Research Paper

# Design and implementation of high speed FIR filter using CSD multiplier algorithm on FPGA's

G. K. Dayanand[1*] , A. Jayarama[2] and C. Shantharama Rai[3]

[1]E & C Dept., Canara Engineering College, Mangalore, 574219 India

[2]Physics Dept., Sahyadri College of Engineering and Management, Adyar, Mangalore, 575007 India.

[3]E&C Dept., AJIET, Mangalore, 570006 India.

* Email: dayanand.gk@gmail.com

## Abstract

A Canonic Signed Digit(CSD) algorithm for Finite Impulse Response (FIR) filters is designed and simulated in Very High Speed Integrated Circuit Hardware Description Language (VHSIC-HDL) and implemented on Spartan 2E Field-Programmable Gate Array (FPGA). The customized Integrated Circuit (IC) can be used as a potential high speed FIR filter for various applications in the field of satellite, mobile communication etc. The designed FIR filter consumes total memory of 75320 Kilobytes with frequency of 146.370MHz in Xilinx Sparten 2E kit

Keywords: FPGA, CSD , FIR Filter, VHDL.

## 1Introduction

The modern technology is moving towards high speed speech processing, image compressing, digital modulation techniques which involve processing of digital signals and applications like optical fiber or wireless communication. The processing, compression and modulation of digital signal should be very fast, so that higher speed transmission can be possible [1, 2].

Digital signals can be stored and removed, transmitted and received, processed and manipulated, all virtually without error. Two major advantages that distinguish DSP from other application oriented general purpose computation processing are the real-time throughput, speed and data driven property. DSP filters are signal conditioners. Finite Impulse Response (FIR) digital filters, widely used in signal processing , are one of two primary types of digital filters used in digital signal applications; the other type is feedback oriented Infinite Impulse Response filter (IIR).

Digital filters can be implemented with less hardware using a CSD representation. For multipliers computation is commonly done by addition and shift methods. In CSD method, number of additions and shifts operations implemented for a multiplier depends on the number of non-zero binary bits in the binary digits representation. Extracting common sub-computations between the filter coefficients can minimize the number of additions and shift operations [3, 4]. This realizes an area-efficient implementation of the filter. Coefficient transform are very important in realizing area-efficient multiplier. By applying coefficient transform in combination with subexpression and pre-computation technique, we can save many additions and subtractions. Strength reduction at the algorithm level can be used to reduce the number of additions. An efficient algorithm is represented by the reduced finite digital filter coefficients, which are represented by a Canonical Signed Digits numbers representation, i.e., numbers represented as sums or subtraction of power-of-two [5].

Recent interest is in using minimum adder/subtracter units for multiplier. The design and implementation of digital filters has extensively contributed to future development using CSD number system [6,7]. The integer coding schemes considered here belong to a class of numbers called fixed point. Fixed point refers to the fact that the binary point is found in fixed location. There exist a few other fixed point systems, which hold some interest to the digital signal processing scientists. One of such system is called the canonical signed digit code. In the early 1950's, this code enjoyed a high degree of popularity.

A common method of carrying out multiplication is by addition and shifting operations of constant numbers. By optimized hardware utilization, it is possible to alter this method by addition or subtractions. The CSD method has the benefits of reducing the number of additions/subtractions, in addition to effortlessly handling negative multipliers; results are obtained by representing the multiplier in CSD form. In this paper we report, design of high-speed computation FIR filters using CSD based multiplier implemented on FPGA's for DSP applications. DSP systems can be realized using programmable processors or custom designed hardware circuits fabricated using very large scale integration (VLSI) circuit technology.

### 1.1CSD algorithm

An algorithm for computing the CSD format of a W-bit number is presented below. Denoted below is the two's compliment representation of the number A as A=$\hat{a}_{w-1}$. $\hat{a}_{w-2}$, .......$\hat{a}_1$,$\hat{a}_0$ and its CSD representation of the number A=$a_{w-1}$. $a_{w-2}$, .......$a_1$,$a_0$, the conversion is illustrated using the following iterative algorithm:

```
$\hat{a}_{-1}=0$
$\gamma_{i-1}=0$
$\hat{a}_w= \hat{a}_{w-1}$
for(i= 0 to W-1)
{ $\theta_i = \hat{a}_i$ xor $\hat{a}_{w-1}$
$\gamma_i$ = not($\gamma_i$) $\theta_i$
$\hat{a}_i = (1-2\ \hat{a}_{i+1})\ \gamma_i$ }
```

An example for CSD algorithm is given in table 1; the input number is 1.01110011 and its CSD representation is given by $a_i$.

Table 1: CSD representation $a_i$

| i | W | W-1 | ... | | | | | | | 0 | -1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\hat{a}_i$ | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| $\theta_i$ | | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | |
| $\gamma_i$ | | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $(1-2\hat{a}_{i-1})$ | | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | |
| $\hat{a}_i$ | | 0 | 1 | 0 | 0 | -1 | 0 | 1 | 0 | -1 | |

## 2Design and Implementation

The applications of high speed digital filtering (> 10MHz) normally necessitate the use of defined application specific integrated circuits (ASIC's). User programmable signal processor cannot handle such huge computation of sample rates devoid of an extreme amount of parallel or hybrid processing which is expensive. For user defined applications flexibility of a multiplier is unnecessary. Hence , more optimized filter modules can be obtained by implementing each filter coefficient with selected arithmetic elements which recognize a particular filter coefficient. Multiplication process by power-of-two is obtained in user defined hardware structure by left-shifting or right-shifting of the data bus to a proper number of adders/substractors in each filter taps for digital filters along with suitable shifting operation. The ensuing hardware complication is a minimal fraction of the intricacy of a general-purpose multiplier and thus a more numbered complex of filter taps can be designed and implemented onto a single IC.

The numbers representing sum/subtractor of power-of-two is properly known as radix-2 signed binary digit code. The radix-2 representing signed-digit in fractional number x has following general form as shown in equation 1.

$$x = \sum_{k=0}^{l} a_i 2^{-pi} \qquad (1)$$

Where $a_i \in \{0,1,-1\}$ and pi $\in \{0,1,. . . . ..,N\}$. The representation has N+1 total number digits and l nonzero digit. In general there are many signed digits symbols for a given number.

## 3Results and Synthesis Report

CSD algorithm was successfully continued for filter containing 30 coefficients of 16 bits binary chosen at random. The results are shown in table-2 and the coefficients are represented by 2's power format as shown in table 3.

The entity of design is shown in figure 1 with input output ports; master_clk is clock signal for entire design; cl_en is clock enable and rst is reset pin. The fir_in is n-bit input samples and fir_out is n-bit samples convoluted with coefficients. VHDL code for FIR filter using CSD algorithm has been successfully implemented on FPGA (Xilinx Spartan 2E); the place and mapping is as shown in figure 2. The output waveforms of 30-tap low pass filter is shown in figure 3. Synthesis and timing summary reports are shown in in table 4 , the circuit will work at maximum frequency of 146.370MHz; Total memory usage is 75320 kilobytes.

Table 2: Original FIR filter coefficients.

| | |
|---|---|
| **h[0]** | 0.008021 |
| **h[1]** | 0.009826 |
| **h[2]** | 0.011483 |
| **h[3]** | 0.008938 |
| **h[4]** | 0.000996 |
| **h[5]** | -0.011612 |
| **h[6]** | -0.025746001 |
| **h[7]** | -0.036224999 |
| **h[8]** | -0.037126999 |
| **h[9]** | -0.023705 |
| **h[10]** | 0.005686 |
| **h[11]** | 0.048388001 |
| **h[12]** | 0.097474001 |
| **h[13]** | 0.143259004 |
| **h[14]** | 0.175752997 |
| **h[15]** | 0.187501997 |
| **h[16]** | 0.175752997 |
| **h[17]** | 0.143259004 |
| **h[18]** | 0.097474001 |
| **h[19]** | 0.048388001 |
| **h[20]** | 0.005686 |
| **h[21]** | -0.023705 |
| **h[22]** | -0.037126999 |
| **h[23]** | -0.036224999 |
| **h[24]** | -0.025746001 |
| **h[25]** | -0.011612 |
| **h[26]** | 0.000996 |
| **h[27]** | 0.008938 |
| **h[28]** | 0.011483 |
| **h[29]** | 0.009826 |

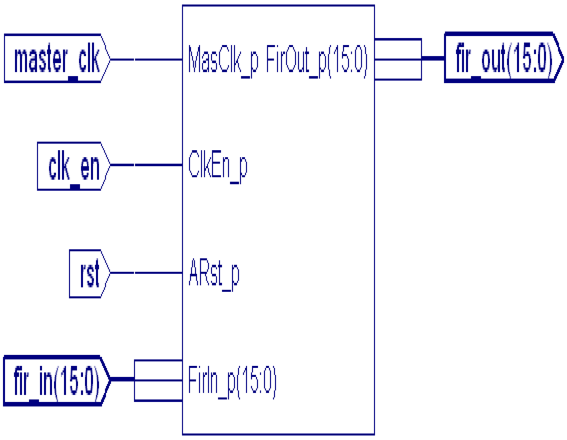

Figure 1: Entity of FIR filter

Table 3: CSD Filter Coefficients

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| h[0] | 2^-5 | 0.00390625 | -0.001953125 | 2^-15 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[1] | 2^-5 | -0.00012207 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[2] | 2^-5 | 2^-8 | 2^-10 | 2^-12 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[3] | 2^-5 | 0.00390625 | 2^-10 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[4] | 2^-8 | 0.000976563 | 2^-12 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[5] | -2^-5 | 0.00390625 | -0.001953125 | 2^-12 | 2^-14 | 2^-15 | 0^0 | 0^0 | 0^0 | |
| h[6] | -2^-4 | 0.015625 | 0.00390625 | 2^-11 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[7] | -2^-3 | 2^-7 | 2^-9 | 2^-11 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[8] | -2^-3 | 2^-7 | -0.000488281 | 2^-14 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[9] | -2^-4 | 0.015625 | 2^-8 | 0.000976563 | 2^-13 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | |
| h[10] | 2^-6 | 2^-9 | 2^-11 | 6.10352E-05 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[11] | 2^-3 | 2^-5 | 0.00390625 | 2^-10 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[12] | 2^-2 | 2^-4 | 0.00390625 | 2^-12 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[13] | 2^-1 | -0.03125 | 0.015625 | 2^-11 | 2^-12 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[14] | 2^-1 | 2^-4 | 0.00390625 | -0.001953125 | 2^-13 | 2^-15 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[15] | 2^-1 | 2^-3 | -0.03125 | 2^-12 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[16] | 2^-1 | 2^-4 | 0.00390625 | -0.001953125 | 2^-13 | 2^-15 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[17] | 2^-1 | -0.03125 | 0.015625 | 2^-11 | 2^-12 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[18] | 2^-2 | 2^-4 | 0.00390625 | 2^-12 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 |
| h[19] | 2^-3 | 2^-5 | 0.00390625 | 2^-10 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[20] | 2^-6 | 2^-9 | 2^-11 | 6.10352E-05 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[21] | -2^-4 | 0.015625 | 2^-8 | 0.000976563 | 2^-13 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | |
| h[22] | -2^-3 | 2^-7 | -0.000488281 | 2^-14 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[23] | -0.125 | 2^-7 | 2^-9 | 2^-11 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[24] | -2^-4 | 0.015625 | 0.00390625 | 2^-11 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[25] | -0.03125 | 0.00390625 | -0.001953125 | 2^-12 | 2^-14 | 2^-15 | 0^0 | 0^0 | 0^0 | |
| h[26] | 2^-8 | 0.000976563 | 2^-12 | -3.05176E-05 | 0^0 | 0^0 | 0^0 | 0^0 | 0^0 | |
| h[27] | 0.008938 | | | | | | | | | |
| h[28] | 0.011483 | | | | | | | | | |
| h[29] | 0.009826 | | | | | | | | | |

## 4Conclusion

The CSD algorithm is successfully implemented on Spartan 2E FPGA . The major factors that influence our choice of a specific realization are Computational complexity, Memory requirements and finite-word-length effects. From above method there is significant reduction in the filter complexity, minimal hardware and high speed of computation. Complexity of filter increases, with minimal changes in CSD representation which reduces the number of steps involved. There is significant frequency response also obtained from this algorithm. This method may be used in concurrence with other algorithm techniques for optimized digital FIR filter complexity like pre- filter module or the interpolated digital FIR filters. Further, same algorithm can be implemented for infinite response digital filters. Clock frequency (masclk_p) =146 MHz; Enable (cken_p) =1; Reset (arst_p)=0; Filter input( rin_p)= Sample in; Filter output ( rout_p)

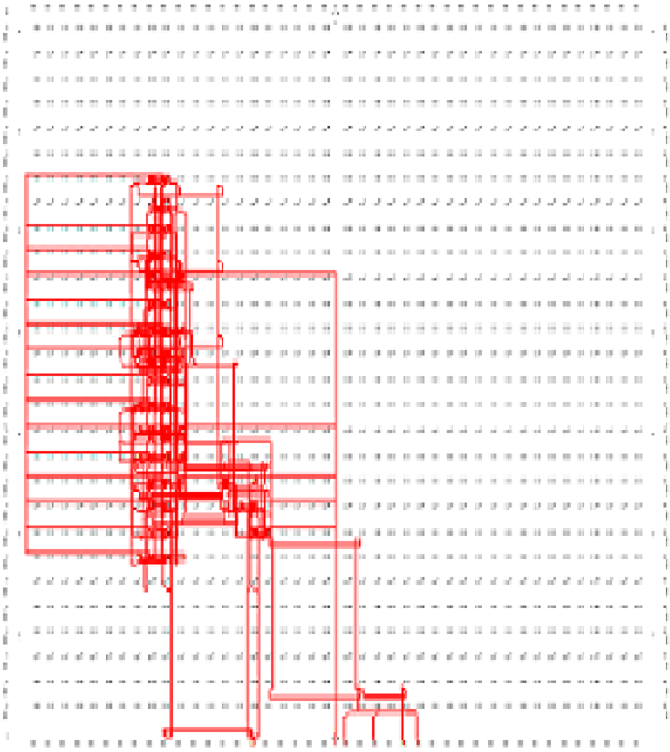

Figure 2: Placement & routing on Spartan 2E FPGA

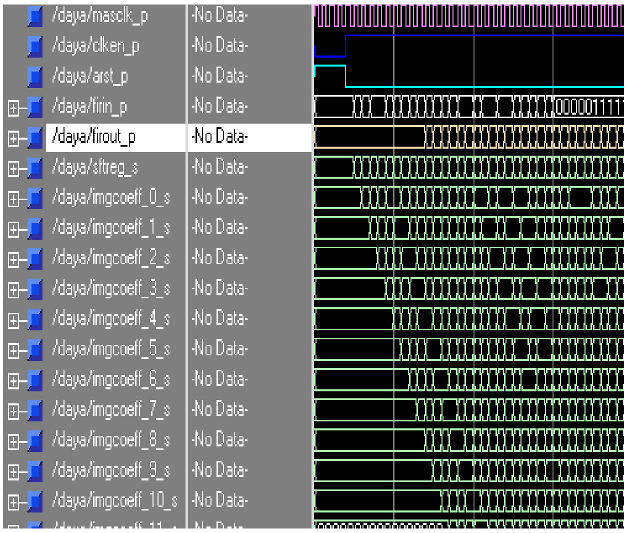

Figure 3: Output waveform of FIR pass filter

Table 4: Synthesis report and Device utilization summary

| Selected Device : 2s100pq208-5 Speed Grade: -5 | |
|---|---|
| Number of Slices | 821 out of 1200 (68%) |
| Number of Slice Flip Flops | 1436 out of 2400(59%) |
| Number of 4 input LUTs | 947 out of 2400(39%) |
| Number of bonded IOBs | 34 out of 144(23%) |
| Number of GCLKs | 1 out of 4(25%) |
| Timing Summary | |
| Minimum period | 6.832ns |
| Max Frequency | 146.370MHz |
| Minimum input arrival time before clock | 16.513ns |
| Maximum output required time after clock | 7.999ns |
| Total memory usage | 75320 Kilobytes |

# References

[1]F. J. Taylor, "Digital Filter Design Hand book", Marcel inc., NYC,1984.

[2]K. Keshab Pahri, VLSI Digital Signal Processor Design and Implementation", WSE Wiley 2004 Edition.

[3]G. K. Ma and F. J. Taylor, Multiplier policies for digital Signal processing , IEEE ASSP Mag.vol 21, pp. 6-20, 1990.

[4]W. J. Oh and Y. H. Lee, Implementation of programmable multiplier less FIR filters with powers-of-two coefficients, IEEE Trans. On CAS II, vol.42, pp.553-555, 1995.

[5]R. Hartley, Optimization of canonic signed digit multipliers for filter design , Int. symp. circuits syst., Singapore, pp. 1992-1995, 1991.

[6]M. Reid Hewlitt, Canonical signed digit representation for Fir digital Filters , IEEE Workshop on Signal Processing Systems, pp. 416-426, 2000.

[7]Murko Kosunen and Kuri Halonen, Multiplier and shift using signed digit representations , Patent No. US 7,257,609 B1, Date of Patent: Aug 14, 2007.