High performance and Low power FIR Filter Design Based on Sharing Multiplication

Jongsun Park, Woopyo Jeong, Hunsoo Choo,
Hamid Mahmoodi-Meimand, Yongtao Wang, Kaushik Roy
School of Electrical and Computer Engineering, Purdue University,
West Lafayette, IN 47906, USA
+1-765-494-3372
{jongsun, jeongw, chooh, mahoodi, yw, kaushik} @ecn.purdue.edu

ABSTRACT
We present a high performance and low power FIR filter design, which is based on computation sharing multiplier (CSHM). CSHM specifically targets computation re-use in vector-scalar products and is effectively used in our FIR filter design. Efficient circuit level techniques: a new carry select adder and conditional capture flip-flop (CCFF), are also used to further improve power and performance. The proposed FIR filter architecture was implemented in 0.25 μm technology. Experimental results on a 10 tap low pass CSHM FIR filter show speed and power improvement of 19% and 17%, respectively, with respect to an FIR filter based on Wallace tree multiplier.

Keywords
Computation sharing, FIR filter design, high performance and low power carry select adder, conditional capture flip-flop

1. INTRODUCTION
Recent advances in mobile computing and multimedia applications demand high-performance and low-power VLSI Digital Signal Processing (DSP) systems. One of the most widely used operations in DSP is *finite impulse response* (FIR) filtering. As shown in the equation below, FIR filtering operation involves an inner product of coefficient vector $\boldsymbol{C}$ with input signals $\boldsymbol{x}$.

$$y(n) = \sum_{k=0}^{M-1} c_k \bullet x(n-k)$$

Several techniques have been proposed in literature to achieve high performance and low power implementation of FIR filters with fixed coefficients. *Canonical-sign-digit* [4] and *distributed arithmetic* [5] are widely used in the FIR filter design with fixed coefficients. Using those techniques, the FIR filtering operation can be simplified to add and shift operations. However, for FIR filter design with programmable coefficients, dedicated multipliers are usually used and filter design techniques mentioned may not be applicable.
In this research we propose high-performance and low-power implementation for FIR filter with programmable coefficients.

ISLPED '02, August 12-14, 2002, Monterey, California, USA.

The FIR filter architecture is based on the *Computation sharing multiplier* (CSHM) [1, 2], which targets the reduction of redundant computations in FIR filtering operation.

We also present the circuit level techniques for carry select adder and flip flop, which are effectively used in the FIR filter implementation. In the CSHM structure, adders are critical for high performance. In order to achieve high performance with low power consumption, a new Carry Select Adder is presented. Flip-flops are also crucial elements from both a delay and power standpoint. Conditional capture flip-flop (CCFF) [11] is explained and used in our filter design. CCFF is a dynamic style flip-flop that has a negative set-up time and small clock-to-output delay. Moreover, depending on data switching activity, CCFF can statistically reduce the power consumption.

The rest of this paper is organized in five sections. Section 2 describes the architecture of FIR filter based on the CSHM. Section 3 presents the circuit level techniques. The new carry select adder and conditional capture flip-flop (CCFF) are presented in this section. We present FIR filter implementation in section 4 and section 5 shows the numerical results. Finally, conclusions are drawn in section 6.

2. FIR FILTER ARCHITECTURE
2.1. CSHM Algorithm and architecture
The *transposed direct form* (TDF) FIR filter is shown in figure 1. We notice that the TDF implements a product of the coefficient vector $\boldsymbol{C} = [c_0, c_1, \ldots\ldots, c_{M-1}]$ with the scalar $x(n)$ at time n. The input $x(n)$ is multiplied by all the coefficients c_0, c_1, c_2,……, c_{M-1} simultaneously. In the sequel, such products will be referred to as a *vector scaling* operation [2].

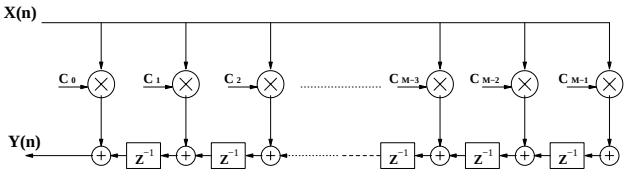


Figure1: Transposed direct form (TDF) FIR filter.

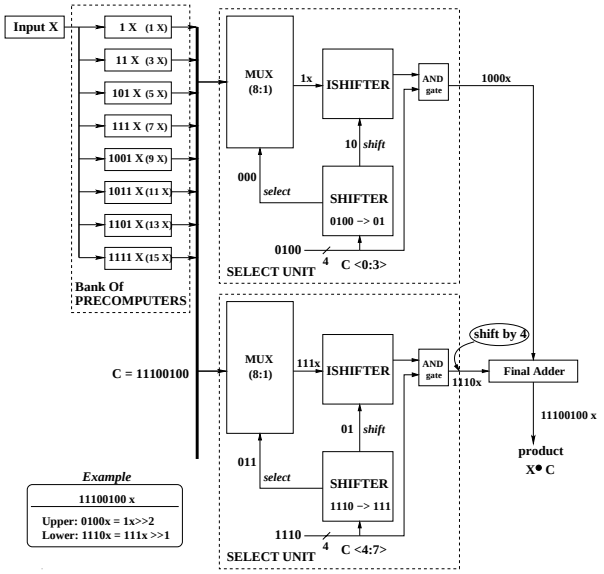


Figure 2: Computation sharing multiplier (CSHM) architecture

In the vector scaling operations, we can carefully select a set of small bit sequences so that the same multiplication result can be obtained by only add and shift operations. For instance, a simple vector scaling operation $[c_0, c_1] \bullet x$, $c_0 = 00110111$, $c_1 = 10001011$, can be decomposed as $c_0 \bullet x = 2^4 \bullet (0011) \bullet x + (0111) \bullet x$, $c_1 \bullet x = 2^7 \bullet (0001) \bullet x + (1011) \bullet x$. If x, $(0011) \bullet x$, $(0111) \bullet x$ and $(1011) \bullet x$ are available, the entire multiplication process is reduced to a few add and shift operations. We refer to these chosen basic bit sequences as *alphabet*s. Also, an *alphabet set* is a set of *alphabets* that spans all the coefficients in vector ***C***. In the above example, the *alphabet set* is {0001, 0011, 0111, 1011}.

Figure 2 shows CSHM architecture based on the algorithm explained above. CSHM is composed of a precomputer, Select units and final adders (S & A). The precomputer performs the multiplication of alphabets with input x. The Select unit and final adder (S & A) perform appropriate select/shift and add operations required to obtain the multiplication output. In order to cover every possible coefficient and perform general multiplication operation, we used 8 alphabets {1,3,5,7,9,11,13,15}. Figure 3 shows an example 17×17 CSHM, which is used in our FIR filter implementation.

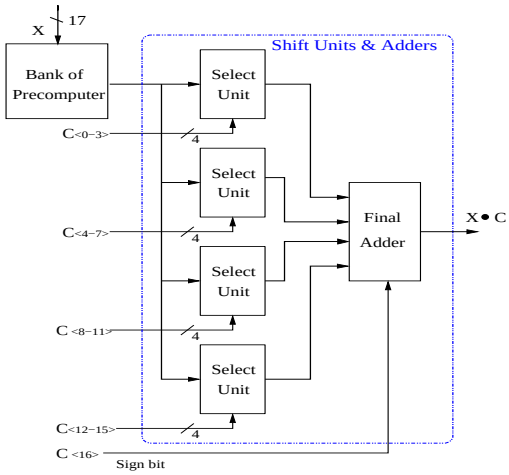


Figure 3: 17×17 CSHM structure.

2.2 FIR filter based on CSHM Architecture

In the *transposed direct form* (TDF) FIR filter shown in figure 1, CSHM algorithm is effectively used to reduce computations. As shown in figure 4, The FIR filter using CSHM consists of one precomputer and M Select units and Final adders (S & A)'s, where M represents the number of taps in FIR filter. We can easily notice from the figure that the precomputer outputs are shared by all the S & A's. In other words, the computations $\alpha_k \bullet x$, $k = 0, 1, 2....8$, are performed only once for all k's and these values are shared by all the Select units for generating $c_k \bullet x$, $i=0, 1, 2, 3,....$ The CSHM scheme efficiently removes the redundant computation in FIR filtering operation, which leads to low power and high performance design.

3. CIRCUIT LEVEL TECHNIQUES

3.1 High performance low power Carry Select Adder

As shown in figure 3, the final adder, which merges four vectors from Select units, is the critical component in terms of performance. A new Carry Select Adder, which can be efficiently used in the final adder implementation, is proposed in this subsection. The Carry Select Adder used in our design has good noise-immunity and achieves high performance with low power consumption. The architecture is similar to a Carry Select Adder using DTSL (Dual Transition Skewed Logic), where dual paths are used carry propagation – one path is used for fast propagation of rising

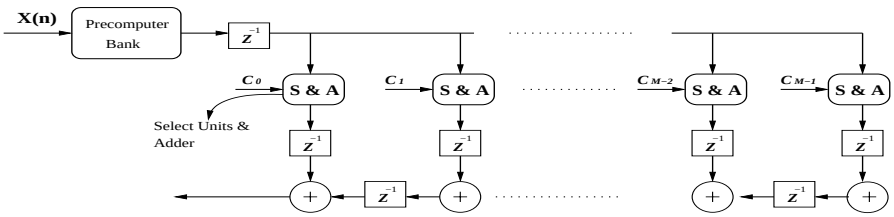


Figure 4: The architecture of FIR filter based on CSHM.

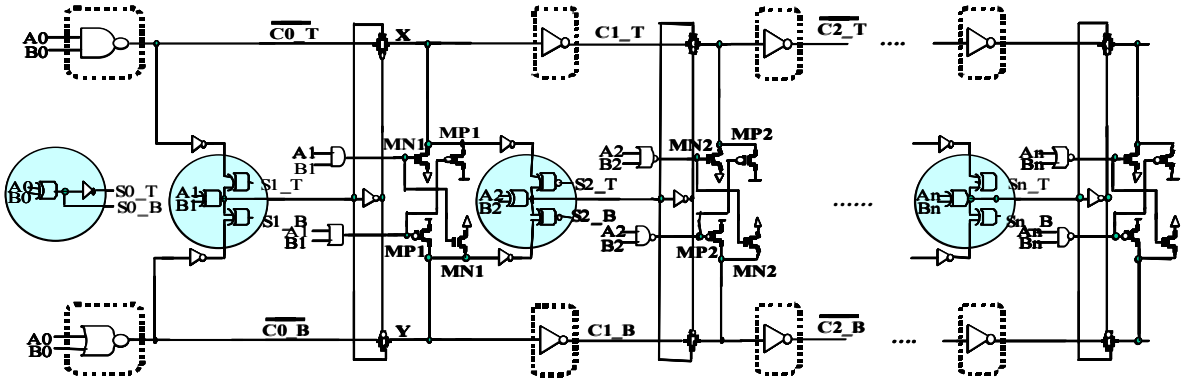


Figure 5: Block diagram of the Carry Select Adder used in the FIR filter.

transition, while the other path is used for fast propagation of falling transition [6].

The Carry Select Adder used for our FIR filter implementation has the same architecture as the conventional DTSL Carry Select Adder [6], however, a simplification has been applied that the skewed carry propagation logic circuits have been replaced with static CMOS inverters. This simplification reduces the complexity of carry propagation paths, and therefore, makes the control logic circuitry simpler and improves performance as compared to the conventional DTSL Carry Select Adder. Moreover, the power consumption of this new Carry Select Adder are comparable to those of its static CMOS counterpart because of the simple carry propagation paths of proposed Carry Select Adder.

Figure 5 shows the implementation of the Carry Select Adder used in our FIR filter. It consists of two data paths for carry propagation, logic for generating SUM, and control logic. Control logic consists of transmission gates (X, Y) between each inverter for carry propagation on the data path, switching transistors (MN, MP), and some static CMOS gates to control the transmission gates and switching transistors. The logic in the circle is for generating SUM. As shown in Figure 5, the carry propagation logic of each block of Carry Select Adder has two data paths: one has '0' as its CARRY input and the other has '1' as its CARRY input.

If inputs Ai's are different from Bi's ($Ai \neq Bi$, for all i = 0 to n), transmission gates X, Y will turn on and the switching transistors (MNi, MPi) will be disabled. Carry-out of the first stage will propagate to the last stage. Therefore, the carry propagation delay is the largest under such condition. Under such inputs the Carry-outs will be inversions of Carry-ins for every stage because each Carry-in goes through one inverter and one transmission gate.

However, if any Ai is equal to Bi at Stage i (i = 0 to n), the Carry-outs on both paths from that stage to the last stage (i ~ n) will be the same, and determined only by inputs Ai and Bi regardless of Carry-outs of the previous stage. This means that the carry propagation starts simultaneously at the first stage and the ith stage. Hence, in this case, the propagation delay of Carry Select Adder is the same as one of the carry propagation delays from the first stage to (i-1)th stage and from ith stage to the last stage. Then, we have to switch Carry-in of the next stage ([i+1]th) to low or high depending on the value of Ai and Bi. For example, let us assume A1=B1=0 at Stage 1, then the outputs of AND and OR gates in the control logic of this stage will be 0, and PMOS switching transistors (MP1) will turn on. Therefore, the Carry-out (C1_T, C1_B) at that stage will be low regardless of Carry-in (C0_T, C0_B) of Stage 1. Hence, we do not need to wait for the Carry-in to propagate to the output node of Stage 1, i.e. when inputs A1, B1 of Stage 1 are set, we can switch Carry-in of the next stage (Stage 2) immediately to low after turning off the transmission gates on the data path.

Similarly, if A1=B1=1 at Stage 1, then we change Carry-in of the next stage (Stage 2) to high. For such cases, the total propagation delay will be shorter than the total delay of the previous case ($Ai \neq Bi$, for all i = 0 to n) because the time taken to switch Carry-in of the next stage (Stage 2) is shorter than the time in which Carry-in of the first stage (Stage 0) propagates to the Carry-in node of the Stage 2 having A2, B2 as inputs. In Stage 2, NAND and NOR gates are used instead of AND and OR. The operation is similar to that of the previous stage.

3.2 Flip Flop Design

Traditionally, Transmission-Gate Flip-Flop (TGFF) has been used in standard cell design [7]. TGFF has a fully static master-slave structure by cascading two identical pass-gate latches and provide a short clock-to-output latency. However, it has a poor data-to-output latency because of positive set-up time. It also requires two phases of clock that can cause a problem with data feed-through when there is a skew between them and it has a relatively large clock load.

Considering the fact that in critical paths the flip-flop delay is the sum of set-up time and clock-to-output delay, dynamic latches have less total delay than master-slave latch pairs, which are fully static. Examples are hybrid latch flip-flop (HLFF) [8], semi-dynamic flip-flop (SDFF) [9], and sense amplifier based flip-flop (SAFF) [10]. They can also provide advantages such as absorbing the clock skew, reducing the clock load, and embedding logic functions into themselves. However, they are inefficient as far as power consumption is concerned. This is because of the fact that in moderate and low data switching rate these flip-flops can have unnecessary internal transitions that can lead to substantial increase in total power consumption. Conditional capture flip-flop (CCFF) [11], figure 6, removes this problem by addition of the internal clock gating. In this way, CCFF achieves statistical power reduction by eliminating redundant transitions of internal nodes while maintaining soft clock edge and negative setup time properties.

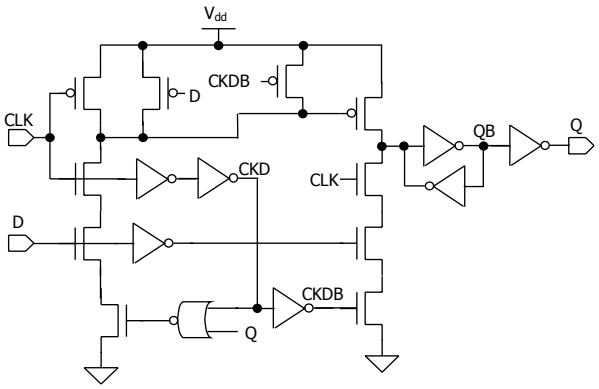


Figure 6. Conditional Capture Flip-Flop

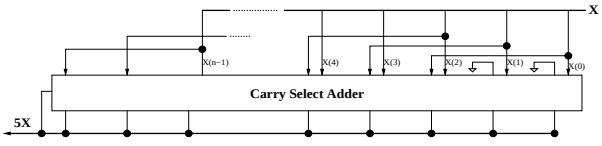
CCFF is used in our FIR filter implementation. The overall performance and power consumptions of designed TGFF, HLFF, and CCFF for different input patterns were simulated in TSMC 0.25um CMOS technology. The power consumption of CCFF has a large dependency on input pattern. CCFF can save 65% power with zero inputs switching activity as compared to the HLFF. When input changes at every other cycle, the power saving is nearly 14%. When the input changes at every cycle or the input switching activity is the maximum, which is very rare, the overall power consumption is comparable to the conventional designs.

4. FIR FILTER IMPLEMENTATION

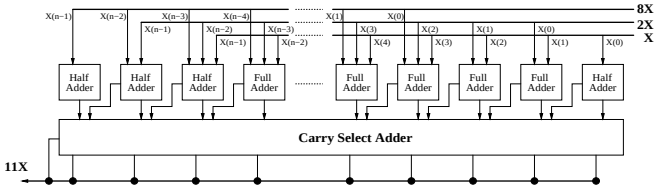
4.1. CSHM Implementation

A 17×17 CSHM is implemented using 0.25 µm TSMC library. As shown in figure 2, the CSHM is composed of precomputer, S & A. In our CSHM implementation, the input ***X*** is two's complement and coefficient C has a sign and magnitude format.

Precomputer*:* The multiplications, *1x, 3x, 5x, 7x, 9x, 11x, 13x, 15x,* performed by the precomputer are simply implemented using the new Carry Select Adder, which is mentioned in section 3.1. Figure 7 shows the basic structures of *5x* and *11x* and figure 8 shows the whole precomputer structure.



(a) 5X (0101X) = 100X(<<2) + 1X



(b) 11X (1011X) = 8X (1000X) + 2X (10X) + X

Figure 7 : Precomputer (5x, 11x) architecture.

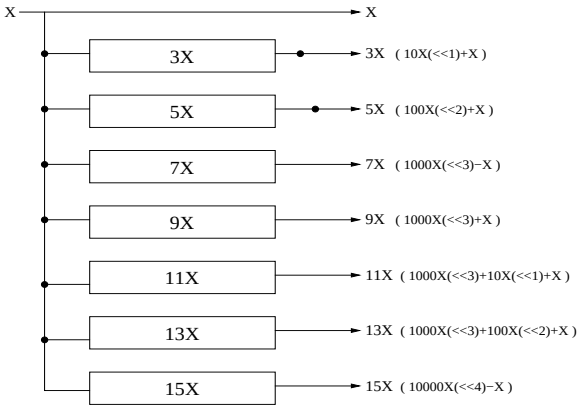


Figure 8: The precomputer structure

Select Unit*:* As shown in figure 2, Select unit is composed of SHIFTER, MUX, ISHIFTER and AND gates. Since SHIFTER is directly connected to the coefficients, it does not lie on the critical path. Static CMOS design with minimum size is used for SHIFTER implementation. ISHIFTER lies on the critical path and the maximal shift width is 3 bits. A barrel shifter [3] is used since signal has to pass through at most one transmission gate in the barrel shifter. MUX using pass transistor logic was implemented to achieve a compact and high-speed design.

Final Adder*:* The final adder is the largest component in the S & A, which sums the outputs of four Select units. Carry save array [3] and the new Carry Select Adder proposed in section 3.1 are used for high performance as shown in figure 9. The XOR gate array in the middle is used for the two's complement data format.

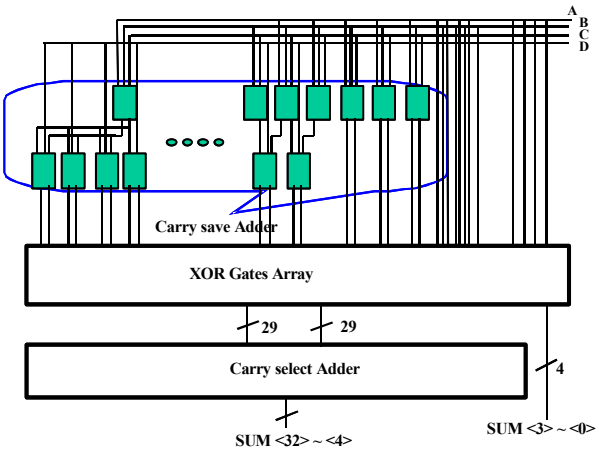


Figure 9: Final adder architecture

4.2.FIR filter Implementation

The 10 tap FIR filter using CSHM is implemented for fabrication. The structure of FIR filter is shown in figure 4. As mentioned before, the precomputer outputs are performed only once and shared by 10 S & A, which leads to low power design.

The clock network in FIR filter has been implemented using an H-tree structure. Figure 10 shows the timing of the critical path and the clock network. Path 2 is the critical path of the

design and its delay is almost twice the delay of path 1 and path 3. Therefore, pipeline stages are unbalanced in terms of delay. Insertion of more pipeline stages was not possible because of the overhead of latching a large number of internal signals in the select/shift and add unit. Time borrowing and slack passing are powerful methods for improving performance in unbalanced pipeline stages [12]. Since the delay of path 3 is much less than the critical path delay, path 2, by applying a negative clock skew some time can be borrowed from path 3 to path 2 as shown in the timing waveforms of figure 10. The minimum clock cycle can be expressed as

$$t_{cycle} = t_{clk-to-Q} + t_{critical} - t_{skew}.$$

Therefore, by using a negative clock skew, the clock cycle time can be reduced. Since flip-flops used in our design are CCFF's having a negative setup time, setup time does not contribute to the cycle time.

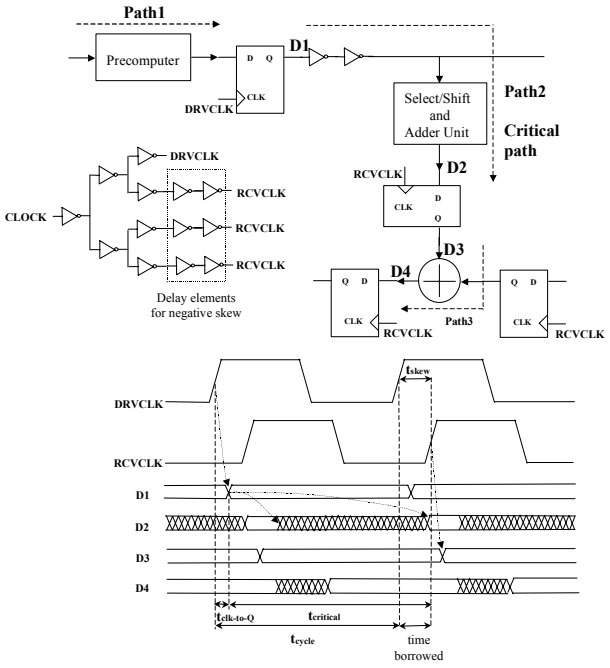


Figure 10: Clock network and timing of critical path

Figure 11 shows the layout of the FIR filter design. Floorplaning was done to minimize the total interconnect lengths especially for global signals. The layout was sent for fabrication to MOSIS.

We also implemented the 10 tap FIR filters using Wallace tree multiplier (WTM) and Carry save array multiplier (CSAM) for comparison. A 5:3 compressor was used in the WTM.

5. NUMERICAL RESULTS

Table 1 shows the clock cycle and power of the FIR filters using different multipliers. As shown in the table, FIR filter using CSHM has 19% and 43% performance improvement over FIR filter using WTM and CSAM, respectively. In terms of power consumption, CSHM scheme has 17% and 20% improvement with respect to FIR filter based on WTM and CSAM. The power results shown in table 1 are measured with the uniform clock cycle of 10ns.

Clearly, FIR filter using CSHM has one more pipeline stage than FIR filter based on WTM and CSAM. The performance of FIR filter using WTM and CSAM can be improved by adding additional pipeline stage. However, due to the tree structure of WTM and carry save array of CSAM, the number of flip-flops required to add additional pipeline stage is quite large. Moreover, as the number of filter taps increases, the increase in the number of flip-flops for additional pipeline stage will become significantly large. In the CSHM architecture, since precomputer outputs are shared by all the S&A's, we can add additional pipeline stages without incurring large latch overhead. The CSHM architecture has performance and power advantage through the additional pipelining and the sharing of the precomputer outputs by all the S&A's, respectively.

Table 1: Numerical results

	FIR using SHM	FIR using WTM	FIR using CSAM
Clock cycle [ns]	5.7	7.0	10
Power [mW]	286.6	344.3	357.1
Area [μm^2]	5.0×10^6	4.4×10^6	4.1×10^6

6. CONCLUSION

An FIR filter based on CSHM is implemented using 0.25 μm technology. The CSHM algorithm specifically targets reduction of redundant computation in FIR filtering operation. Using the CSHM scheme, the multiplications in vector scaling operation is significantly simplified to add and shift operations of *alphabets* multiplied by input *x*. These common computations are shared by the sequence of operations in vector scaling operations.

Adders and flip-flops are critical components in CSHM and FIR filter implementation. Circuit level techniques for adder and flip-flop are proposed and used in the full custom FIR filter implementation.

CSHM scheme and circuit level techniques helped us achieve low power and high performance FIR filtering operation. The proposed CSHM architecture is also applicable to adaptive filter and matrix multiplication implementation. The idea presented in this paper will help the design of DSP algorithms and their implementation for high performance and low power applications.

7. REFERENCES

[1] K. Muhammad, "Algorithmic and Architectural Techniques for Low Power Digital Signal Processing," Ph.D. thesis, Purdue University, 1999.

[2] J. Park, H. Choo, K. Muhammad, K. Roy, " Non adaptive and Adaptive filter implementation based on sharing multiplication," ICASSP, June 2000.

[3] Jan M. Rabaey, "Digital Integrated Circuits : A Design Perspective," Prentice Hall, New Jersey, 1996.

[4] H. Samueli, "An improved Search Algorithm for the Design of Multiplierless FIR filter with Powers-of-Two Coefficients," IEEE Trans. On circuits and systems, Vol. 36, No. 7, pp. 1044-1047, Jul. 1989.

[5] S. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," IEEE ASSP Magazine, July 1989, pp. 4 –19.

[6] W. Jeong, K, Roy, and C. Koh. "High-Performance Low-Power Carry Select Adder using Dual Transition Skwed Logic," *ESSCIRC*, 2001

[7] G. Gerosa *et al.*, "A 2.2 W 80 MHz superscalar RISC microprocessor," IEEE J. Solid-State Circuits, vol. 29, pp. 1440-1452, Dec. 1994

[8] H. Partovi *et al.*, "Flow-through latch and edge-triggered flip-flop hybrid elements," in Int. Solid-State Circuits Conference, Dig. of Tech. Papers, Feb. 1996, pp. 138-139

[9] F. Klass, "Semi-dynamic and dynamic flip-flops with embedded logic," in Symp. on VLSI Circuits, Dig. of Tech. Papers, June 1998, pp. 108-109.

[10] B. Nikolic *et al.*, "Sense amplifier-based flip-flop," in Int. Solid-State Circuits Conf., Dig. of Tech. Papers, Feb. 1999, pp. 282-283

[11] B. S. Kong *et al.*, "Conditional Capture Flip-Flop for Statistical Power Reduction," in *ISSCC Dig. Tech. Papers*, pp. 290-291, Feb. 2000

[12] H. Partovi, "Clocked storage elements," in Chandrakasan, A., Bowhill, W.J., and Fox, F. (eds.). Design of High-Performance Microprocessor Circuits. IEEE Press, Piscataway NJ, 2000, 207-23.

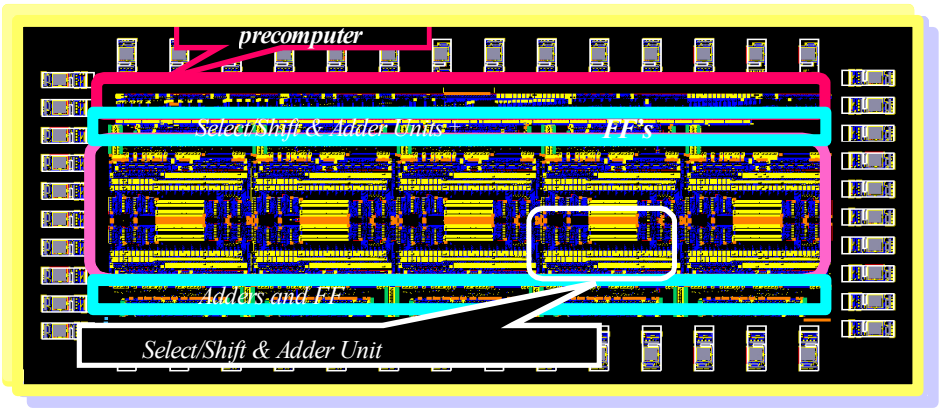


Figure11: Layout of FIR filter using CSHM