Research Article

Implementation of Energy Efficient and Low Complex Fir Filters with Reconfigurability Usage

Authors:

[1] G.J.V.S.N. Lakshmi Devi, [2]M. Ramesh Kumar.

Address for Correspondence:

[1]M.Tech II Year, ASR Institute of Technology
[2]Assistant Professor, ASR Institute of Technology.

ABSTRACT:

Power consumption and area optimization are the key requirements of finite impulse response (FIR) filters that are widely used in multi standard wireless communication systems. FIR filter is used to find the response of periodical system under any circumstances. In this project implementation of FIR filter is presented which possess low power consumption and low area occupancy by using modified booth algorithm, gated driver tree techniques respectively. FIR filter involves multiplications, additions, shifting operations, along with storing of yielded outputs. These methods include low power array multiplier and parallel adder. So, dynamically reconfigurable filters can be efficiently implemented by using proposed algorithms for various DSP applications.

INTRODUCTION:

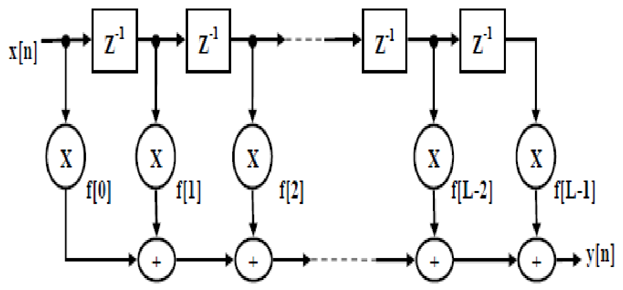
Strength reduction leads to a reduction in hardware complexity by exploiting substructure sharing and leads to less silicon area or power consumption in a VLSI ASIC implementation or less iteration period in a programmable DSP implementation Strength reduction enables design of parallel FIR filters with a less- than-linear increase in hardware. A filter is used to modify an input signal in order to facilitate further processing. A digital filter works on a digital input (a sequence of numbers, resulting from sampling and quantizing an analog signal) and produces a digital output. The most common digital filter is the Linear Time-Invariant (LTI) filter. Designing an LTI involves arriving at the filter coefficients which, in turn, represents the impulse response of the proposed filter design. These coefficients, in linear convolution with an input sequence will result in the desired output. The linear convolution process can be represented by the equation:

$$y[n] = x[n] * f[n] = \sum_k x[k] f[n-k] = \sum_k f[k] x[n-k]$$

Here, y[n] signifies the output of the filter and x[n] is the digital input to the filter. The impulse response of the filter is given by f[k] and the operator '*' denotes the convolution operation. It can be seen that the extent of the summation is denoted by k, which denotes the extent of the impulse response of the filter. Therefore, if the filter has an infinite impulse response, the summation extends to infinity and the filter is said to be an Infinite Impulse Response (IIR) filter. A filter with a finite value or k is said to be a Finite Impulse Response (FIR) filter. It can be inferred that the output of an FIR filter remains dependent only on the inputs and the coefficients. Therefore, the FIR filter discussed above is an LTI filter [2]. The above equation can be re-written for an order of L as

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

Calculating the constant coefficients of such a digital filter involves considerable amount of computation and this is generally performed using software tools.



While sequential FIR filter implementation has been given extensive consideration, very little work has been done that deals directly with reducing the hardware complexity or power consumption of parallel FIR filters. Traditionally, the application of parallel processing to an FIR filter involves the replication of the hardware units that exist in the original filter. The topology of the multiplier circuit also affects the resultant power consumption. Choosing multipliers with more hardware breadth rather than depth would not only reduce the delay, but also the total power consumption.

FIR FILTER THEORY:

Digital filters are typically used to modify or alter the attributes of a signal in the time or frequency domain. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution, denoted by y = f _ x where f is the filter's impulse response, x is the input signal, and y is the convolved output. The linear convolution process is formally defined by

$$y[n] = x[n] * f[n] = \sum_{k=0} x[n]f[n-k] = \sum_{k=0} f[k]x[n-k]$$

LTI digital filters are generally classified as being finite impulse response (i.e., FIR), or infinite impulse response (i.e., IIR). As the name implies, an FIR filter consists of a finite number of sample values, reducing the above convolution sum to a finite sum per output sample instant. An FIR with constant coefficients is an LTI digital filter. The output of an FIR of order or length L, to an input time series x[n], is given by a sum given in (1), namely: where f[0] □ 0 through f[L □ 1] □ 0 are the filter's L coefficients. They also correspond to the FIR's impulse response. For LTI systems it is sometimes more convenient to express in the z-domain with

$$Y(z) = F(z)*X(z),$$

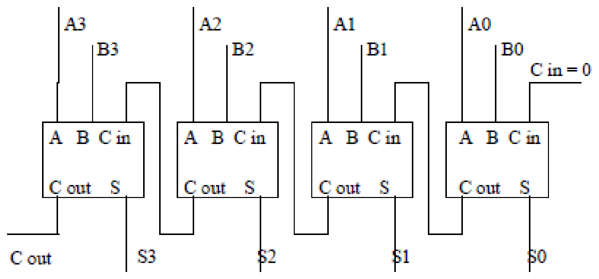
Where F(z) is the FIR's transfer function defined in the z domain by the L^{th}-order LTI FIR filter is graphically interpreted in the figure. It generally consists of tapped delay line adders and multipliers. One of the operands presented to each multiplier is an FIR coefficient, often referred to as a "tap weight" for obvious reasons. Historically, the FIR filter is also known by the name "transversal filter," suggesting its "tapped delay line" Structure.

EXISTING TECHNIQUE:
ARRAY MULTIPLICATIONS:

MULTIPLICATION is a complex arithmetic operation, which is reflected in its relatively high signal propagation delay, high power dissipation, and large area requirement. When choosing a multiplier for a digital system, the bit width of the multiplier is required to be at least as wide as the largest operand of the applications that are to be executed on that digital system. The bit width of the multiplier is, therefore, often much larger than the data represented inside the operands, which leads to unnecessarily high power dissipation and unnecessary long delay.

SERIAL ADDER:

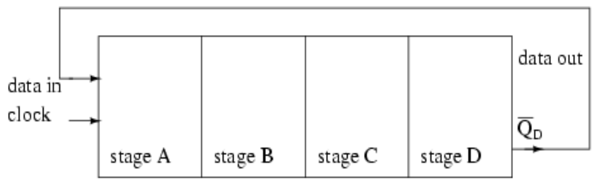
Two multi-bit binary numbers (or words) can be summed by using full adders to emulate the process of manual addition. Figure 17 shows a circuit that can add together two 4-bit numbers A = A3.. A0 and B = B3 .. B0. Four full adders are used, in a chain. The Nth stage adds together the corresponding bits AN & BN from each word (together with any carry from the N-1st stage) and produces the Nth bit in the sum.



There is no need for a carry into the 0th stage, but there may be a carry out of the 3rd stage if the sum exceeds 1111. Note that the Nth stage addition will only generate the correct sum when the carry from the N-1st stage is ready. This is a serial adder, also known as a ripple-through adder, as the final value of the sum will keep changing while the carry's propagate through the chain from the right.

GENERAL RING COUNTER IN MEMORY STRUCTURES:

The simplest way to implement a delay buffer is to use shift registers as shown in below Figure .If the buffer length is N and the word-length N is, then a total of N DFFs are required, and it consumes more and more power.

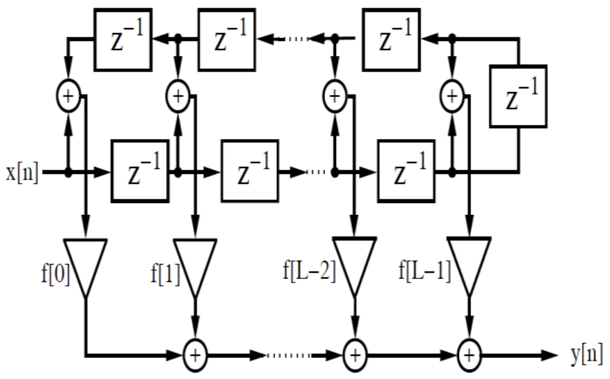


Ring Counter, shift register output fed back to input

Even though, activation of flip flops are only one at a time; Continuous power supply goes to all data flip-flops. So, lot of power is wasted while running this ring counter.

PROPOSED TECHNIQUE:

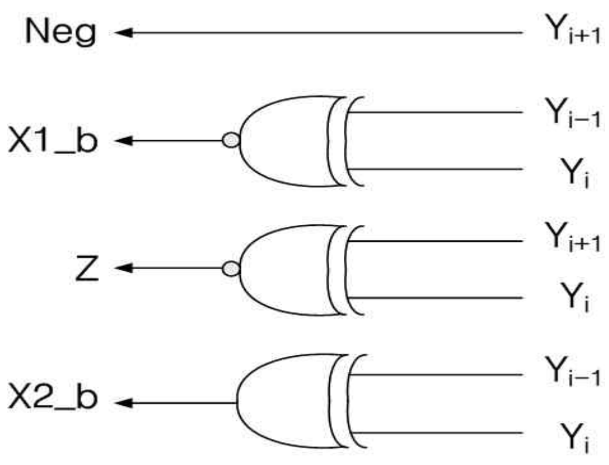
MODIFIED BOOTH ENCODER



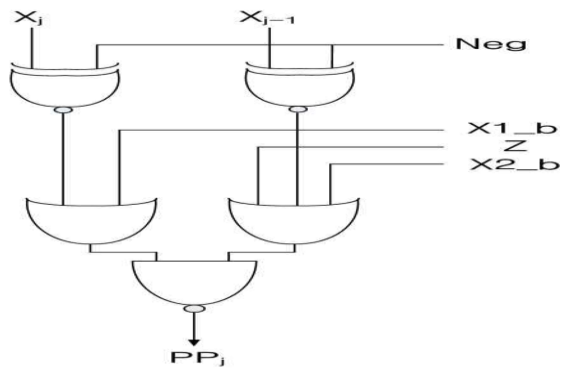
Multiplication consists of three steps:

1) The first step to generate the partial products
2) The second step to add the generated partial products until the last two rows are remained.
3) The third step to compute the final multiplication results by adding the last two rows.

The modified Booth algorithm reduces the number of partial products by half in the first step. We used the modified Booth encoding (MBE) scheme proposed in. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. The below table shows the rules to generate the encoded signals by MBE scheme and the figure represents the corresponding logic diagram. The Booth decoder generates the partial products using the encoded signals as shown in below figure.



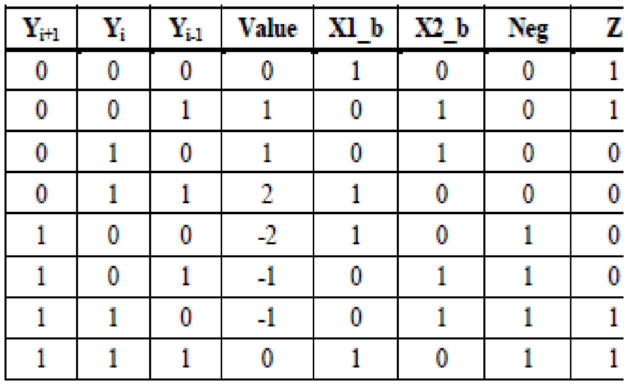
Booth Encoder



Booth Decoder

The figure shows the generated partial products and sign extension scheme of the 8-bit modified Booth multiplier. The partial products generated by the modified Booth algorithm are added in parallel using the Wallace tree until the last two rows are remained. The final multiplication results are generated by adding the last two rows. The carry propagation adder is usually used in this step.

Truth table for MBE Scheme:



Y_{i+1}	Y_i	Y_{i-1}	Value	X1_b	X2_b	Neg	Z
0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	2	1	0	0	0
1	0	0	-2	1	0	1	0
1	0	1	-1	0	1	1	0
1	1	0	-1	0	1	1	1
1	1	1	0	1	0	1	1

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

CARRY SAVE ADDER:

A carry-save adder is a type of digital adder, used in computer micro architecture to compute the sum of three or more n-bit numbers in binary. It differs from other digital adders in that it outputs two numbers of the same dimensions as the inputs, one which is a sequence of partial sum bits and another which is a sequence of carry bits. A Carry-Save Adder is just a set of one-bit full adders, without any carry-chaining. Therefore, an n-bit CSA receives three n-bit operands, namely A(n-1)..A(0), B(n-1)..B(0), and CIN(n-1)..CIN(0), and generates two n-bit result values, SUM(n-1)..SUM(0) and COUT(n-1)..COUT(0).

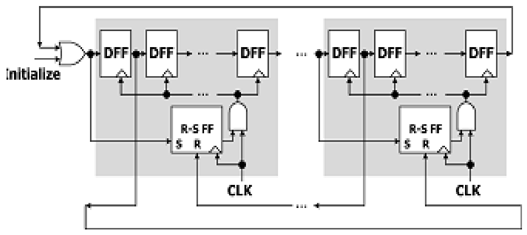
The most important application of a carry-save adder is to calculate the partial products in integer multiplication. This allows for architectures, where a tree of carry-save adders is used to calculate the partial products very fast. One 'normal' adder is then used to add the last set of carry bits to the last partial products to give the final multiplication result. Usually, a very fast carry-look ahead or carry-select adder is used for this last stage, in order to obtain the optimal performance.

RING COUNTER:

A ring counter is a type of counter composed of a circular shift register. The output of the last shift register is fed to the input of the first register.

- A straight ring counter or Over beck counter connects the output of the last shift register to the first shift register input and circulates a single one (or zero) bit around the ring. For example, in a 4-register one-hot counter, with initial register values of 1000, the repeating pattern is: 1000, 0100, 0010, 0001, 1000... . Note that one of the registers must be pre-loaded with a 1 (or 0) in order to operate properly.

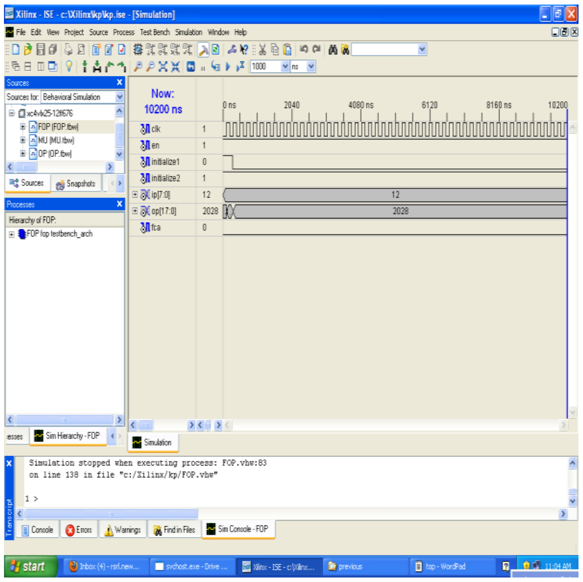
We make provisions for loading data into the parallel-in/ serial-out shift register configured as a ring counter below. Any random pattern may be loaded. The most generally useful pattern is a single 1.



Ring counter with SR flip-flops

The above block diagram shows the power controlled Ring counter. First, total block is divided into two blocks. Each block is having one SR FLIPFLOP controller.

SIMULATION RESULTS:



Signal	Value	
clk	1	
en	1	
initialize1	0	
initialize2	1	
ip[7:0]	12	12
op[17:0]	2028	2028
fca	0	

```
Simulation stopped when executing process: FOP.vhw:83
on line 138 in file "c:/Xilinx/kp/FOP.vhw"

1 >
```

In the above snap 12 is given as a input vector and the final response obtained is 2028.

SYNTHESIS RESULTS:

Techniques	Power at 50 MHz	Area (Gate count)
Existing	269 (mw)	42946
Proposed	228 (mw)	24110

Above results are synthesized on VIRTEX Family, XCV1020E Device and with a package of tq1022.

CONCLUSION:

Finally we implement the FIR filter with low power and low area. In order to reduce the power consumption and area, we use combinational booth multiplier, low power carry save adders, power optimized ring counter. These filters are compared with area and power with respect to other common elements and finally we come to a conclusion that our approach is the most effective for low cost and low power. The proposed FIR filters have been synthesized and implemented by using Xilinx ISE and Virtex-IV FPGA and the power is analyzed by using Xilinx Power analyzer.

REFERENCES:

1. N.Shibata, M.Watanabe, and Y.Tanabe, ìAcurrent-sensed high-speed and low-power, rst-inrst-out memory using a word line / bit-line, swapped dual-port SRAM cell, iIEEE J.Solid-State circuits, Vol.37, No.6, pp.735ñ750, Jun.2002.
2. E.Sutherland,ìMicropipelines,îCommun. CM,vol.32,no.6,pp.720ñ738,Jun.1989.
3. K. Zhang, U. Bhattacharya, Z. Chen, F. Hamzaoglu, D. Murray, N. Vallepalli, Y. Wang, B. Zheng, and M. Bohr, ìSRAM design on 65nm CMOS technology with dynamic sleep transistor for leakage reduction, îIEEE J.Solid State Circuits, vol.40,no.4,pp.895ñ901,Apr.
4. “Design and Implementation of Low Power Digital FIR Filters relying on Data Transition Power Diminution Technique” DSP Journal, Volume 8, pp. 21-29, 2008.
5. A. Senthilkumar, Natarajan, “FPGA Implementation of Power Aware FIR Filter Using Reduced Transition Pipelined Variable Precision Gating,” Journal of Computer Science , pp. 87-94, 2008.
6. Uwe Meyer-Baese, “Digital Signal with Field Programmable Gate Arrays”, Springer-Verla Berlin Heidelberg 2007
7. Shibi Thankachan, “64 x 64 Bit Multiplier Using Pass Logic”, 2006.