# Dynamic Voltage Scaling Multiplier Using Twin-Precision Technique for Low Power Design

**Mythili.C[1] Chandrakala.S[2]**
[1]M.E. (VLSI-Design) [2]Assistant Professor
[1,2]Kalaignar Karunanidhi Institute of Technology

*Abstract*— The flexible twin-precision multiplier is combined with variable precision, parallel processing (PP), Razor based dynamic voltage scaling (DVS), and dedicated MP operand scheduling to provide optimum performance for different operating conditions. The twin-precision technique can reduce the power dissipation by adapting a multiplier to the bit width of the operands being computed. The technique also enables an increased computational throughput, by allowing several narrow-width operations to be computed in parallel. All of the building blocks of proposed flexible twin-precision multiplier can either work as independent small precision multiplier or parallel to perform higher-precision multiplier. While still maintain the full through-put, the dynamic voltage and frequency scaling management unit configures the twin-precision multiplier to operate at the proper precision and frequency. Adapting to the run-time workload for targeted application, that flexible multiplier can used to design for DSP application. Razor flip-flops together with a dithering voltage unit then configure the multiplier to achieve the lowest power consumption. The single-switch dithering voltage unit and razor flip-flops help to reduce the voltage margin and overhead typically associated to DVS to lowest level. Finally, the proposed high speed flexible twin-precision multiplier, can further benefits from an operand scheduler that rearranges the input data, hence determine the optimum voltage and frequency operating conditions for minimum power and delay consumption. This architecture is simulated and results are obtained using TANNER EDA 13, MODEL SIM 10.1b and XILINX ISE tool.

***Key words:*** twin precision multiplier, dynamic voltage scaling (DVS), parallel processing (PP), razor flip-flops

## I. INTRODUCTION

With the widespread use of portable computing and communication systems, power consumption has become an important issue in very large scale integration (VLSI) design. Furthermore, multipliers are fundamental building blocks and the bottleneck in terms of performance and power consumption in many multimedia and digital signal processing (DSP) applications. Therefore, it is crucial to develop a multiplier with high performance but low power consumption. Multiplier is typically designed for a fixed maximum word-length to suit the worst case scenario. However, the real effective word-lengths of an application vary dramatically. The use of a non-proper word-length may cause performance degradation or inefficient usage of the hardware resources.

In addition, the minimization of the multiplier power budget requires the estimation of the optimal operating point including clock frequencies, supply voltage, and threshold voltage [1].In most VLSI system designs, the supply voltage is also selected based on the worst case scenario. In order to achieve an optimal power/performance ratio, a variable precision data path solution is needed to cater for various types of applications. Dynamic Voltage Scaling (DVS) can be used to match the circuit's real working load and further reduce the power consumption. Given their fairly complex structure and interconnections, multiplier can exhibits a large number of unbalanced paths, resulting in substantial glitch generation and propagation [8].This spurious switching activity can be mitigated by balancing internal paths through a combination of architectural and transistor-level optimization techniques. In addition to equalizing internal path delays, dynamic power reduction can also be achieved by monitoring the effective dynamic range of input operands so as to disable unused section of multiplier.

Therefore, an 8-bit multiplication computed on a 32-bit Booth multiplier would result in unnecessary switching activity and power loss. Many analyzed this word-length optimization. They proposed an ensemble of multipliers of different precisions, with each optimized for a particular scenario. Each pair of incoming operands is routed to the smallest multiplier that can give the result to take advantage of the lower energy consumption of the smaller circuit. This ensemble the systems is reported to consume the low power but this came at the high cost, chip area given the used ensemble structure. To address this issue, [3], [5] proposed to share and reuse some functional modules within the ensemble.

In [3], an 8-bit multiplier is reused for the 16-bit multiplication, adding scalability without large area penalty. Reference [5] extended this method by implementing pipelining to further improve the multiplier's performance. Combining multi-precision (MP) [1] with dynamic voltage scaling (DVS) can provide a reduction in power consumption. In proposed technique twin-precision is used rather than multi-precision array multiplier. Power and delay is reduced dramatically than multi-precision multiplier.

## II. EXISTING METHODS

The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm is one of the most popular algorithms. To achieve speed improvements Wallace Tree algorithm can be used to reduce the number of sequential adding stages. Further by combining both Modified Booth algorithm and Wallace Tree technique we can see advantage of both algorithms in one multiplier. However with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from

complex routing. On the other hand "serial parallel" multipliers compromise speed to achieve better performance for area and power consumption. The selection of a multiplier actually depends on the nature of application.

### *A. Summary:*

In this section performance measures of multipliers discussed so far are summarized and compared. These results were obtained after synthesizing individual architectures targeting Xilinx FPGA 4052XL-1HQ240C. All comparisons are based on the synthesis reports keeping one common base for comparison. We summarize Area (Total number of CLBs required), Delay and Power Consumption and also calculate Delay. Power (DP), Area Power (AP), Area Speed (AT) and Area·Speed2 (AT2) product

| | Array Multiplier | Modified Booth Multiplier | Wallace Tree Multiplier | Modified Booth -Wallace Tree Multiplier | Twin Pipe Serial-Parallel Multiplier |
|---|---|---|---|---|---|
| Area – Total CLB's (#) | 1165 | 1292 | 1659 | 1239 | 133 |
| Maximum Delay D (ns) | 187.87 | 139.41 | 101.14 | 101.43 | 22.58 (722.56)* |
| Power(mW) (at highest speed) | 16.6506 (at188 ns) | 23.136 (at 140ns) | 30.95 (101.14ns) | 30.862 (at 101.43ns) | 2.089 (at 722.56ns) |
| Power P (mW) when delay = 722.56ns | 4.329 | 4.638 | 4.332 | 4.332 | 2.089 |
| Delay ·Power Product (DP) (ns mW) | 813.28 | 622.30 | 438.138 | 439.39 | 1509.42 |
| Area·Power Product (AP) (# mW) | 5043.28 | 5767.23 | 7186.788 | 5367.35 | 277.837 |
| Area·Delay Product (AD) (# ns) | 218.868 x $10^3$ | 180.118 x $10^3$ | 167.791 x $10^3$ | 125.671 x $10^3$ | 96.101 x $10^3$ * |
| Area·Delay$^2$ Product(AD$^2$) (# ns$^2$) | 41.119 x $10^6$ | 25.110 x $10^6$ | 16.970 x $10^6$ | 12.747 x $10^6$ | 69.438 x $10^6$ * |

Table 1: Comparision of various multiplier

Recently they proposed array multiplier which consumes less power and delay than all other existing multipliers. Hence in existing they used array multiplier. Let discuss briefly about existing and proposed its comparison.

### *B. Array Multiplier:*

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length.

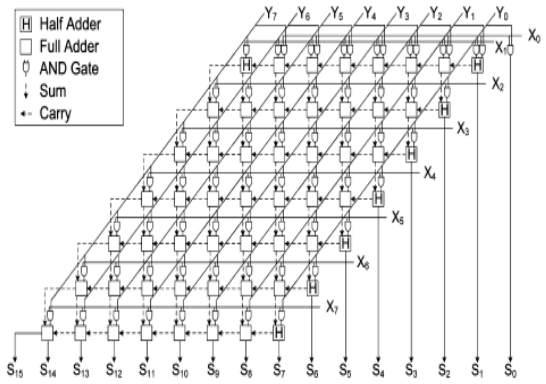



Fig. 1: Block diagram for 8bit unsigned multiplier

#### *1) Advantage:*

- An array multiplier –a multiplication method in which an array of identical cells generates new partial product and accumulation of it at the same time.
  - We can use pipelines at each level.
- Result from the adder can be latched at each level and used as input for next level adder circuit.
- The delay is logarithmically proportional to bit size of multiplicand and multiplier if we use the high speed array multiplier circuit.

#### *2) Disadvantage*

- Large number of logic gates required to design an array multiplier.
- Switching activity is high.

## III. PROPOSED METHOD

The twin-precision technique, that offers the power reduction as operand guarding and the possibility of performing double-throughput multiplications. The twin-precision technique is an efficient way of achieving double throughput in a multiplier with low area overhead and with only a small delay penalty.

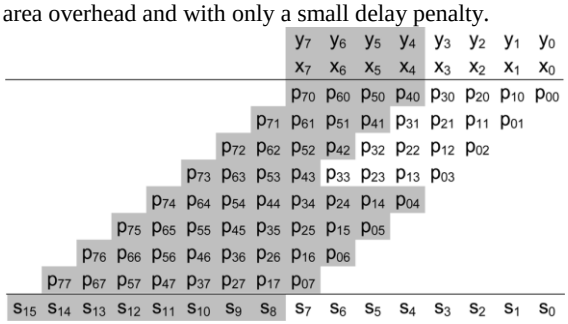


Fig. 2: Illustration of an unsigned 8-bit multiplication,where precision of the operands is smaller than the precision of the multiplication.Unused bits of operands and product,as well as unused partial products are shown in grey

### *A. Twin-Precision Fundamentals:*

Initially we present the twin-precision technique using an illustration of unsigned binary multiplication. In an unsigned binary multiplication each bit of one of the operands, called the multiplier, is multiplied with the second operand, called multiplicand Pij=YiXj. That way one row of partial products is generated. Each row of partial products is shifted according to the position of the bit of the multiplier, forming what is commonly called the partial-product array. Finally, partial products that are in the same column are summed together, forming the final result. An illustration of an 8-bit multiplication is shown in Fig. 2.

Let us look at what happens when the precision of the operands is smaller than the multiplier we intend to use. In this case, the most significant bits of the operands will only contain zeros, thus large parts of the partial-product array will consist of zeros. Further, the summation of the most significant part of the partial-product array and the most significant bits of the final result will only consist of zeros.

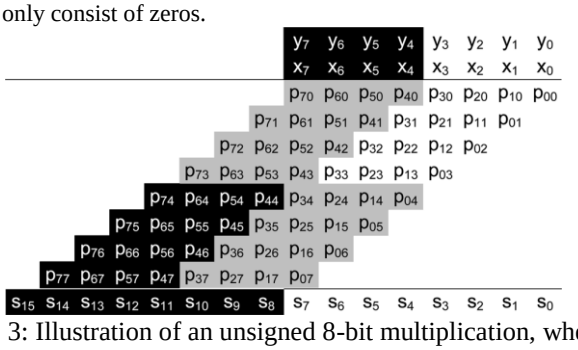


Fig. 3: Illustration of an unsigned 8-bit multiplication, where a 4-bit multiplication, shown in white is computed in parallel with second 4-bit multiplication shown in black

An illustration of an 8-bit multiplication, where the precision of the operands is four bits, is shown in Fig. 2. It shows that large parts of the partial-product array only consist of zeros and are, thus, not contributing any useful information to the final result. What if these partial products could be utilized for a second, concurrent multiplication .Since partial products of the same

column are summed together, it would not be wise to use any of the partial products that are in the same column as the multiplication that is already computed. Looking closer at the 4-bit multiplication marked in white in Fig. 2, one can also observe that the column at position should not be used either. This is because that column might have a carry from the active part of the partial-product array that will constitute the final. Altogether this makes only the partial products in the most significant part of the partial-product array available for a second multiplication.

In order to be able to use the partial products in the most significant part, there has to be away of setting their values. For this we can use the most significant bits of the operands, since these are not carrying any useful information. If we are only looking at the upper half of the operands, the partial products generated from these bits are the ones shown in black in Fig. 3

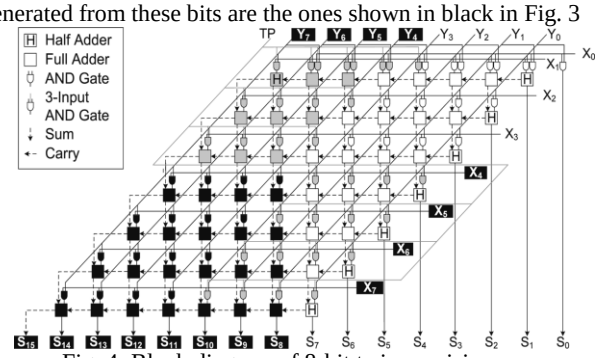
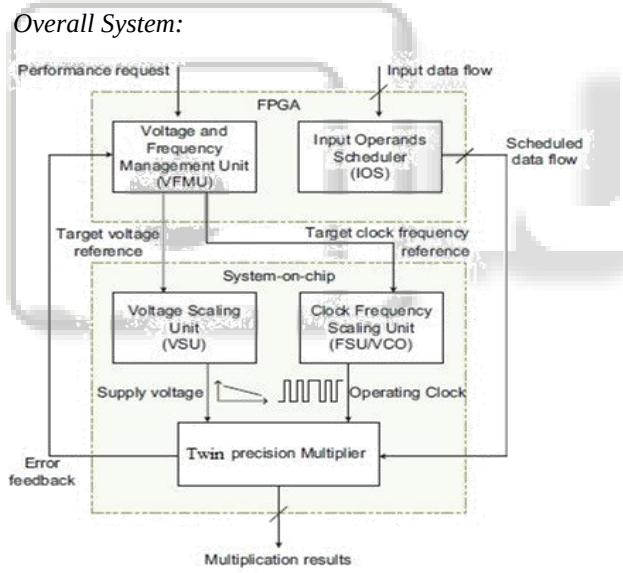



Fig. 4: Block diagram of 8-bit twin precision array Multiplier

*B. Overall System:*



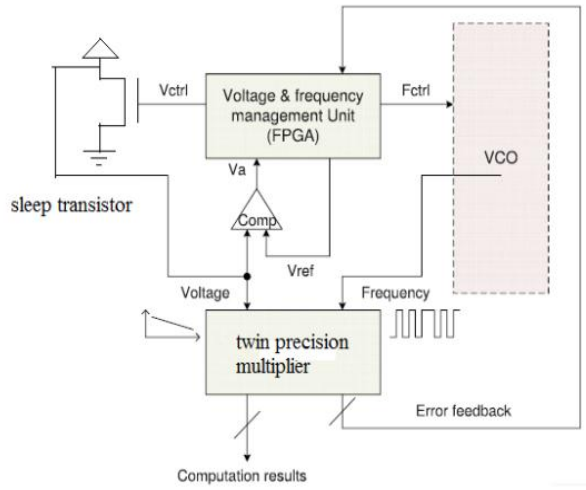




Fig. 5: Multiplier Unit

*1) Parallel Processing:*

Parallel Processing is the ability to carry out multiple operations. This multiplier comprises 8 × 8 bit reconfigurable multiplier.These building blocks can either work as nine independent multipliers work in parallel to perform one, two or three 16 × 16 bit multiplications or a single 32 × 32bit operation parallel processing can be used to increase the throughput or reduce the supply voltage level for low power operation.

*2) Voltage Scheduler:*

Voltage Scheduler is a new operating system component for use in a DVS system. It controls the processor speed by writing the desired clock frequency to a system control register. The register's value is used by the regulation loop to adjust the CPU clock frequency and regulated voltage.

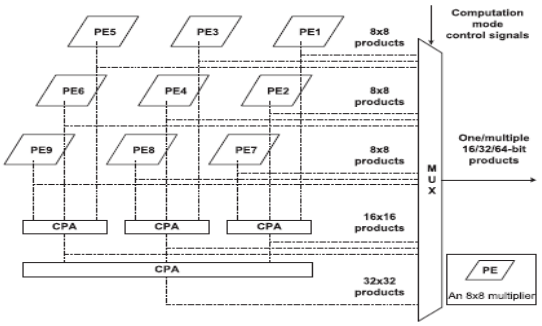



Fig. 6: Parallel Processing

*3) Dynamic Voltage and Frequency Unit:*

Dynamic Voltage and Frequency Management (DVFM) approach offers both Dynamic Frequency Control (DFC) and Dynamic Voltage Control (DVC). It controls the clock frequency and also adaptively controls the supply voltage and achieving power reduction depends on the management scheduler. General methods of power reduction are voltage scaling and lowering the operating clock frequency.

*4) Voltage Dithering Technique:*

Voltage Dithering Technique Voltage dithering was proposed as a low overhead implementation of DVS to provide near-optimum power savings using only a few discrete voltage and frequency. This corresponds to a fixed-throughput system.

*5) Input Operands Scheduler:*

Operand scheduler that rearranges operations on input operands so as to reduce the number of transitions of the supply voltage and, in turn, minimize the overall power consumption of the multiplier the input operands scheduler (IOS) whose function is to reorder the input data stream into a buffer, hence to reduce the required power supply voltage transitions.

*6) Algorithms:*

In the following, term discuss three different algorithms to reduce this overall power consumption in which the three types of algorithm are described below so Each of these algorithms constitutes a different approach to the mixed-precision data held in the operands buffer that operand buffer are present in input operand scheduler.

*7) Algorithm A:*

The algorithm-A states that the multiplier can perform different (8, 16, and 32) bit precision operation so at the time level of supply voltage and frequency are varying depend on different bit operation are perform depend on the input variation (clock, voltage and frequency), in algorithm-A states that the input voltage is varying depend on the bit

operation as well as operating frequency also varying depend on the bit operation.

*8) Algorithm B:*

This algorithm removes all transitions of the power supply voltage by making Vmin32, Vmin16, and Vmin8 equal and adjusting f32, f16, and f8 such that the overall throughput is kept unchanged.

*9) Algorithm C:*

Although Algorithm B removes power supply voltage transitions by setting a single-voltage level V, there may be better power saving combinations of power supply voltages and operating frequencies: (Vmin32, f32), (Vmin16, f16), and (Vmin8, f8). The aim of algorithm C is to find such an optimum for reduced power consumption. To limit complexity, we will only seek to minimize the dynamic power dissipated as a result of the computation.

## IV. Simulation Results and Discussion

### *A. Simulation Tools:*

*1) Modelsim and Xilinx:*

ModelSim PE, our entry-level simulator, offers VHDL, Verilog, or mixed-language simulation. Coupled with the most popular HDL debugging capabilities in the industry, ModelSim PE is known for delivering high performance, ease of use, and outstanding product support. Xilinx is a software tool produced by Xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmed.

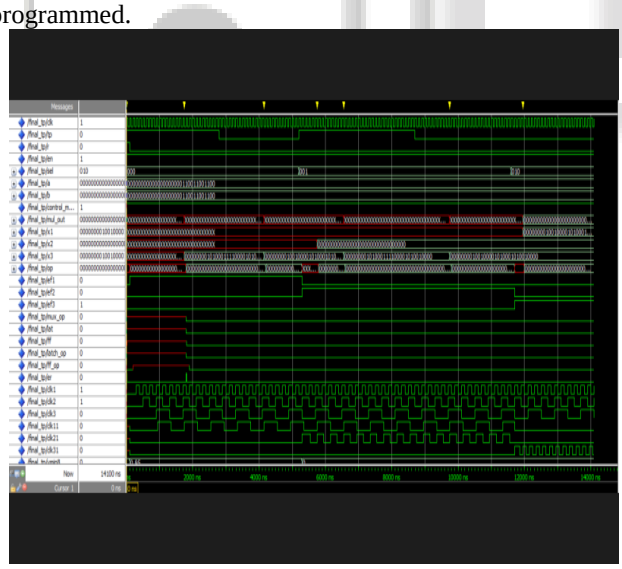


Fig. 7: Twin-Precision Multiplier for 32bit

| FACTORS | EXISTING METHOD | PROPOSED METHOD |
|---|---|---|
| POWER | 370mW | 348mW |
| DELAY | 70.86ns | 66.82ns |

Table 2: Comparison using Modelsim & Xilinx

### *B. Tanner Implementation:*

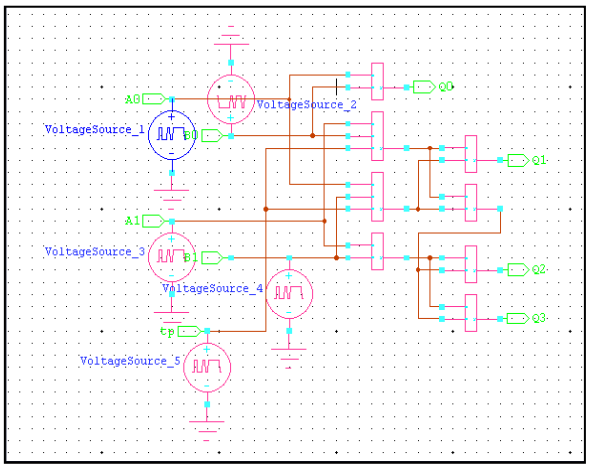



Fig. 8: Twin-Precision For Two Bit

| FACTORS | EXISTING METHOD | PROPOSED METHOD |
|---|---|---|
| POWER | $2.1\times10^{-3}$mW | $1.1\times10^{-6}$Mw |
| DELAY | $4.118\times10^{-3}$ns | $4.1\times10^{-3}$ns |

Table 3: Comparison using Tanner method

## V. Conclusion

The proposed twin-precision razor-based DVS multiplier provided a solution towards achieving full computational flexibility and low power consumption for various general purpose low-power applications. The presented twin-precision technique allows for flexible architectural solutions, where the variation in operand bit-width that is common in most applications can be harnessed to decrease power dissipation and to increase throughput of multiplications.

## VI. Future Work

This paper can be modified by proposing an architecture which includes variable latency adder. The CPA is replaced by VL adder which greatly reduced the complexity of the circuit achieves low power. It can be implemented in decimator filter application.

## References

[1] Amine Bermak, Farid Boussaid and Xiaoxiao Zhang , IEEE,"32 Bit×32 Bit Multiprecision Razor-Based Dynamic Voltage Scaling Multiplier With Operands Scheduler" IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 22, No. 4, April 2014 759.

[2] D. T. Blaauw, D. M. Bull ,S. Das, S. Kalaiselvan, K. Lai, S. Pant, and C. Tokunaga, "RazorII: In situ error detection and correction for PVT and SER tolerance," IEEE J. Solid-State Circuits, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[3] M. Anis, M. Elmasry, and A. Youssef, "A comparative study between static and dynamic sleep signal generation techniques for leakage tolerant designs," IEEE Trans. Very Large Scale Integration multiplier," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57,no. 3, pp. 568–580, Mar. 2010.

[4] F. Buergin, F. Carbognani, N. Felber, W. Fichtner, H. Kaeslin,,"Transmission gates combined with level-restoring CMOS gates reduce glitches in low-power low-frequency multipliers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 7, pp. 830–836, Jul. 2008.
[5] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modelling," Computer, vol. 35, no. 2, p. 59–67, Feb. 2002.
[6] Embedded Microprocessor Benchmark Consortium. [Online]. vailable: http://www.eembc.org
[7] S. K. Raman, V. Pentkovski, and J. Keshava, "Implementing streaming SIMD extensions on the Pentium III processor," IEEE Micro, vol. 20, no. 4, pp. 47–57, Jul./Aug. 2000.
[8] K. Diefendorff, P. K. Dubey, R. Hochsprung, and H. Scale, "AltiVec extension to PowerPC accelerates media processing," IEEE Micro, vol. 20, no. 2, pp. 85–95, Mar./Apr. 2000.
[9] M. Tremblay, M. O'Connor, V. Narayanan, and L. He, "VIS speeds new media processing," IEEE Micro, vol. 16, no. 4, pp. 10–20, Aug.1996.
[10] R. B. Lee, "Accelerating multimedia with enhanced microprocessors, "IEEE Micro, vol. 15, no. 2, pp. 22–32, Apr. 1995.
[11] R. B. Lee, "Subword parallelism with MAX-2," IEEE Micro, vol. 16,no. 4, pp. 51–59, Aug. 1996.
[12] J. Goodacre and A. N. Sloss, "Parallelism and the ARM instruction set architecture," IEEE Comput., vol. 38, no. 7, pp. 42–50, Jul. 2005.
[13] "ARM Architecture Reference Manual," 1st ed. ARM Limited, Cambridge, U.K., Jul. 2005.
[14] J. Sklansky, "Conditional-sum addition logic," IRE Trans. Electron. Comput., pp. 226–231, Jun. 1960.
[15] M. Själander, P. Larsson-Edefors, and M. Björk, "A flexible datapath interconnect for embedded applications," in IEEE Comput. Soc. Annu. Symp. VLSI, May 2007, pp. 15–20.
[16] M. Thuresson, M. Själander, M. Björk, L. Svensson, P. Larsson-Edefors,and P. Stenström, "FlexCore: Utilizing exposed datapath control for efficient computing," in IEEE Int. Conf. Embedded Computer Systems: Architectures, Modeling and Simulation, Jul. 2007, pp. 18–25.