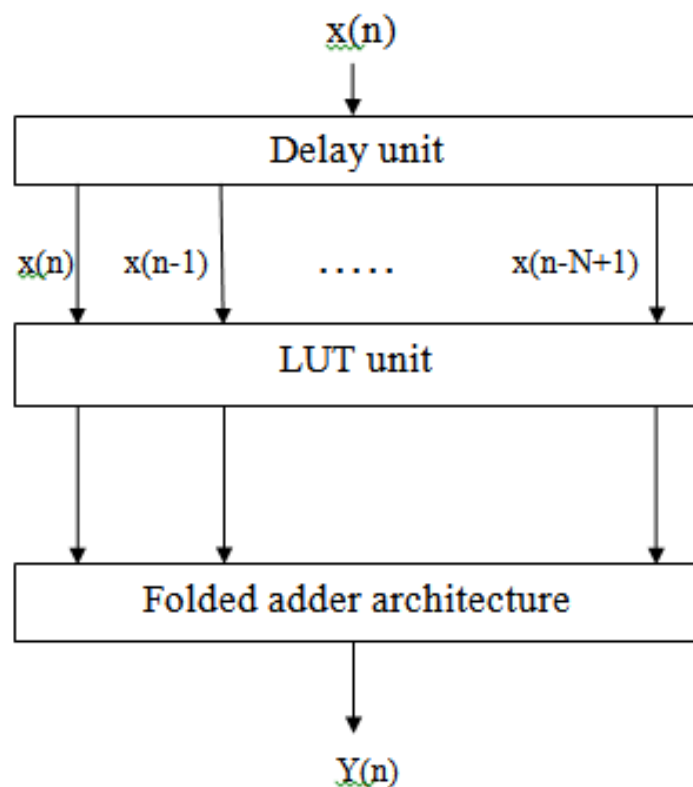


Figure 3 Folded Architecture in Fir Filters

It consists of 1 delay unit (DU), one find Table (LUT) unit and one closed adder design. The DU consists of $(N - 1)$ number of B-bit registers, The LUT unit comprises M, B-ROM cells of breadth B0, the closed adder design comprises one adder, one register and 3 electronic device unit, during this the electronic device unit can act because the change logic unit.

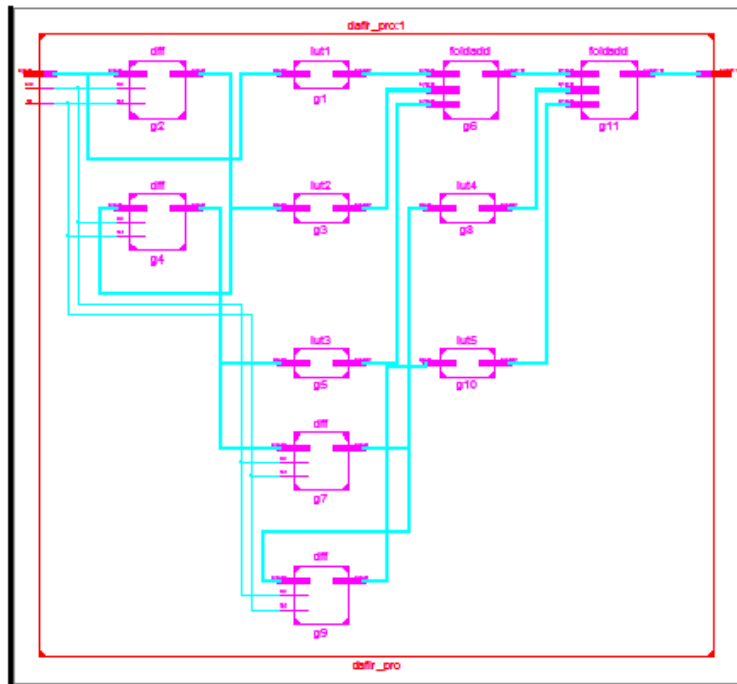


Figure 4 RTL Schematic representing DA with Folding Architecture

3 SIMULATION RESULTS:

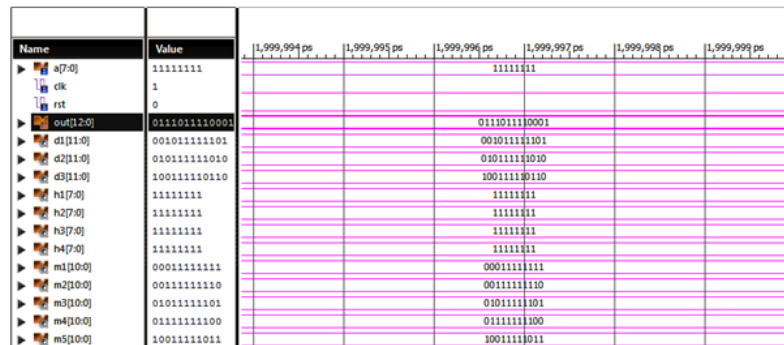


Figure 5 Output waveform of FIR Filter using DA without Folding Architecture

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	32	11,440	1%	
Number used as Flip Flops	32			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	77	5,720	1%	
Number used as logic	61	5,720	1%	
Number using O6 output only	39			
Number using O5 output only	0			

Figure 6 Device Utilization Summary when implementing FIR Filter using DA without Folding Architecture

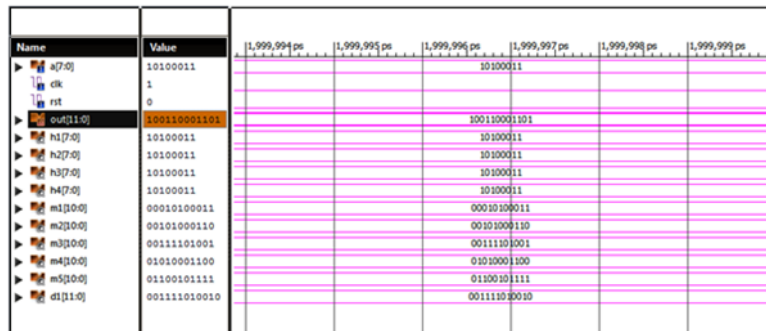


Figure 7 Output waveform of FIR Filter using DA with Folding Architecture

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	50	11,440	1%	
Number used as Flip Flops	32			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	18			
Number of Slice LUTs	56	5,720	1%	
Number used as logic	47	5,720	1%	
Number using O6 output only	20			
Number using O5 output only	0			

Figure 8 Device Utilization Summary when implementing FIR Filter using DA and Folding Architecture

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
lut5	307	26432.064	28247.209	54679.272
mux_h_41_5	11	22863.618	5866.511	28730.129
ctl_a_41_5	296	3568.446	20018.198	23586.644

Figure 9 Power Consumption of FIR Filter using DA without Folding Architecture

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
a[4]	(u) in port	9	63.0	0	+0	0	F
ctl_a_41_5/in_0[4]							
g773/in_0					+0	0	
g773/z	(u) unmapped_not	8	56.0	0	+106	106	R
g721/in_3					+0	106	
g721/z	(u) unmapped_nand4	16	28.0	0	+216	322	F
g765/in_1					+0	322	
g765/z	(u) unmapped_nor2	11	77.0	0	+150	472	R
ctl_a_41_5/out_0[1]							
mux_h_41_5/ctl[1]					+0	472	
g21/ctl254							
g21/z	unmapped_mux512	1	0.0	0	+519	991	R
mux_h_41_5/z[0]							
h[0]	out port				+0	991	R

Figure 10 Time Delay in FIR Filter using DA without Folding Architecture

Instance	Cells	Cell Area	Net Area	Total Area	Wireload
lut5	307	23956	0	23956	<none> (D)
mux_h_41_5	11	21484	0	21484	<none> (D)
ctl_a_41_5	296	2472	0	2472	<none> (D)

Figure 11 Area analysis of FIR Filter using DA without Folding Architecture

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
lutfir	2	71.590	12715.277	12786.867
g1	0	0.000	1358.437	1358.437
g2	0	0.000	1712.812	1712.812
g3	0	0.000	1358.437	1358.437
u1	0	0.000	236.250	236.250
u2	0	0.000	295.312	295.312
u3	0	0.000	295.312	295.312
u4	0	0.000	295.312	295.312
u5	0	0.000	295.312	295.312
u6	0	0.000	295.312	295.312
u7	0	0.000	295.312	295.312
u8	0	0.000	287.930	287.930
u9	0	0.000	295.312	295.312
u10	0	0.000	295.312	295.312
u11	0	0.000	295.312	295.312
g4	0	0.000	1712.812	1712.812
g5	0	0.000	1358.437	1358.437
n1	0	0.000	236.250	236.250
n2	0	0.000	295.312	295.312
n3	0	0.000	295.312	295.312
n4	0	0.000	295.312	295.312

Figure 12 Power Consumption of FIR Filter using DA with Folding Architecture

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
n11/carry	(u) fadd_19	1	7.0	0	+0	0	R
n12/in_1					+0	0	
n12/z	(u) unmapped_xor2	1	7.0	0	+126	126	R
j12/in_0					+0	126	
j12/z	unmapped_xor2	1	0.0	0	+106	232	R
k12/a	hadd				+0	232	

Figure 13 Time Delay in FIR Filter using DA with Folding Architecture

Instance	Cells	Cell Area	Net Area	Total Area	Wireload
lutfir	2	15	0	15	<none> (D)

Figure 13 Area Analysis of FIR Filter using DA with Folding Architecture

4 CONCLUSION

Finite Impulse Response filter plays a crucial role in several Digital Signal process applications. During this technique, the number less FIR filter is enforced exploitation Distributed Arithmetic that consists of hunt Table during which access time is a smaller amount than multiplication time. This system reduces the delay, space and power consumption. The performance is more improved by folding adder transformation that reduces the quantity of useful adder units within the design. This design provides Associate in area-time power implementation that involves considerably less latency and less area-delay quality when put next with existing structures for FIR Filter.

References

- [1] M. Yazhini¹ and R. Ramesh². FIR Filter Implementation using Modified Distributed Arithmetic Architecture.
- [2] Subodh Kumar Singhal and Basant Kumar Mohanty. Efficient Parallel Architecture for Fixed-Coefficient and Variable-Coefficient FIR Filters Using Distributed Arithmetic.
- [3] P.Premkumar, Dr.S.Kavitha, S.Nandhini. Design and implementation of folded FIR filter structures using high speed multipliers.
- [4] K. Subramanian, Dr. R. Prema, S.Muthukrishnan. Transposed form of folded FIR filter.
- [5] K.Indraja¹ , M.Madhusudhan Reddy. Low-Power, High-Throughput and Low-Area Adaptive Fir Filter Based On Distributed Arithmetic Using FPGA.

