# Computation Sharing Programmable FIR Filter for Low-Power and High-Performance Applications

Jongsun Park, Woopyo Jeong, Hamid Mahmoodi-Meimand, *Student Member, IEEE*, Yongtao Wang, Hunsoo Choo, and Kaushik Roy, *Fellow, IEEE*

***Abstract*—This paper presents a programmable digital finite-impulse response (FIR) filter for high-performance and low-power applications. The architecture is based on a computation sharing multiplier (CSHM) which specifically targets computation re-use in vector-scalar products and can be effectively used in the low-complexity programmable FIR filter design. Efficient circuit-level techniques, namely a new carry-select adder and conditional capture flip-flop (CCFF), are also used to further improve power and performance. A 10-tap programmable FIR filter was implemented and fabricated in CMOS 0.25-$\mu$m technology based on the proposed architectural and circuit-level techniques. The chip's core contains approximately 130 K transistors and occupies 9.93 mm$^2$ area.**

***Index Terms*—Computation sharing, dual transition skewed logic, programmable finite impulse response (FIR) filter.**

## I. 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. The FIR filter performs the weighted summations of input sequences and is widely used in video convolution functions, signal preconditioning, and various communication applications. Recently, due to the high-performance requirement and increasing complexity of DSP and multimedia communication applications, FIR filters with large filter taps are required to operate with high sampling rate, which makes the filtering operation very computationally intensive.

Many previous efforts have focused on the design of FIR filters with low computational complexity since computational complexity reduction leads to high performance as well as low-power design. Canonical-signed-digit [3] and signed-power-of-two [4] coefficient representations are widely used in the parallel implementation of FIR filters. Using those techniques, the FIR filtering operation can be simplified to add and shift operations. Common subexpressions elimination [5], [6] and differential coefficients method [7], [8] also explore low-complexity design of FIR filters by minimizing the number of additions in filtering operations. However, most of the previous work has been limited to the design of FIR filters with fixed coefficients, allowing the hardware to be optimized only for a particular fixed coefficient set. For FIR filters with programmable coefficients, which are used in many applications like adaptive pulse shaping and signal equalization on the received data in real time, dedicated multipliers are generally used and the filter design techniques mentioned above may not be applicable since the modification of filter coefficients is difficult to accomplish in real time.

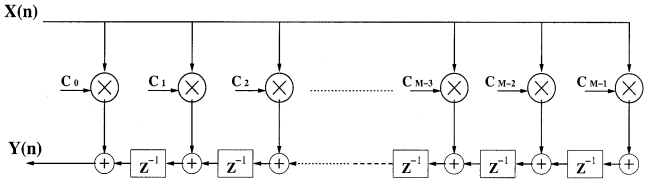


Fig. 1. Transposed direct form (TDF) FIR filter.

In this paper, we propose the high-performance and low-power design for FIR filters with programmable coefficients. Generally, FIR filtering can be expressed as a sequence of multiplication and addition operations. Since multiplication is most computationally expensive in FIR filtering, simplifying the multiplication operations is highly desirable for low-complexity design. In the proposed FIR filter architecture, the computation sharing multiplier (CSHM) [1] is efficiently used for the low-complexity design of the FIR filter. The main idea of CSHM is to represent the multiplications in the FIR filtering operation as a combination of add and shift operations over the common computation results. The common computations are identified and those are shared without additional memory area. This sharing property enables the computation sharing multiplier approach that achieves high performance and low power in FIR filter implementation.

In addition to the architectural-level technique, circuit-level techniques are also presented and used in the FIR filter implementation. In the CSHM structure, adders are critical for performance. A new carry-select adder, which is based on the dual transition skewed logic (DTSL), is presented and efficiently used in our filter implementation. The proposed carry-select adder based on DTSL is superior to the Domino-based carry-select adder in terms of power and performance. Flip-flops are also crucial elements from both a delay and power standpoint. Conditional capture flip-flop (CCFF) [15] is explained and used in our filter design. CCFF is a dynamic style flip-flop that has a negative setup time and small clock-to-output delay. Moreover, depending on data switching activity, CCFF also reduces the power consumption.

The remainder of this paper is as follows. Section II describes the architecture of the programmable FIR filter based on CSHM. Section III presents the circuit-level techniques used for the

Manuscript received May 13, 2003; revised September 15, 2003.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: jongsun@ecn.purdue.edu).

Digital Object Identifier 10.1109/JSSC.2003.821785

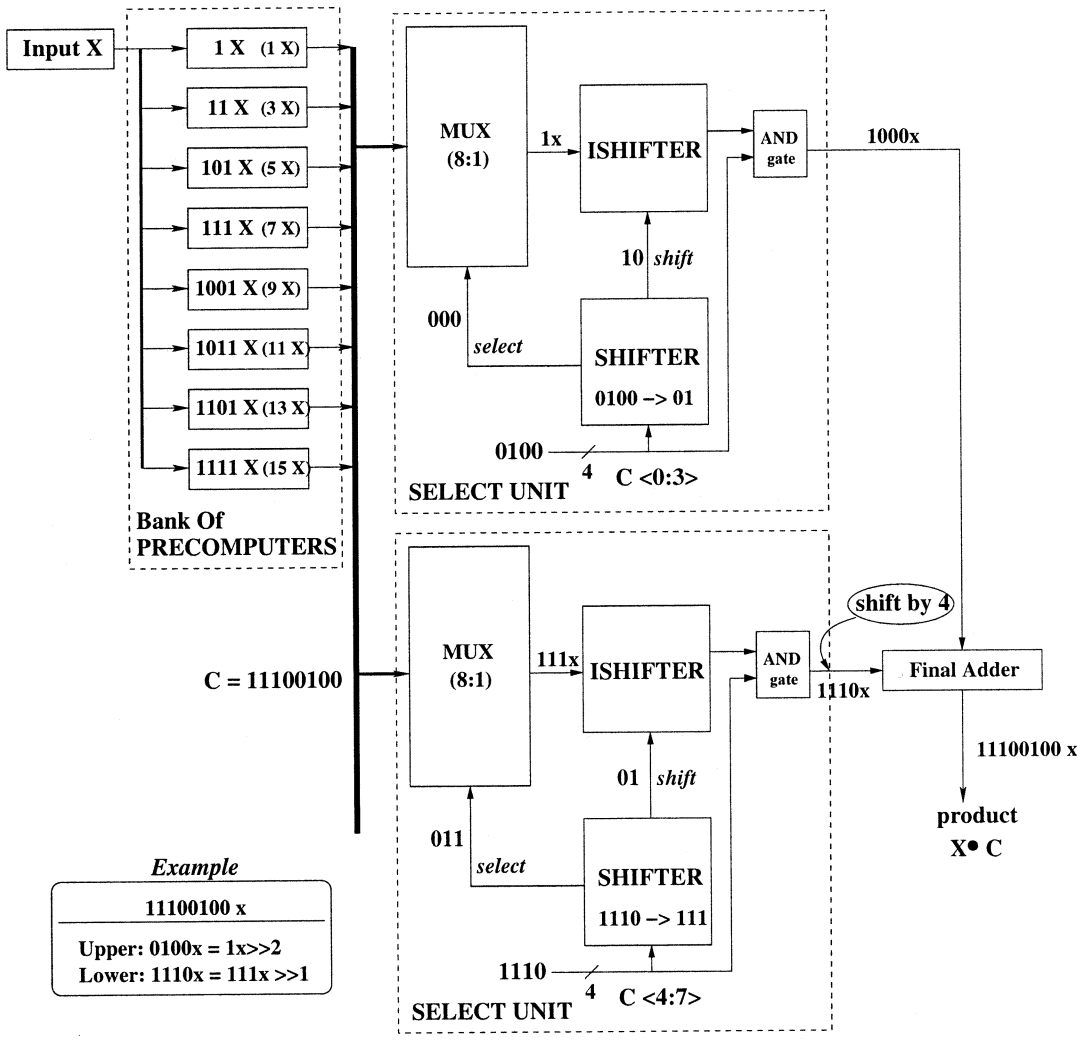


Fig. 2. Computation sharing multiplier (CSHM) architecture.

physical implementation. The new carry-select adder and CCFF are described in Section III. Section IV shows the simulation and measurement results. Finally, conclusions are drawn in Section V.

## II. FIR Filter Architecture

### A. CSHM Algorithm and Architecture

The input–output relationship of the linear time invariant (LTI) FIR filter can be described as

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

where $M$ represents the length of the FIR filter, the $c_k$'s are the filter coefficients and $x(n-k)$ denotes the data sample at time constant $(n-k)$.

Fig. 1 shows a transposed direct form (TDF) implementation of the FIR filter. We notice that the TDF implements a product of the coefficient vector $\boldsymbol{C} = [c_0, c_1, \ldots, c_{M-1}]$ with the scalar $x(n)$ at time $n$. In other words, the input $x(n)$ is multiplied by all the coefficients $c_0, c_1, c_2, \ldots, c_{M-1}$ simultaneously. In the sequel, such products will be referred to as a *vector scaling* operation [1].

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 $[c_0, c_1]\cdot x, c_0 = 00110111, c_1 = 10\,001\,110$, can be decomposed as $c_0 \cdot x = 2^4 \cdot (0011) \cdot x + (0111) \cdot x, c_1 \cdot x = 2^7 \cdot (0001) \cdot x + 2^1 (0111) \cdot x$. If $x, (0011) \cdot x$, and $(0111) \cdot x$ are available, the entire multiplication process is significantly simplified 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 $\boldsymbol{C}$. In the above example, the alphabet set is $\{0001, 0011, 0111\}$. In this example, $(0111) \cdot x$ is computed once and the result is shared to calculate both $c_0 \cdot x$ and $c_1 \cdot x$, which shows the concept of computation sharing in the vector scaling operation.

CSHM architecture is based on the algorithm explained above. Fig. 2 shows the CSHM architecture. CSHM is composed of a precomputer, select units and final adders (S&As). The precomputer performs the multiplication of alphabets with

input $x$. Since alphabets are small bit sequences, the multiplication with input $x$ and alphabets can be done without seriously compromising the performance. Once the multiplications of alphabets with input $x$ are calculated by the precomputer, the outputs are shared by all the S&As, which is the main advantage of CSHM. In order to cover every possible coefficients and perform general multiplication operation, we used eight alphabets $\{1, 3, 5, 7, 9, 11, 13, 15\}$ in the precomputer [1].

S&As perform appropriate select/shift and add operations required to obtain the multiplication output. The select unit is composed of SHIFTER, MUX(8:1), ISHIFTER, and AND gate. To find the correct alphabet, SHIFTERs perform the right shift operation until they encounter 1 and send an appropriate *select* signal to MUXes (8:1). SHIFTERs also send the exact shifted values (*shift* signal) to ISHIFTERs. The MUXes (8:1) select the correct answer among the eight precomputer outputs, $1x, 3x, 5x, 7x, 9x, 11x, 13x, 15x$. ISHIFTERs simply inverse the operation performed by SHIFTERs. When the coefficient input is 0000, we cannot obtain a zero output with shifted value of the precomputer outputs. Simple AND gates are used to deal with the zero (0000) coefficient input. The final adder adds the outputs of the select units to obtain the final multiplication output. An example of the multiplication procedure is shown in Fig. 2.

Fig. 3 shows a parallel $17 \times 17$ CSHM, which is used in our FIR filter implementation. In this structure, the S&As, shown in Fig. 2, are connected in parallel to the precomputer and a final adder is used to generate the final output. This parallel CSHM scheme does not introduce additional select unit delay.

### B. CSHM Implementatios

The $17 \times 17$ CSHM, shown in Fig. 3, is implemented using 0.25-$\mu$m TSMC standard cell library. As shown in Fig. 2, the CSHM is composed of a precomputer and S&As. In our CSHM implementation, the input $\boldsymbol{X}$ is represented in two's complement format and coefficient $\boldsymbol{C}$ is in sign and magnitude format. The output of the CSHM is also in two's complement 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 proposed in Section III-A. Fig. 4 shows the basic structures of $5x$ and $11x$ and Fig. 5 shows the precomputer structure.

*Select Unit:* As shown in Fig. 2, the 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 [2] is used since the signal has to pass through at most one transmission gate in the barrel shifter. The 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. The carry-save array [2] and the new carry-select adder presented in Section III-A are used for high performance as shown in Fig. 6. As mentioned before, the input data is in two's complement format, the coefficient in sign and magnitude, and the final

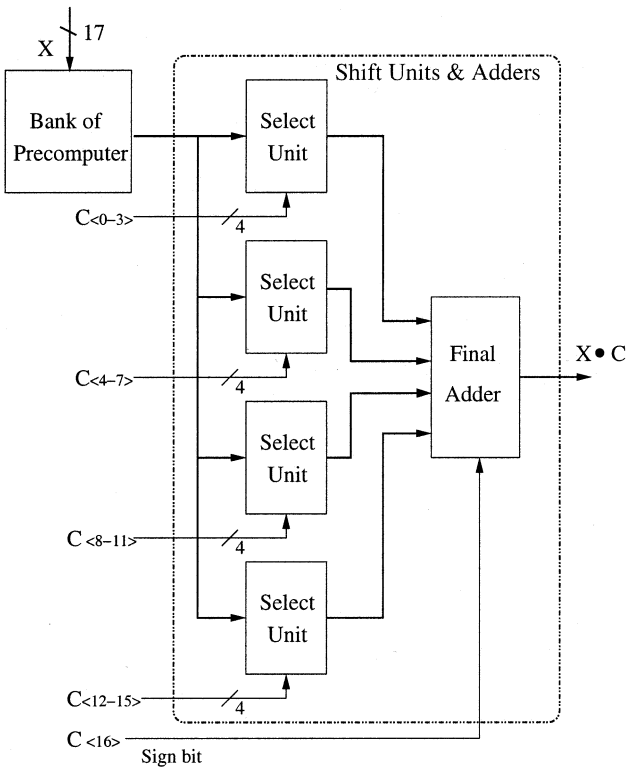


Fig. 3. $17 \times 17$ CSHM structure.

adder output in two's complement. In our CSHM design shown in Fig. 3, the sign bit of coefficient $\boldsymbol{C}$ is not used and $\boldsymbol{C}$ is considered as a positive number in the select unit. The XOR gate array shown in Fig. 6 is efficiently used for controlling the sign of the final adder output. When the coefficient is a positive number (when the sign bit is '0'), since the output of the final adder has the same sign as input data, the inputs of final adder can be added without sign conversion. When the coefficient is a negative number (when the sign bit is '1'), since the output of the final adder has a different sign than the input data, the inputs of the final adder should be converted to numbers with the opposite sign. The architecture is easily realized using the XOR gate array shown in Fig. 6. The addition of the coefficient sign bit and input least significant bit (LSB) can be merged into the carry-select adder.

### C. FIR Filter Based on CSHM

Using the $17 \times 17$ CSHM presented in the previous section, a 10–tap FIR filter with programmable coefficients has been implemented for fabrication. FIR filter can be implemented in *direct form* (DF) [1] or *transposed direct form* (TDF) architecture (Fig. 1). In the DF FIR filter, a large adder in the final stage lies on the critical path and it slows down the FIR filter. For high-performance filter structure, TDF is used in our implementation. In the TDF of the FIR filter shown in Fig. 1, multipliers are replaced by S&As and a precomputer is connected to the input. Therefore, as shown in Fig. 7, the FIR filter using CSHM consists of one precomputer and ten S&As. We can easily see from the figure that the precomputer outputs are shared by all the S&As. In other words, the computations $\alpha_k \cdot x, k = 0, 1, 2, \ldots, 8$, are performed only once for all $k$'s and

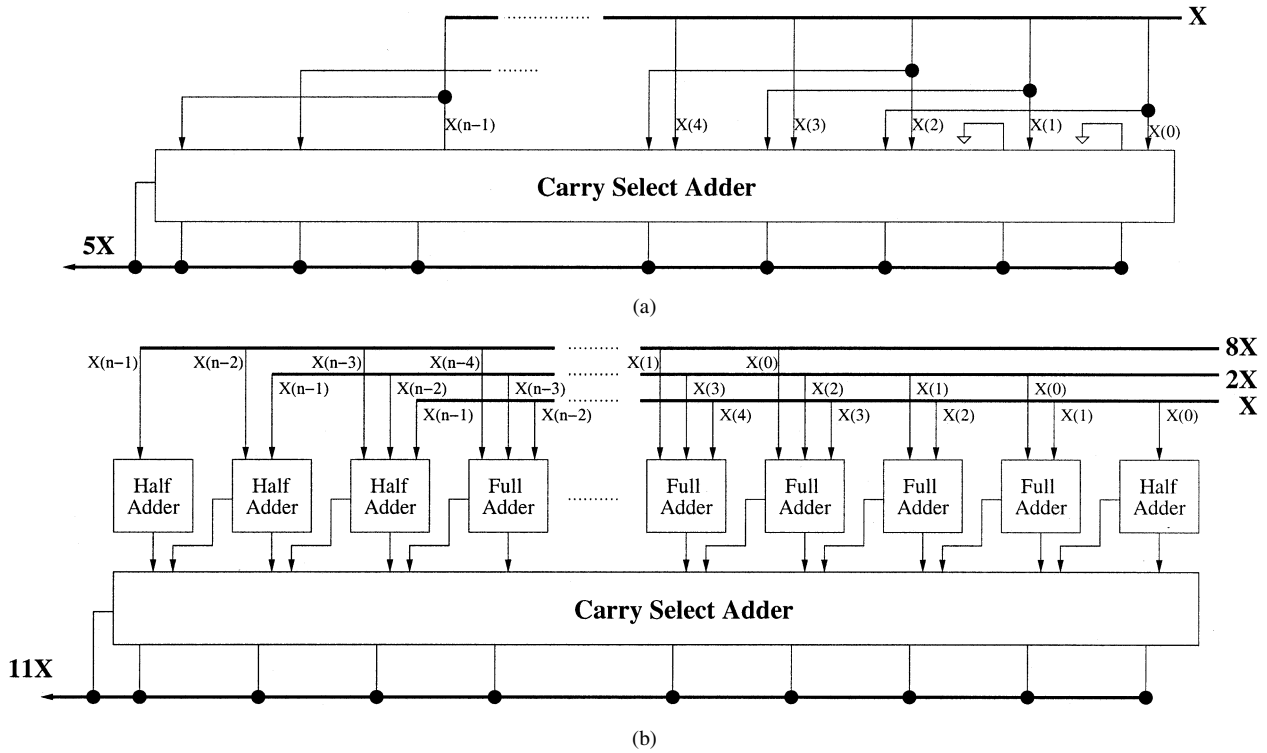


Fig. 4. Precomputer $(5x, 11x)$ architecture. (a) $5x\,(0101x) = 100x\,(<< 2) + 1x$. (b) $11x\,(1011x) = 8x\,(1000x) + 2x\,(10x) + x$.

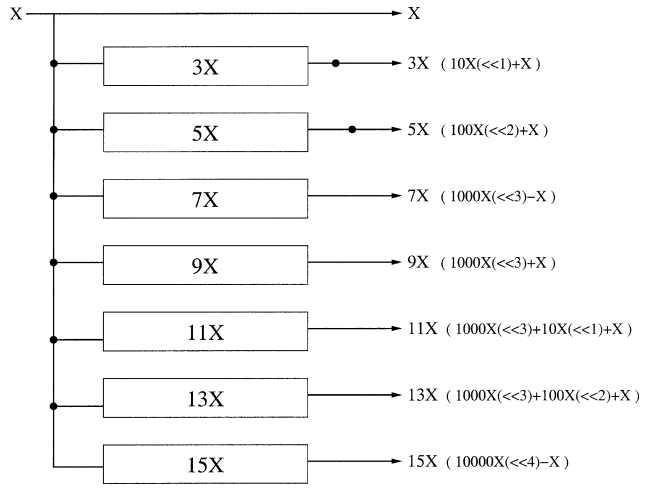


Fig. 5. Precomputer structure.

these values are shared by all the S&As for generating $c_k \cdot x, i = 0, 1, 2, 3, \ldots$. The CSHM scheme efficiently removes the redundant computations in the FIR filtering operation, which leads to low-power and high-performance design.

## III. Circuit Level Techniques

### *A. High-Performance Low-Power Carry-Select Adder Using DTSL*

We have used a new high-performance and low-power carry-select adder using DTSL in the critical functions of the filter. DTSL is based on a skewed logic style [9]. Skewed logic is a noise-tolerant design style and achieves high performance with low power consumption. The circuit topology of skewed logic is the same as that of static CMOS logic, however, the PMOS or the NMOS transistors are preferentially sized to achieve fast high-to-low or low-to-high transitions. For example, to speed up high-to-low transition, the sizes of PMOS transistors are reduced while the NMOS transistors are sized up [9].

DTSL consists of dual data paths using skewed circuits. Fig. 8 shows one example of a DTSL circuit that achieves high performance by duplicating signal paths: one signal path is for fast rising transitions and the other is for fast falling transition. The arrows represent the skew direction. If the input of the first stage of the logic block toggles from high to low, faster data transition takes place through the top data path [Fig. 8(a)]. If the input toggles from low to high, the data transits faster through the bottom path than through top path [Fig. 8(b)]. The combiner detects the earliest transition, latches it, and then transfers the data to the next stage.

The carry-select adder using DTSL, which can be efficiently used in FIR filter 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 carry-select adder using DTSL uses dual paths skewed in opposite directions for carry propagation—one path is used for fast propagation of Carry-in 0, while the other path is used for fast propagation of Carry-in 1. Proper skewing is achieved by preferential sizing of the pull-up and pull-down transistors in static CMOS circuits [10].

Fig. 9 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 some 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 static CMOS gates to control the transmission gates and switching transistors. The logic in the circle

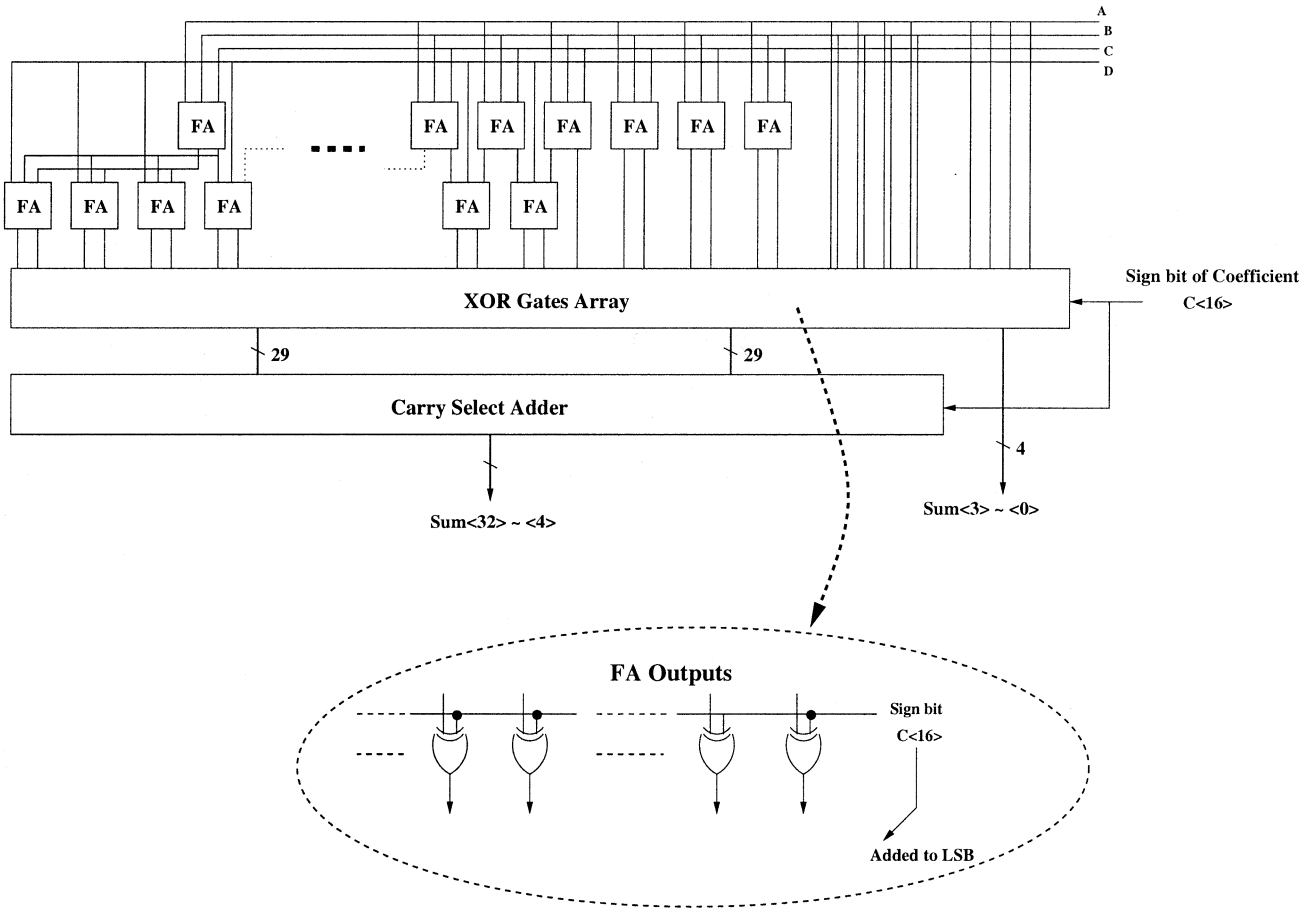


Fig. 6. Final adder architecture.

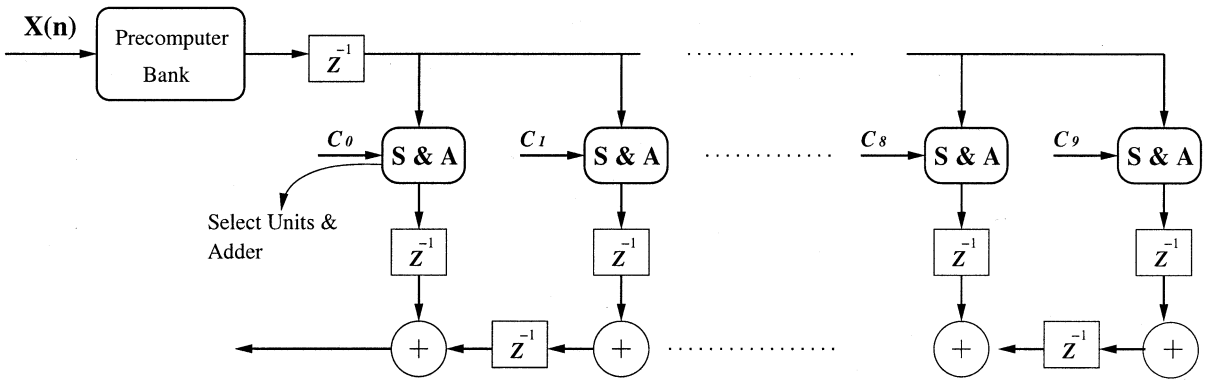


Fig. 7. Architecture of FIR filter based on CSHM.

is for generating SUM. As shown in Fig. 9, the carry propagation logic of each block of the 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 $i = 0, \ldots, n$), transmission gates X, Y will turn on and the switching transistors (MN$i$, MP$i$) 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 conditions. 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 $i$ ($i = 0, \ldots, n$), the Carry-outs on both paths from that stage to the last stage ($i \sim n$) 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 $i$th stage. Hence, in this case, the propagation delay of the carry-select adder is the same as the one of the carry propagation delays from the first stage to the $(i-1)$th stage and from the $i$th stage to the last stage. Then, we have to switch the Carry-in of the next stage (the $[i+1]$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 the AND and OR gates in the

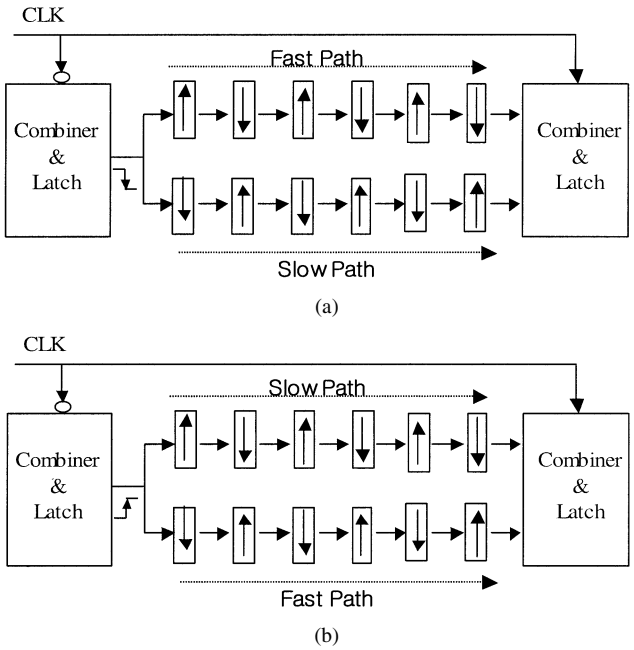


Fig. 8. Block diagram of DTSL when (a) input toggles from high to low, and (b) input toggles from low to high.

control logic of this stage will be 0, and the 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 $i = 0, \ldots, n$) 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.

The proposed carry-select adder using DTSL can achieve high performance and low power. In CSHM, it is used in the precomputer and final adder, which merges four vectors from the select units and is the critical component for performance.

The propagation delay, power consumption, and layout area of the proposed DTSL-based carry-select adder are compared with the carry-select adder using Domino logic and static CMOS logic. Table I shows a comparison of simulation results and layout area of the proposed carry-select adder using DTSL, carry-select adders using Domino, and static CMOS circuits. The propagation delays are obtained under the worst carry propagation case. The average power consumption is obtained with random input vectors with a clock cycle of 10 ns. In Table I, the proposed carry-select adder has 36.7% and 17.7% improvements in power and performance, respectively, compared to the Domino-based carry-select adder. The results also show that the area of the proposed carry-select adder is comparable to that of the static CMOS implementation. The power delay product of the proposed carry-select adder is almost half of that of Domino-based carry-select adder.

TABLE I
COMPARISON RESULTS FOR 31-BIT CARRY-SELECT ADDERS

| Parameter | Domino Logic | Static CMOS | Proposed Adder | Improvement over Domino | Improvement over Static |
|---|---|---|---|---|---|
| Delay [ns] | 1.812 | 2.501 | 1.491 | 17.7 % | 40.4% |
| Power [mW] | 5.092 | 2.667 | 3.224 | 36.7 % | -20.9% |
| PDP [mW x ns] | 9.227 | 6.670 | 4.807 | 47.9 % | 27.9% |
| Layout Area[mm$^2$] | 0.241 | 0.282 | 0.297 | -23.3 % | -5.4% |

### *B. Flip-Flop Design*

Traditionally, the transmission-gate flip-flop (TGFF) has been used in standard cell design [11]. 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 setup time.

Considering the fact that in critical paths the flip-flop delay is the sum of setup time and clock-to-output delay, dynamic latches have lower delay than master–slave latch pairs, which are fully static. Examples of such high-performance flip-flops are the hybrid latch flip-flop (HLFF) [12], semi-dynamic flip-flop (SDFF) [13], and sense-amplifier-based flip-flop (SAFF) [14]. 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 for moderate and low data switching rate, these flip-flops can have unnecessary internal transitions that lead to substantial increase in total power consumption. The conditional capture flip-flop (CCFF) [15] of Fig. 10 eliminates this problem by adding internal clock gating. When the clock input (CLK) is at a low logic value, the node X is precharged to the supply voltage. The NOR gate and transistor M7 provide internal clock gating. If the output Q is at a low logic value and input D is at a high logic value at the rising edge of the clock, the node X is discharged through M5 and the output Q changes to a high logic value. If the output Q is at a high logic value, the output of the NOR gate (NB) remains at the low logic value and M7 remains off, cutting off the discharging path of node X. Therefore, as long as Q remains at a high logic value, there is no redundant transition on node X. When the input D is at a low logic value at the rising edge of the clock, the output Q is changed to a low logic value through the transistor M6. Due the internal clock gating, CCFF achieves statistical power reduction by eliminating redundant transitions of internal nodes while maintaining soft clock edge and negative setup time.

The overall performance and power consumptions of designed TGFF, HLFF, and CCFF for different input patterns were simulated in TSMC 0.25-$\mu$m CMOS technology. The power consumption of the CCFF has a large dependency on input pattern. The CCFF can save 65% power with zero-input 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

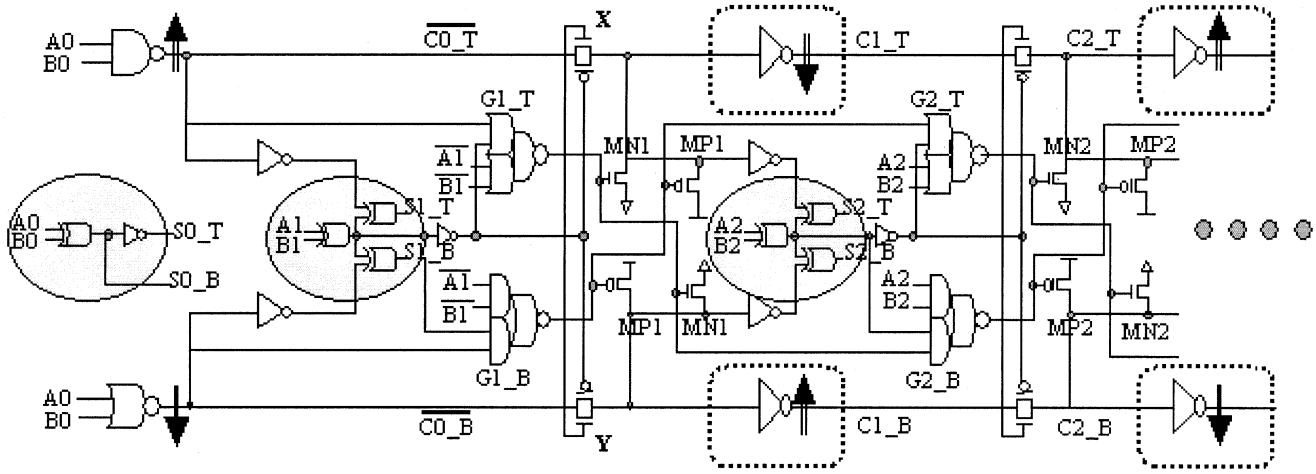


Fig. 9. Block diagram of the carry-select adder used in the FIR filter.

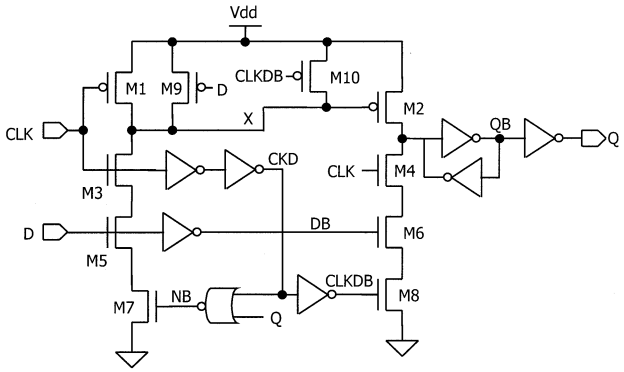


Fig. 10. Conditional capture flip-flop.

activity is the maximum, which is very rare, the overall power consumption is comparable to the HLFF. The TGFF has power comparable to the CCFF, but its minimum data to output latency is far greater due to a positive setup time, leading to poor speed performance. Since flip-flops are in critical paths of our design, we have used the CCFF in our FIR filter implementation to take advantage of its speed and statistical power reduction.

### C. Clock Design

The clock network in FIR filter has been implemented using an H-tree structure. Fig. 11 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 [16]. 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 Fig. 11. The minimum clock cycle can be expressed as

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

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

### D. Physical Design

The floorplan of the filter is shown in Fig. 12. Floorplaning was done to minimize the total interconnect lengths, especially for global signals. The precomputer is placed in a rectangular area on the top of the floorplan so that its outputs can be distributed to all the taps through shortest possible paths. The power supply of the core is separated from the power supply of PADs to be able to separately measure the power of the filter core and the power of the PADs and interfacing to the testing instrument.

## IV. Results and Comparison

The designed 10–tap FIR filter with programmable coefficients was fabricated in the TSMC 0.25-$\mu$m CMOS technology. Fig. 13 shows the die photo of the fabricated chip. The die has an area of 9.91 mm$^2$ and includes more than 130 000 transistors. For the test and measurement of the chip, input test patterns are generated and output patterns and waveforms are monitored using a logic analyzer. Separate power supplies for the core and PADs allow exact measurement of the power consumption of the core of the chip.

Table II shows the characteristics of the CSHM FIR filter chip. The chip could successfully operate at 7-ns cycle time, which was the smallest cycle time provided by our test instruments. However, simulation results show a minimum clock cycle of 5.7 ns. We also implemented the 10-tap FIR filters using a Wallace tree multiplier (WTM) [2] and carry-save array multiplier (CSAM) [2] for comparison. WTM and CSAM are the two most widely used multipliers. Generally, WTM has better performance than CSAM due to the tree-like structure of partial-sum adders. However, WTM had the disadvantage of having very irregular interconnect. A 5:3 compressor [2] is used in the WTM.

Table III shows the minimum clock cycle and power of the FIR filters using different multipliers. Since WTM- and

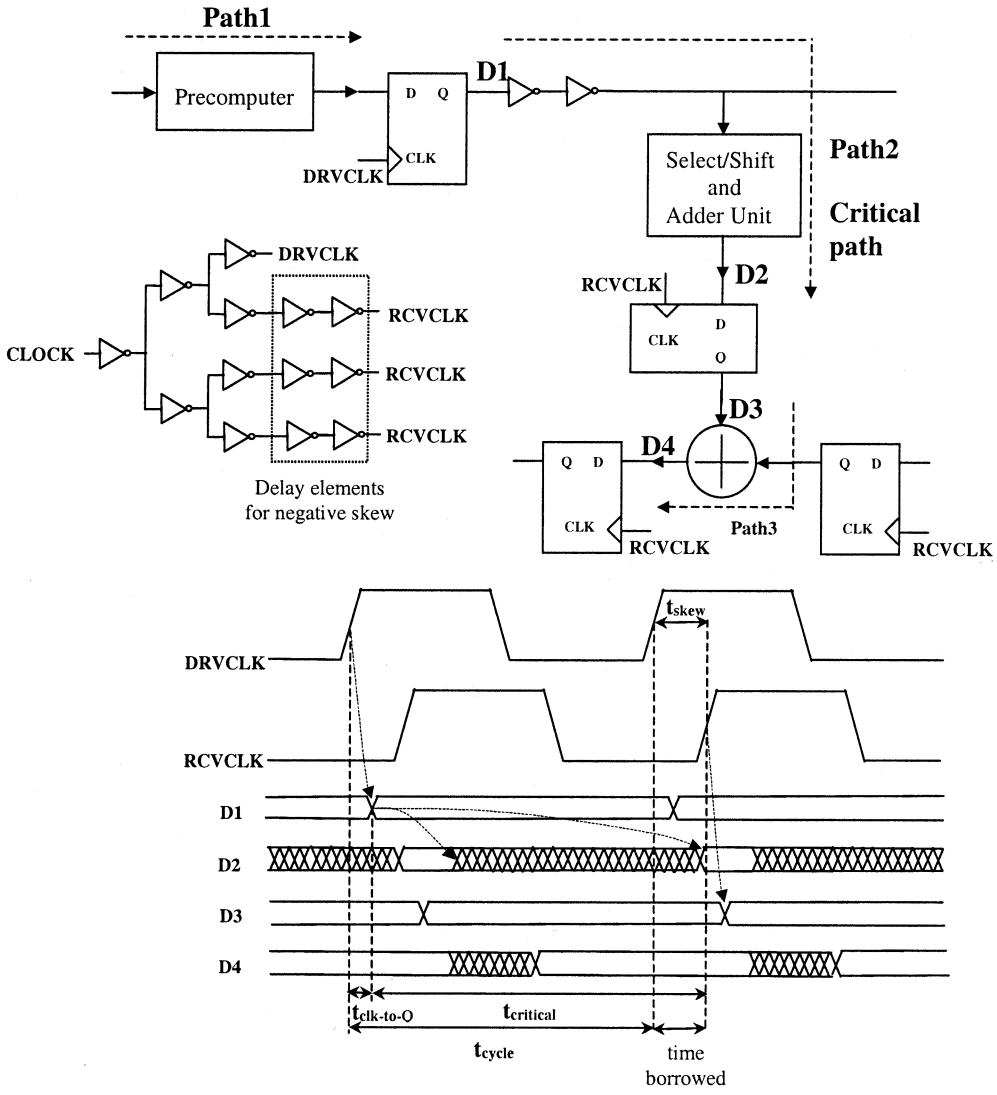


Fig. 11. Clock network and timing of critical path.

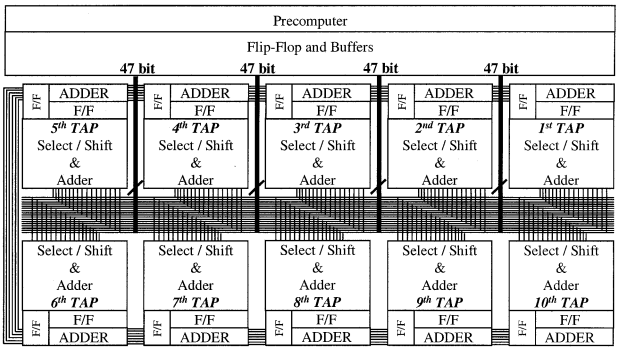


Fig. 12. Floorplan of FIR filter.

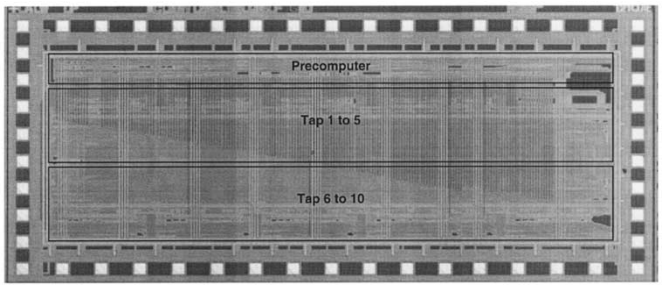


Fig. 13. Die photo of FIR filter.

CSAM-based architectures do not use the concept of computation sharing and perform redundant computations for all filter taps, the FIR filter using CSHM shows better results in terms of performance and power. As shown in the table and based on the simulation results, the FIR filter using CSHM has 19% and 43% performance improvement over FIR filter using WTM and CSAM, respectively. In terms of power consumption, the CSHM scheme has 17% and 20% improvement with respect to the FIR filter based on WTM and CSAM. The power results shown in Table II are measured with a clock cycle of 10 ns. The measured power of the core is smaller than the simulated result.

As shown in Fig. 7, the FIR filter using CSHM has one more pipeline stage than the FIR filter based on WTM and CSAM. The performance of the FIR filter using WTM and CSAM can be improved by adding additional pipeline stage. However, due to the tree structure of WTM and the carry-save array of CSAM,

TABLE II
FEATURES OF CSHM FIR FILTER CHIP

| Process | TSMC 0.25μm |
|---|---|
| Voltage | 2.5V |
| Minimum Cycle Time (ns) | 7.0* |
| Power at 100MHz (mW) | 238.8 |
| die area [mm$^2$] | 9.91 |

* Limited by testing instrument

TABLE III
MEASUREMENT AND SIMULATION RESULTS

| | CSHM Filter (Measurement) | CSHM Filter (Simulation) | FIR using WTM (Simulation) | FIR using CSAM (Simulation) |
|---|---|---|---|---|
| Minimum Cycle Time (ns) | 7* | 5.7 | 7.0 | 10 |
| Power Consumption at 100MHz (mW) | 238.75 | 286.6 | 344.3 | 357.1 |
| Area [μm$^2$]† | $5.0 \times 10^6$ | $5.0 \times 10^6$ | $4.4 \times 10^6$ | $4.1 \times 10^6$ |

* Limited by testing instrument. (Simulation result shows 5.7ns clock cycle.)
† Core area (without I/O pads)

approximately 80 and 50 flip-flops are required to add additional pipeline stage in a single $17 \times 17$ WTM and CSAM, respectively. Moreover, as the number of filter taps increases, the required number of flip-flops for additional pipeline stages also increases linearly, which causes large power consumption in the FIR filter. In the CSHM architecture, since precomputer outputs are shared by all the S&As, we can add additional pipeline stages without incurring large latch overhead. The CSHM architecture has performance and power advantages through the additional pipelining and the sharing of the precomputer outputs by all the S&As, respectively.

## V. CONCLUSION

An FIR filter based on CSHM was implemented using 0.25-$\mu$ m technology. The CSHM algorithm specifically targets reduction of redundant computation in FIR filtering operation. Using the CSHM scheme, the multiplications in vector scaling operation were significantly simplified to add and shift operations of alphabets multiplied by input $x$. These common computations were 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 the adder and flip-flop were proposed and used in the full-custom FIR filter implementation. The CSHM scheme and circuit-level techniques helped to achieve low-power and high-performance FIR filtering operation. The proposed CSHM architecture is also applicable to adaptive filter and matrix multiplication implementation. The ideas presented in this paper will help the design of DSP algorithms and their implementation for high-performance and low-power applications.

## REFERENCES


[1] J. Park, H. Choo, K. Muhammad, and K. Roy, "Non-adaptive and adaptive filter implementation based on sharing multiplication," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, June 2000, pp. 460–463.

[2] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[3] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filter with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.

[4] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Trans. Acoust., Speech Signal Processing*, vol. ASSP-31, pp. 583–591, June 1983.

[5] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 677–688, Oct. 1996.

[6] M. Potkonjak, M. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 151–165, Feb. 1996.

[7] N. Sankarayya, K. Roy, and D. Bhattacharya, "Algorithms for low power high speed FIR filter realization using differential coefficients," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 488–497, June 1997.

[8] K. Muhammad and K. Roy, "A graph theoretic approach for synthesizing very low-complexity high-speed digital filters," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 204–216, Feb. 2002.

[9] A. Solomatmikov, D. Somasekhar, N. Sirisantana, and K. Roy, "Skewed CMOS: Noise-tolerant high-performance low-power static circuit family," *IEEE Trans. VLSI Syst.*, vol. 10, pp. 469–476, Aug. 2002.

[10] W. Jeong, K. Roy, and C. Koh, "High-performance low-power carry-select adder using dual transition skewed logic," in *Proc. ESSCIRC*, 2001, pp. 172–175.

[11] 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.

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

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

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

[15] B. S. Kong *et al.*, "Conditional capture flip-flop for statistical power reduction," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2000, pp. 290–291.

[16] H. Partovi, A. Chandrakasan, W. J. Bowhill, and F. Fox, "Clocked storage elements," in *Design of High-Performance Microprocessor Circuits*. Piscataway, NJ: IEEE Press, 2000, pp. 207–234.


**Jongsun Park** received the B.S. degree in electronics engineering from Korea University, Seoul, Korea, in 1998 and the M.S. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2000. He is currently working toward the Ph.D. degree in electrical and computer engineering at Purdue University.

His research interests focus on the low-power high-performance VLSI architectures and circuit design for digital signal processing and digital communications.

**Woopyo Jeong** received the B.S. and M.S. degrees in electrical engineering from Yonsei University, Seoul, Korea, in 1991 and 1993, respectively. He is currently working toward Ph.D. degree in electrical and computer engineering at Purdue University, West Lafayette, IN.

In 1993, he joined Samsung Electronics Company, Ltd., Korea, where he was engaged in research and development for EDO, synchronous DRAM, and Rambus DRAM. His research interests include high-performance and low-power circuit design.

**Hamid Mahmoodi-Meimend** (S'00) received the B.S. degree in electrical engineering from the Iran University of Science and Technology, Tehran, in 1998 and the M.S. degree in electrical engineering from the University of Tehran in 2000. He is currently working toward the Ph.D. degree in electrical engineering at Purdue University, West Lafayette, IN.

His research interests include low-power and high-performance circuit design for deep- submicrometer CMOS technologies.

**Yongtao Wang** received the B.Sc. and M.Eng. degrees from Fudan University, Shanghai, P.R. China, in 1996 and 1999, respectively. He is currently working toward the Ph.D. degree in electrical and computer engineering at Purdue University, West Lafayette, IN.

His main research interest includes VLSI hardware architecture for digital signal processing and communication systems with particular emphasis in low-complexity and low-power design, and energy-efficient multimedia transmission within wireless OFDM systems.

**Hunsoo Choo** received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea, in 1998 and the M.S. degree from Purdue University, West Lafayette, IN, in 2000, both in electrical and computer engineering. He is currently working toward the Ph.D. degree in the electrical and computer engineering at Purdue University.

His main research interest is high-level synthesis techniques for low-complexity and low-power design, and low-power VLSI design of multimedia wireless communications and signal processing systems.

**Kaushik Roy** (SM'95–F'01) received the B.Tech. degree in electronics and electrical communications engineering from the Indian Institute of Technology, Kharagpur, India, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1990.

He was with the Semiconductor Process and Design Center, Texas Instruments Incorporated, Dallas, TX, where he worked on FPGA architecture development and low-power circuit design. He joined the electrical and computer engineering faculty at Purdue University, West Lafayette, IN, in 1993, where he is currently a Professor. He is on the Technical Advisory Board of Zenasis Inc. and a Research Visionary Board Member of Motorola Laboratories. He has published more than 250 papers in refereed journals and conferences, holds six patents, and is a coauthor of a book on low-power CMOS VLSI design. His research interests include VLSI design/CAD with particular emphasis in low-power electronics for portable computing and wireless communications, VLSI testing and verification, and reconfigurable computing.

Dr. Roy received the National Science Foundation Career Development Award in 1995, the IBM Faculty Partnership Award, the AT&T/Lucent Foundation Award, Best Paper Awards at the 1997 International Test Conference, IEEE 2000 International Symposium on Quality of IC Design, and IEEE Latin American Test Workshop, and is currently a Purdue University Faculty Scholar Professor. He has been on the editorial board of IEEE *Design and Test*, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, and IEEE TRANSACTIONS ON VLSI SYSTEMS. He was Guest Editor for the Special Issue on Low-Power VLSI in the IEEE *Design and Test* (1994), IEEE TRANSACTIONS ON VLSI SYSTEMS (June 2000), and IEE *Proceedings—Computers and Digital Techniques* (July 2002).