Improving Power-awareness of Pipelined Array Multipliers using 2-Dimensional Pipeline Gating and its Application on FIR Design

Jia Di, J. S. Yuan and R. DeMara
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, Florida 32816, U.S.A.
jdi@pegasus.cc.ucf.edu, yuanj@mail.ucf.edu, demara@mail.ucf.edu

Abstract: Power-awareness indicates the scalability of the system energy with changing conditions and quality requirements. Although Boolean multipliers have natural power awareness to the changing of input precision, deeply pipelined designs do not have this benefit. A 2-dimensional pipeline gating scheme is proposed in this paper to improve the power awareness in these designs. This technique is to gate the clock to registers in both vertical direction (data flow direction in pipeline) and horizontal direction (within each pipeline stage). For signed multipliers using 2's complement representation, sign extension, which wastes power and causes longer delay, could be avoided by implementing this technique. Very little additional area is needed so that the overhead is hardly noticeable. Simulation results show that an average power saving of 65-66% and latency reduction of 44-47% can be achieved for multipliers under equal input precision probabilities. An application of power-aware multipliers on FIR design is also included.

Index terms – power-awareness, 2-dimensional pipeline gating, array multiplier

1. Introduction

Due to the trend of portable communication and computing devices and the dramatic decrease of feature size, low power technique has long been a major interest of IC designers. Many low power techniques have been developed to match different circuits and conditions [1]. Bhardwaj et al., [2] introduced a new measurement, power-awareness, to indicate the ability of the system power to scale with changing conditions and quality requirements. Scalability is an important figure-of-merit since it allows the end user to implement operational policy [2], just like the user of mobile multimedia equipment needs to select between better quality and longer battery operation time. The examples include that a well-designed system must gracefully degrade its quality and performance as the available energy resources are depleted [3]. In such systems like digital camera,

users are allowed to select certain parameters like resolution. After user selects a resolution, there will be a short period of time to allow the system to set up. During this period, the CPU will configure itself and set up the control to the whole system. Such parameters will not change frequently. After each change, the new value will remain stable for sometime. So for a power aware system in these applications, on-the-fly control is not needed.

The power dissipation in CMOS circuit has three components: switching power, short-circuit power, and leakage power. Among these components, switching power is the dominant figure. When a node in circuit is switching, the load capacitance on this node will dissipate power due to the charging/discharging operation. If the switching activity could be reduced, the total power dissipation will be saved. For Boolean non-pipelined multipliers, starting from reset-to-zero state, low input precision calculation (like 0001×0001) dissipates much less power than high input precision calculation (like 1111×1111) because there are much less switching activities in internal nodes. Here the input precision is defined as the number of useful input bits (without padded 0's in high order bits) during the calculation. For example, the input precision of 0101 is 3, while the input precision of 1000 is 4. So Boolean non-pipelined multipliers are said to have natural power awareness to the changing of input precisions.

Deeply pipelined multipliers are used in such systems that need either high throughput or accurate timing control, like retimed FIR filters. In pipelined multipliers, each pipeline stage contains a number of registers. Clock is connected to each register. In each clock cycle, a transition will occur on the clock input node of each register. This transition is independent of input data and will cause power dissipation even when the

current input data of the register is the same as the current data output. Since in deeply pipelined designs, the number of registers is much larger than that of other elements, these designs do not have the natural power awareness to the changing of input precision due to the large portion of power dissipated on clock input nodes. The power dissipation in deeply pipelined multipliers is nearly stable under different input precisions. Figure 1 shows the average power dissipation under different input precisions of a deeply pipelined 16-bit unsigned array multiplier.

For signed multipliers using 2's complement number representation, this problem is even worse. The Baugh-Wooley algorithm for signed multiplication is used as an example in this paper. The equation of Baugh-Wooley algorithm for an $n \times n$ multiplication is shown in (1).

$$A \times B = -2^{2n-1} + \left(\overline{a_{n-1}} + \overline{b_{n-1}} + a_{n-1}b_{n-1}\right) \cdot 2^{2n-2} + \sum_{0}^{n-2}\sum_{0}^{n-2} a_i b_j 2^{i+j} + \left(a_{n-1} + b_{n-1}\right) \cdot 2^{n-1} + \sum_{0}^{n-2} b_{n-1} \cdot \overline{a_i} \cdot 2^{i+n-1} + \sum_{0}^{n-2} a_{n-1} \cdot \overline{b_i} \cdot 2^{i+n-1} \quad (1)$$

The tablet form of a 4×4 multiplication process using modified Baugh-Wooley algorithm is shown in Fig. 2. *X* and *Y* are 4-bit operands with the first bit as sign bit, and *S* is the 7-bit output. There are two major differences between Fig. 2 and 4×4 unsigned multiplication process shown in Fig. 3. One is that there are six inversed partial products in Fig. 2 but none in unsigned multiplication. The other is that there is an individual term "1" to be added to produce S_4 in Fig. 2 but none in Fig. 3.

These two differences bring reconfiguring problem for signed multipliers to operation under different input precisions. In unsigned multiplier, if two operands with less precision than the designed multiplier length to be multiplied, it will not cause any

problem. For example, if using a 4×4 unsigned multiplier to calculate 101×011, just do it as 0101×0011. But in signed multiplier, there are some inversed terms inside. If these terms are not the corresponding partial products that should be inversed, incorrect result will occur. Also, the individual "1" also needs to appear on correct place. For example, if using the signed multiplier to multiply two signed operands 101 and 011, calculating them as 0101 and 0011 will cause wrong result. The reason is for a 3×3 signed multiplication process, X_2Y_0, X_2Y_1, X_1Y_2, and X_0Y_2 should be inversed and the individual "1" should appear in the column containing X_2Y_1. So unlike unsigned multiplier, signed multiplier cannot be automatically reconfigured for different input precisions.

Commonly used method to solve this problem is sign extension. Sign extension is to repeat the sign bit to fill the vacant high order bits in the operand until the length of the operand matches the length of multiplier. For the example in last paragraph, instead of 0101×0011, 1101×0011 should be used. The problem of sign extension method is that the extended sign bits are totally redundant and will cause more power and delay. When the difference between the length of multiplier and the length of operands is large, for example, calculating signed number 11×11 using a 16×16 multiplier, a lot of extended bits are in logic high. These bits will cause significant redundant power dissipation. The use of sign extension will also make the signed multiplier lose the natural power awareness as that exists in unsigned multiplier.

To solve these problems and improve the power awareness of deeply pipelined multipliers, a novel technique, 2-dimensional pipeline gating, is proposed in this paper. This technique is to gate the clock to the registers in both vertical direction (data flow direction in pipeline) and horizontal direction (within each pipeline stage). The additional

area cost to implement this technique to design array multipliers is very little and the overhead is hardly noticeable. The effectiveness will increase with the growth of the multiplication length. Simulation results show that an average power saving of 66% and an average latency reduction of 47% can be achieved for 16-bit unsigned array multiplier using 2-dimensional pipeline gating technique under equal input precision probabilities. And these numbers are 65% and 44%, in terms of average power saving and latency reduction, respectively, for 16-bit signed array multiplier. At the end of this paper, an application of these power-aware multipliers on FIR design is also included.

2. Previous Work

Several techniques have been developed to reduce the power dissipation in multipliers. Huang et al., [4] introduced a 2-dimensional signal gating method for low power array multiplier design. This approach provides gating lines for both multiplicand and multiplier operands. By deactivated different regions in the multiplier, power dissipation could be reduced. This approach is for non-pipelined array multiplier and cannot be extended to pipelined design because it cannot reduce the switching activities in registers. Bhardwaj et al., [2] introduced a selective method to design power-aware multiplier. This method is also for non-pipelined designs and brings high area cost. Meier et al., [5] introduced a polarity-inversion technique for the adders in signed multiplier. This technique does not solve the sign extension problem so that the multiplicands in lower precision still cannot be processed directly. Lee et al, [6] introduced a reduced architecture based on the redundancy of lower order bits in some DSP applications. This technique is not for general use and does not solve the sign extension problem in signed multiplier.

Kim et al., [7] introduced a clock gating method to design reconfigurable multiplier. This method is to selectively disable pipeline stages by gating clocks and to select correct results by multiplexers. Very little additional area cost is needed (only several AND2 gates and multiplexers) to implement this technique. Good power and latency saving can be achieved due to the reduced switching activities of registers in corresponding pipeline stages. The outputs of the multiplier are selected from different stages to ensure the correctness and obtain latency reduction. The basic idea of this method is shown in Fig. 4. This technique can be seen as 1-dimensional pipeline gating because it only considers gating clocks to unnecessary stages along data flow direction. As the computational width of multiplier growing from 4-bit, 8-bit, to 32-bit and 64-bit, 1-dimensional pipeline gating is far from enough.

As shown later in this paper, 2-dimensional pipeline gating is able to achieve much more power saving thus greatly improves the power awareness in pipelined multipliers. Also, 2-dimensional pipeline gating only needs the same additional hardware as 1-dimensional technique, and has the same latency reduction. For a 16-bit pipelined array multiplier, if the probabilities of all input precisions are assumed to be equal, 2-dimensional pipeline gating can have 66% power saving over the original design, while 1-dimensional technique only has 25.7%. In the rest of the paper, 2-D pipeline gating is used to represent 2-dimensional pipeline gating technique while 1-D pipeline gating is used for 1-dimensional pipeline gating.

3. 2-Dimensional Pipeline Gating Technique

As stated before, 2-D pipeline gating is to gate clock to the registers in both vertical direction (data flow direction in pipeline) and horizontal direction (within each pipeline

stage), while 1-D pipeline gating technique gates clock in vertical direction only. The principle of 2-D pipeline gating technique is shown in Fig. 5.

In the 1-D pipeline gating scheme shown in Fig. 4, the system clock is gated by different gating signals to generate sub-clocks. Each sub-clock is connected to one pipeline stage and drives all registers in that stage. If under a certain case the results could come directly from stage 3, then the *Gating Signal 4* is set effective and *Clock 4* is disabled. The output of register 3 is then bypassed through a multiplexer, which is also controlled by the clock gating signals, to the system output. Since the *Clock 4* is disabled, the total number of switching is reduced. Also, since the system output now comes from stage 3 instead of stage 4, the pipeline latency is reduced.

In a real pipeline, the data going through a register in a certain pipeline stage is most likely to correlate with the data going through the register in the previous stage. So if under a certain case one pipeline stage could be disabled, some of the registers in its previous stage may also be redundant and could be disabled too. This happens especially in such pipelines in which only some data are processed in this stage, others are just passed to the next stage. Computer arithmetic circuits like multipliers and adders always contain such pipelines [8]. By applying 2-D pipeline gating technique to these circuits, significant power saving can be achieved.

In the 2-D pipeline gating scheme shown in Fig. 5, when under a certain case pipeline stage 4 could be disabled, some of the registers in previous stages (the first two registers in stage 1, 2, and 3) could also be disabled if the data going through them was to be processed only in stage 4 thus is no longer useful. These registers can be disabled by using *Clock 4* as their clock inputs. For the same reason, if stage 3 needs to be disabled,

the third and fourth registers in stage 1 and 2 could also be disabled. The total number of transition is further reduced compared to that in 1-D pipeline gating system. As the number of registers in each stage as well as the total number of stages in the pipeline (pipeline depth) increase, this further benefit becomes more and more significant. As shown later in this paper, the 16-bit unsigned multiplier using 2-D pipeline gating has more than 54% power saving over the same multiplier using 1-D technique. And this number is 55.6% for signed multiplier.

4. Power-aware Unsigned Array Multiplier Design

To design power-aware pipelined multiplier using 2-D pipeline gating technique, firstly the multiplication process should be examined. The 4×4 unsigned multiplication process is shown in Fig. 3.

In Fig. 3, X and Y are inputs while S is the output. When the input precision is 4, for example, calculating 1111×1111, S is generated based on all inner partial products. If the input precision is 3, for example, calculating 0111×0111, the partial products containing X_3 or Y_3 are all zero (these products are enclosed by a circle in Fig. 3), and S only has six digits instead of eight. From a reset-to-zero state, there is no need to let registers propagate these zeros because the reset state of register is zero. So clocks connected to these registers can be disabled. If the input precision is 2, for example, calculating 0011×0011, the partial products containing X_2 or Y_2 (the ones enclosed by a rectangular in Fig. 3) can also be disabled. If the input precision is 1 as 0001×0001, the partial products enclosed by an ellipse in Fig. 3 containing X_1 or Y_1 can be disabled. As the length of output S decreases, the number of necessary pipeline stages is also reduced. The

circuit structure of a 4-bit pipelined unsigned array multiplier using 2-D pipeline gating technique is shown in Fig. 6.

In Fig. 6, “HA” represents half adder; “FA” represents full adder; “Reg” represents register; “*n*-1” represents *n*-to-1 multiplexer. Current input precision information is provided through four gating signals from CPU. These signals are combined with system clock to generate four sub-clocks, which are connected to the corresponding registers in all pipeline stages. Under a certain input precision, one or more sub-clocks may be disabled. The registers connected to these sub-clocks will not function during the calculation. The multiplexers select correct outputs from corresponding stages. For example, while performing 0001× 0001, only S_0 has useful value. This value is selected from the stage right after the AND matrix. Except for this register and the two registers in the first stage for X_0 and Y_0, all other registers do not function because their clocks have been disabled. The power dissipation is reduced significantly. The output S_0 is from the first stage after the AND matrix instead of the eighth one, thus the pipeline latency has also been reduced by a factor of eight.

The detection of current input precision is a typical interrupt-response scheme for a CPU. For example, when the user of digital camera pushes the button to reduce the resolution, an interrupt is sent to the CPU. Then CPU reads the corresponding register and sets up the clock gating signals based on the register value. So the additional area cost is very low, just a few AND gates and some multiplexers are needed. The clock gating signals are also used as the control signals of these multiplexers.

Based on the discussion above, a set of unsigned array multipliers were designed. The computation lengths of these multipliers are 4-bit, 8-bit, and 16-bit, respectively. To

compare between different numbers of pipeline stages, the 16-bit multiplier was pipelined into 8, 16, and 32 stages. Both 1-D and 2-D pipeline gating techniques have been applied to each multiplier. These multipliers were synthesized by Synopsys Design Analyzer and simulated in Powermill. During the simulation, the multipliers were given data in different input precisions. The power dissipation were recorded and compared. The simulation result comparisons are shown in Fig. 7 to 12.

In Fig. 7 to 12, "Original" represents the simulation data of the unchanged pipelined designs; "1-D" and "2-D" represent the simulation data of the designs using 1-D and 2-D pipeline gating techniques, respectively.

From these figures, several observations are made:

1. Among the three multipliers in each figure, the designs using 1-D and 2-D pipeline gating techniques have lower power dissipations compared to the original designs under different input precision.
2. Among all three multipliers, the designs using 2-D pipeline gating techniques show significant power savings over the corresponding designs using 1-D pipeline gating technique. This advantage is not large in 4-bit multiplier (14.3% under equal input precision probability), but becomes much greater in 8-bit multiplier (41.5% under equal input precision probability), and is quite significant in 16-bit multiplier (54.4% under equal input precision probability for 32-stage design). As shown in Fig. 7 to 12, the data of designs using 1-D pipeline gating technique show convex curves while that of designs using 2-D pipeline gating technique show concave curves. The reason for this difference is that as the length of multiplier goes up, the number of registers in horizontal

direction as well as in vertical direction increases sharply. 1-D pipeline gating technique only deals with the vertical pipeline stage increment, while 2-D pipeline gating technique controls the registers in both directions. Actually, the largest difference between these two techniques occurs when the current input precision is half the designed precision. Under this case, there are lots of registers in middle pipeline stages that are propagating redundant zeros. 1-D technique cannot deal with them. But 2-D pipeline gating technique has the ability to disable them accurately.

3. The overhead of implementing 1-D and 2-D techniques are the same. It is very small (0.03% in 32-stage 16-bit multiplier).
4. Peak power dissipation affects the system reliability in operating under power constraints. 1-D and 2-D pipeline gating techniques both have the ability to reduce system peak power dissipation. But the same as average power dissipation, 2-D technique has great advantage over 1-D technique under different input precisions.
5. When the number of pipeline stages is reduced, the average power savings of 1-D and 2-D pipeline gating designs are also reduced. The reason is that the number of registers is reduced with the decrement of pipeline stages. The number of redundant clock switching in the original design is also reduced. So the power saving percentage becomes less. But from Fig. 11 and 12, 2-D designs still show significant average power savings over the original designs and 1-D designs.

6. In Fig. 12, the curve of 1-D design looks step-like by every two data. The reason is for 8-stage pipelined array multiplier, each pipeline stage outputs four bits of the final calculation result. So every input precision decrement by two will cause one pipeline stage to be disabled. For example, 16×16 and 15×15 both need all 8 stages to give the final result; but 14×14 and 13×13 only need 7 stages. For 2-D design in the same figure, with every decrement of input precision, there are always some registers becoming redundant and are disabled. Although the number of stages is the same as 1-D design, the actual number of registers working within each stage is quite different. That's why the curve of 2-D design does not look step-like.

The pipeline latency reduction of the designs using 1-D and 2-D pipeline gating techniques is the same. The comparison data of latency saving as well as other data are shown in Table 1 and 2.

5. Application of 2-D Pipeline Gating Technique to Design Power-aware Signed Array Multiplier

To avoid the sign extension problem, different research and methods have been proposed. A selective method is used in this section to make the signed multiplier have good power awareness. Several important analysis and modifications beside the 2-D pipeline gating technique have to be made. The pipeline stage right after the AND matrix stage of a 4×4 power-aware signed multiplier is shown in Fig. 13.

In Fig. 13 the partial products X_3Y_0, X_3Y_1, X_3Y_2, X_0Y_3, X_1Y_3, and X_2Y_3 are inversed by connecting to NAND gates. Another input (not shown in Fig. 13) called *const_in* is added to the proper adder as the individual "1". Inner products X_2Y_1, X_1Y_2, X_0Y_2, X_2Y_1, X_0Y_1, and

X_1Y_0 are connected to 2-to-1 multiplexers with their inversions. These multiplexers are controlled by different control signals indicating the current input precision. These signals are just as the clock gating signals issued by CPU. The outputs of these multiplexers along with all other outputs of AND/NAND gates are connected to the registers forming next pipeline stage. These registers, just as designing power-aware unsigned array multipliers, are connected to different gated clocks controlled by clock gating signals based on current input precision.

When current input precision is 4×4, all multiplexers are switched to the non-inversed data; all four types of clocks are enabled; the *const_in* bit is set to logic high. Then the multiplier is able to perform 4×4 signed multiplication as shown in Fig. 2.

When current input precision is 3×3, the multiplexers for X_2Y_1, X_1Y_2, X_0Y_2, and X_2Y_1 are switched to their inversed data; *CLK-3* is disabled; the *const_in* bit is set to logic low. Note that the clock connected to the output of NAND gate whose input is X_3Y_0 is *CLK-2*, not *CLK-3*. Since X_3 and Y_3 are all zero, this NAND gate will generate logic high. This "1" becomes the individual "1" needed for 3×3 multiplication.

When current input precision is 2×2, the multiplexers are all switched to the inversed data; both *CLK-2* and *CLK-3* are disabled; the *const_in* bit is still logic low. For the same reason, the clock connected to the output of NAND gate whose input is X_2Y_0 is *CLK-1*, not *CLK-2*. This bit becomes the individual "1" for 2×2 multiplication. Note, there is no 1×1 multiplication for signed multiplier because there has to be a sign bit.

By applying the modifications above, the 4×4 pipelined signed multiplier is able to perform 3×3 and 2×2 multiplication without sign extension. During 3×3 and 2×2 multiplication process, the gated registers will not function, so that the power dissipation

is saved. Also, the redundant power dissipation caused by sign extension is avoided. The same as in applying 1-D or 2-D technique on unsigned multipliers; the output bits can be selected from different pipeline stages prior to the last stage. So the pipeline latency can also be reduced.

Based on the technique described above, just as the testing scheme of unsigned multiplier, nine pipelined signed array multipliers with lengths of 4-, 8-, and 16-bit are designed as original architecture, the designs using 1-D pipeline gating technique, and the power-aware designs using 2-D pipeline gating technique. All designs are also synthesized by Synopsys Design Analyzer, and then simulated in Powermill. The results comparisons are shown in Fig. 14 to 17.

From Fig. 14 to 17 several observations could be made:

1. The same as in unsigned multiplier results, the 2-D designs have great advantage over the other two groups in terms of average and peak power dissipation. There are two reasons for this difference: one is the same as in unsigned multiplier design, which is, 2-D technique not only gates the redundant pipeline stages like 1-D technique does, but also disables the unused registers within the useful pipeline stages. The other reason is the use of sign extension brings more switching to 1-D designs. But the 2-D power-aware designs do not have this problem. In 16-bit multiplier, the 2-D design has 55.6% average power saving and 55.8% peak power saving over the design using 1-D technique.
2. The overheads of the 2-D designs are a little larger than that in unsigned multiplier. But they are still very small, only 0.23% in 16-bit design.

The latency reductions of the 2-D designs are the same as those designs using 1-D technique. The comparison in data form is shown in Table 3.

6. Application of Power-aware Multipliers on FIR Filter Design

As an application of power-aware multipliers, a high-throughput, power-aware FIR filter design method is introduced in this section. FIR filters are essential elements in DSP systems. There are different implementations of FIR filters. To shorten the critical path in order to achieve high throughput, Data-Broadcast structure is used in this paper [9]. A 3-tap Data-Broadcast FIR filter is shown in Fig. 18. There are three multipliers and two adders. The input-output relationship is shown in (2).

$$y(n) = a \cdot x(n) + b \cdot x(n-1) + c \cdot x(n-2) \tag{2}$$

The dashed line in Figure 18 shows the critical path. The length of this critical path is $T_M + T_A$, where T_M is the time taken for multiplication and T_A is the time taken for addition. The period of operating clock must be longer than this length. This results in a very low clock rate. If this FIR is used in a real-time application, the sampling frequency, f_{sample}, must be less than the operating frequency of this FIR filter, that is

$$f_{sample} \leq \frac{1}{T_M + T_A} \tag{3}$$

To improve the throughput of the FIR filter, one commonly used method is to pipeline the multipliers. Since the multiplication time T_M is usually much larger than the addition time T_A, much shorter critical path length can be achieved by carefully balancing the pipeline stages. Figure 19 shows a pipelining scheme for FIR filter.

In Fig. 19, each multiplier in Fig. 18 is divided into two pipeline stages. A series of registers is added between the two sub-multipliers. The time taken by each stage of the

multiplier is denoted by T_{M1} and T_{M2}, respectively; and the delay time in the added registers is denoted by T_{DR}. If the pipeline is perfectly balanced so that $T_{M1} = T_{DR} + T_{M2} + T_A$, the critical path shown as dashed line in Fig. 17 is much shorter than that of Fig. 18.

By pipelining multipliers can only achieve limited throughput improvement. Assume the number of pipeline stages that the multipliers in Fig. 19 can be divided to approaches infinity; the slowest stage will contain the adder and a very small part of multiplier. So the length of critical path is approaching T_A. When the word length of input data and coefficients is short, T_A is small enough for the FIR to operate in high sampling frequency. In recent years, the word length of FIR filter has been growing from 8-bit, 16-bit, up to 32-bit and 64-bit. Under long word length condition, addition also takes significant time. Instead of pipelined multipliers, adders become bottleneck in these FIR filters under such conditions.

To further improve throughput of FIR filters, the critical path in addition process needs to be shortened too. So adders, as well as multipliers, need to be pipelined. Pipelining one adder changes the timing relationship between the two inputs of the next adder. Unlike pipelining multipliers, which doesn't change the relative timing sequence between adder inputs, pipelining adders just likes adding delay elements to the paths between adders. So additional delay elements need to be added between next adder and its corresponding multiplier. The goal is to maintain the timing difference between the two inputs of the adder as one clock cycle. The revised FIR filter structure is shown in Fig. 20.

For a balanced pipeline design, each pipeline stage takes almost the same calculation time. By fine-grain pipelining multipliers and adders, very high throughput can be achieved. For an *N*-tap FIR filter, if the multipliers are divided to *M* pipeline stages, and the adders are divided to *P* pipeline stages, the FIR filter structure is shown in Fig. 21.

Along with the achieved high throughput, pipelining multipliers and adders also causes two problems. Firstly the power dissipation will become larger because some registers are added between the pipeline stages; secondly the total latency from the input to output also becomes larger because the pipelined addition paths are longer. To solve these problems, 2-D pipeline gating technique is used to design power-aware multipliers and adders. These elements are able to scale their power and latency with the changing of input precision. To maintain the correctness of the calculation, 2-D pipeline gating technique needs also be implemented on the additional delay elements to select output from the corresponding pipeline stage thus keep the timing relationship. Based on these discussions, a set of 4-tap FIR were designed and tested. The word length of input data and coefficients are all 16-bit. So the multipliers are 16×16 and the adders are 32×32. These designs are synthesized by Synopsys Design Compiler and then simulated in Powermill. The simulation results are shown in Fig. 22-24. Several discussions are listed as below:

1. The throughput of the pipeline is determined by the slowest stage. Since multipliers and adders are fine-grain pipelined, the delay in each pipeline stage is very small. The technology used in synthesis process is 0.24μm static CMOS logic. Simulation results show that the designed FIR filter is able to work under

1.25GHz clock rate. If using dynamic gate or transistors of smaller channel length, even higher throughput is expected.

2. The same as in multipliers design, the average power dissipation as well as peak power dissipation are significantly reduced by applying 2-D pipeline gating technique. The power reduction rate of 2-D design is much better than that of 1-D design.
3. On reducing pipeline latency, 1-D and 2-D gating techniques have the same rate of advantage. By selecting the correct outputs from corresponding stages, the total pipeline latency is significantly reduced.

7. Conclusion

A novel low power design technique, 2-dimensional pipeline gating, is proposed in this paper. This technique is explained by designing both signed and unsigned power-aware pipelined multipliers. For signed multipliers, it also avoids the sign extension problem while processing low input precision multiplicands. The relation and difference between this 2-D technique and existing 1-D technique are discussed. A set of array multipliers is designed using both techniques. Simulation results show that 2-D pipeline gating technique has great advantage over 1-D technique in terms of average and peak power savings while maintaining the same latency reduction rate. 2-D pipeline technique can be applied with very little additional area like applying 1-D technique so that the overhead is hardly noticeable. An application of this technique on power-aware high-throughput FIR filter design is also included.

REFERENCES

[1] Farid N. Najm, A Survey of Power Estimation Techniques in VLSI Circuits, IEEE Transactions on VLSI Systems, Volume 2, Issue 4, Dec. 1994, pages 446-455

[2] Manish Bhardwaj, R. Min, and A. P. Chandrakasan, Quantifying and Enhancing Power Awareness of VLSI Systems. IEEE Transactions on VLSI Systems. 2001, Volume 9, Issue 6, pages 757-772.

[3] S. H. Nawab, J. M. Winograd, Approximate signal processing, 1995 International Conference on Acoustics, Speech, and Signal Processing, May 9-12, 1995, Page(s): 2857 -2860 vol.5

[4] Z. Huang. M. D. Ercegovac, Two-dimensional signal gating for low-power array multiplier design, IEEE International Symposium on Circuits and Systems, 2002, Volume 1, pages 489-492

[5] P. C. H. Meier, R. A. Rutenber, L. R. Carley, Inverse polarity techniques for high-speed/low-power multipliers, International Symposium on Low Power Electronics and Design, pages 264-266, 1999

[6] K. H. Lee, C. S. Rim, A hardware reduced multiplier for low power design, Proceedings of the second IEEE Asia-Pacific Conference on ASICs. Page: 331-334, 2000

[7] S. Kim, M. C. Papaefthymiou, Reconfigurable low energy multiplier for multimedia system design, Proceedings of IEEE Computer Society Workshop on VLSI, 2000

[8] B. Parhami, Computer arithmetic – algorithms and hardware designs, Oxford University Press, 1999

[9] K. K. Parhi, VLSI Digital Signal Processing Systems, John Willey & Sons Inc., 1999

FIGURE AND TABLE CAPTIONS

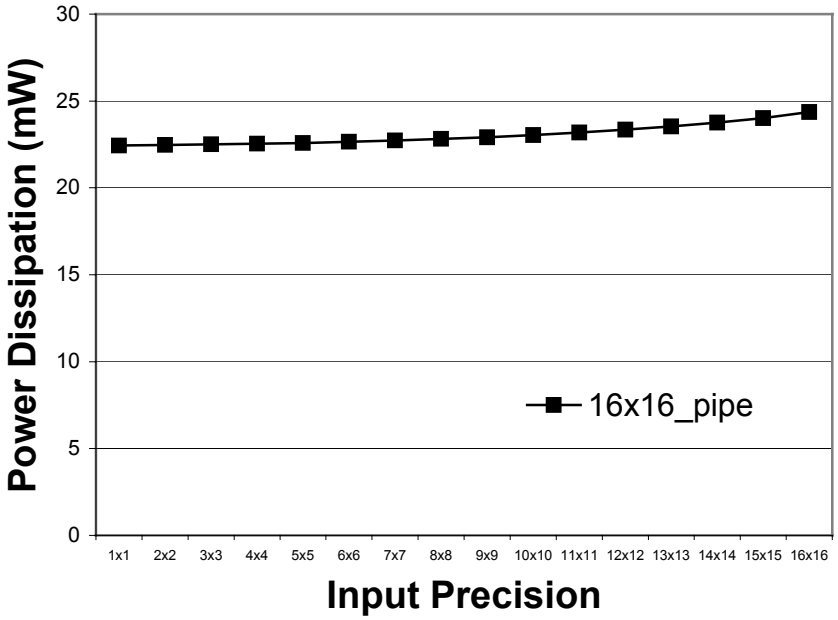
Power Dissipation (mW)
30
25
20
15
10
5
0
16x16_pipe
1x1
2x2
3x3
4x4
5x5
6x6
7x7
8x8
9x9
10x10
11x11
12x12
13x13
14x14
15x15
16x16
Input Precision

Fig. 1

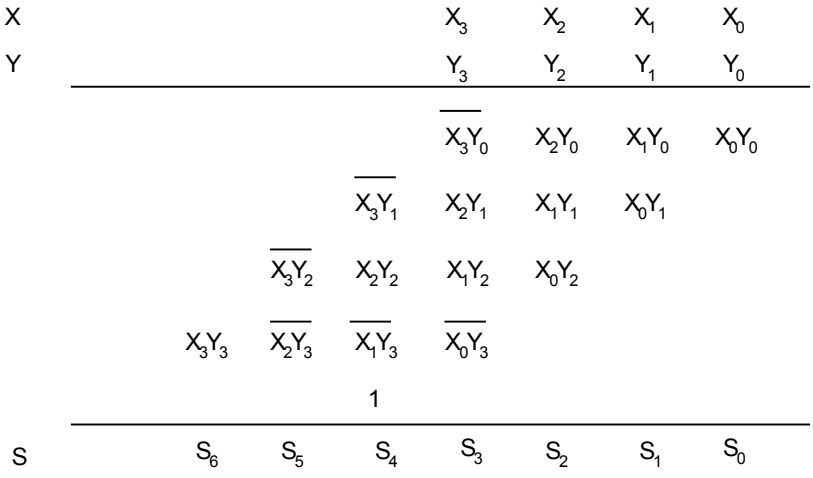
X
Y
S
X_3 X_2 X_1 X_0
Y_3 Y_2 Y_1 Y_0
$\overline{X_3Y_0}$ X_2Y_0 X_1Y_0 X_0Y_0
$\overline{X_3Y_1}$ X_2Y_1 X_1Y_1 X_0Y_1
$\overline{X_3Y_2}$ X_2Y_2 X_1Y_2 X_0Y_2
X_3Y_3 $\overline{X_2Y_3}$ $\overline{X_1Y_3}$ $\overline{X_0Y_3}$
1
S_6 S_5 S_4 S_3 S_2 S_1 S_0

Fig. 2

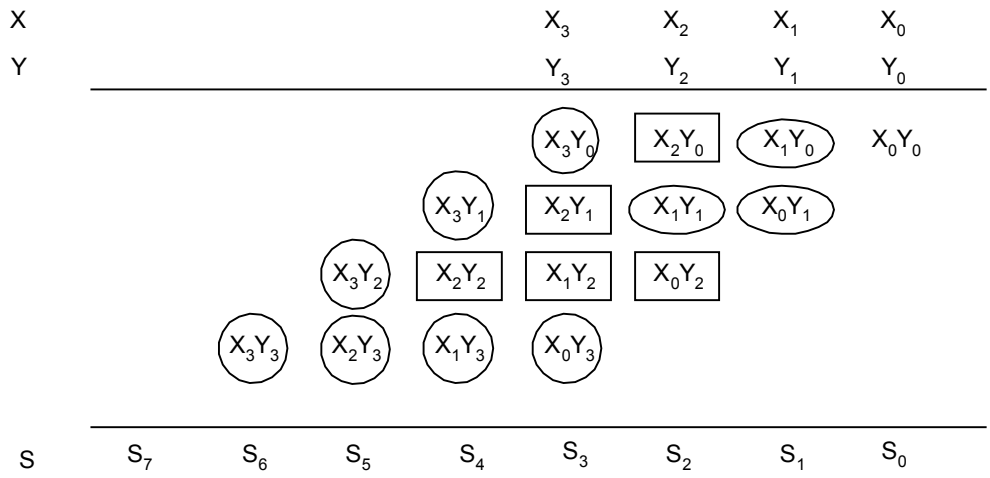
X
Y
X3 X2 X1 X0
Y3 Y2 Y1 Y0
X3Y0 X2Y0 X1Y0 X0Y0
X3Y1 X2Y1 X1Y1 X0Y1
X3Y2 X2Y2 X1Y2 X0Y2
X3Y3 X2Y3 X1Y3 X0Y3
S
S7 S6 S5 S4 S3 S2 S1 S0

Fig. 3

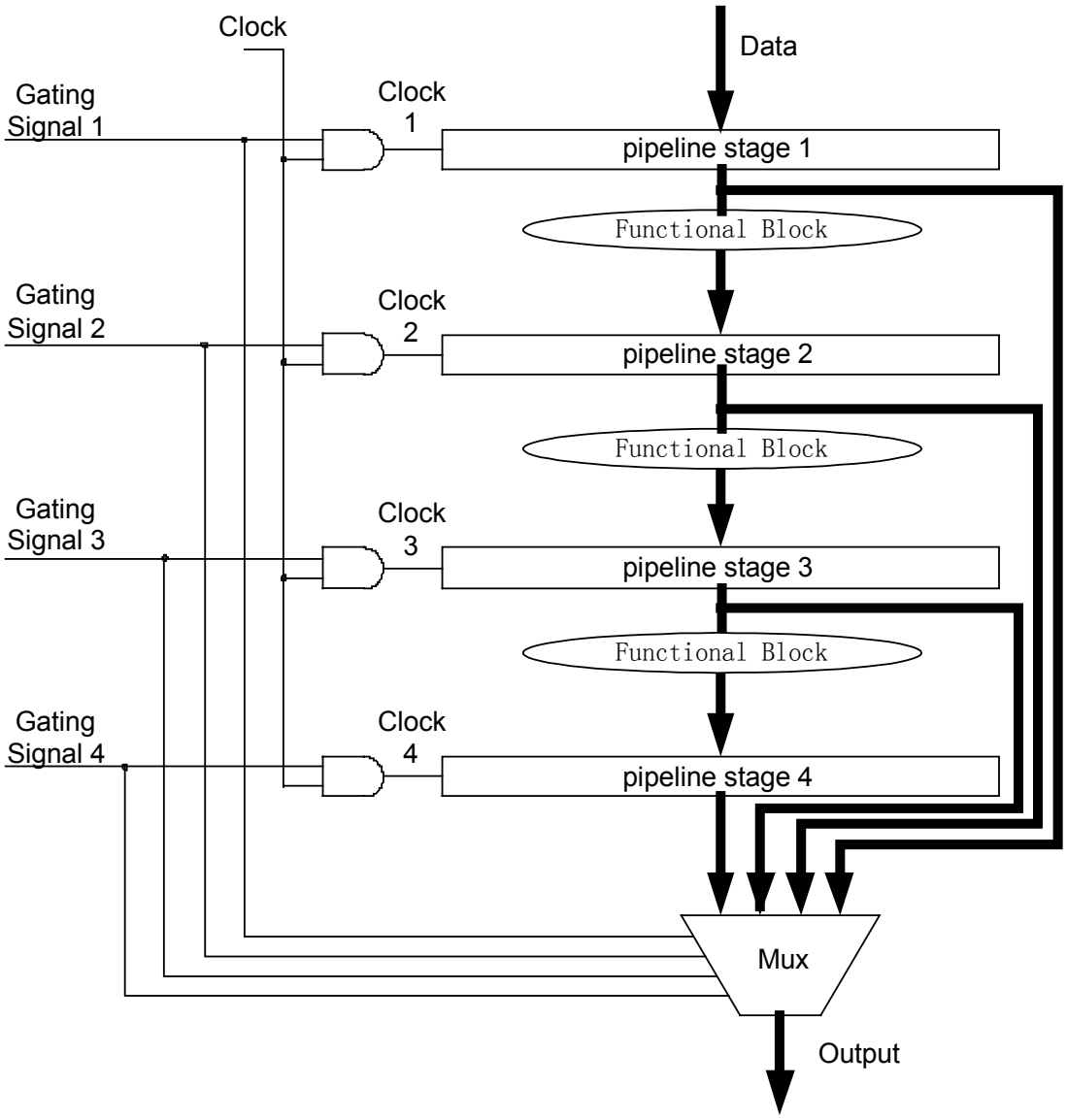
Clock
Data
Gating
Signal 1
Clock
1
pipeline stage 1
Functional Block
Gating
Signal 2
Clock
2
pipeline stage 2
Functional Block
Gating
Signal 3
Clock
3
pipeline stage 3
Functional Block
Gating
Signal 4
Clock
4
pipeline stage 4
Mux
Output

Fig. 4

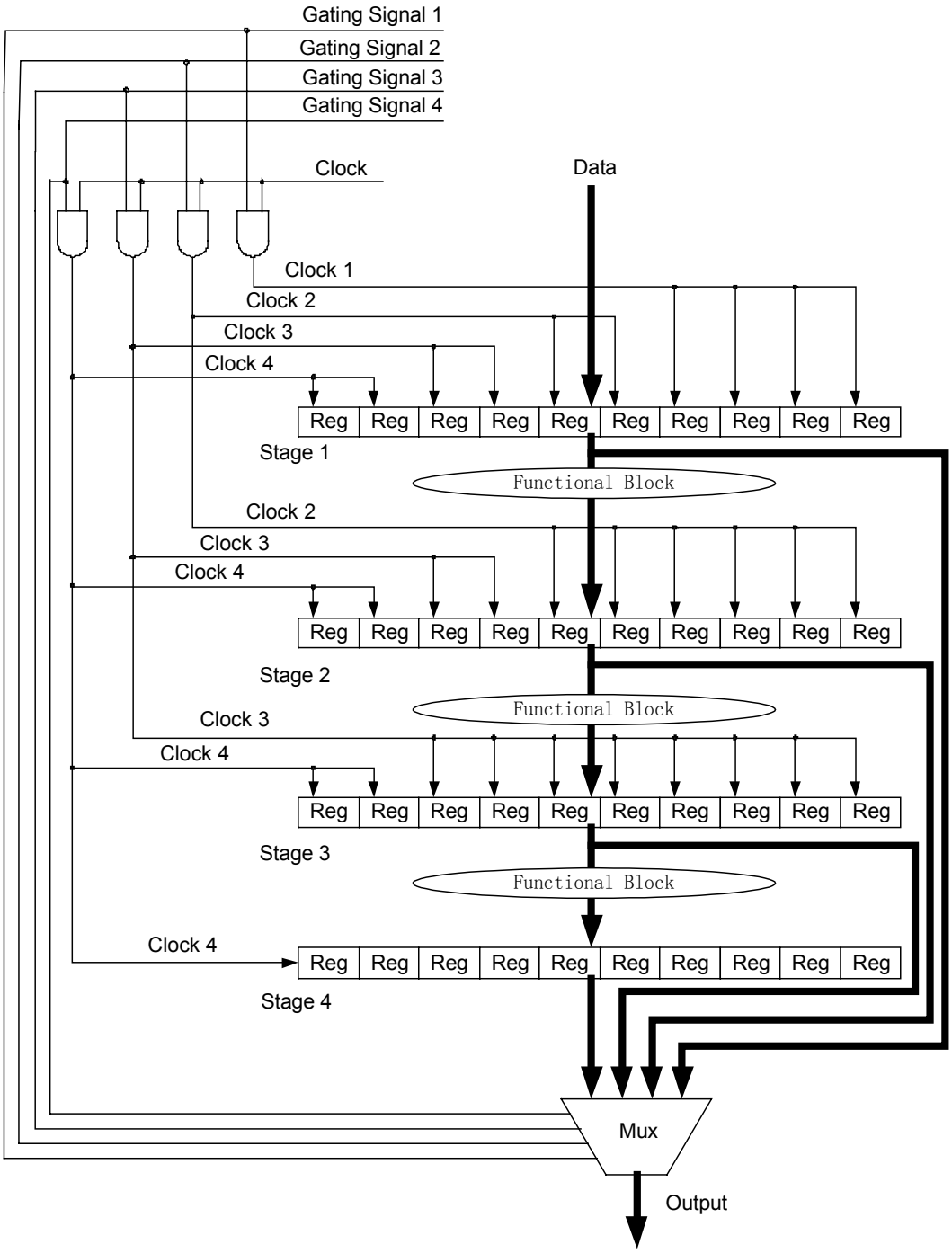
Gating Signal 1
Gating Signal 2
Gating Signal 3
Gating Signal 4
Clock
Data
Clock 1
Clock 2
Clock 3
Clock 4
Reg
Stage 1
Functional Block
Clock 2
Clock 3
Clock 4
Reg
Stage 2
Functional Block
Clock 3
Clock 4
Reg
Stage 3
Functional Block
Clock 4
Reg
Stage 4
Mux
Output

Fig. 5

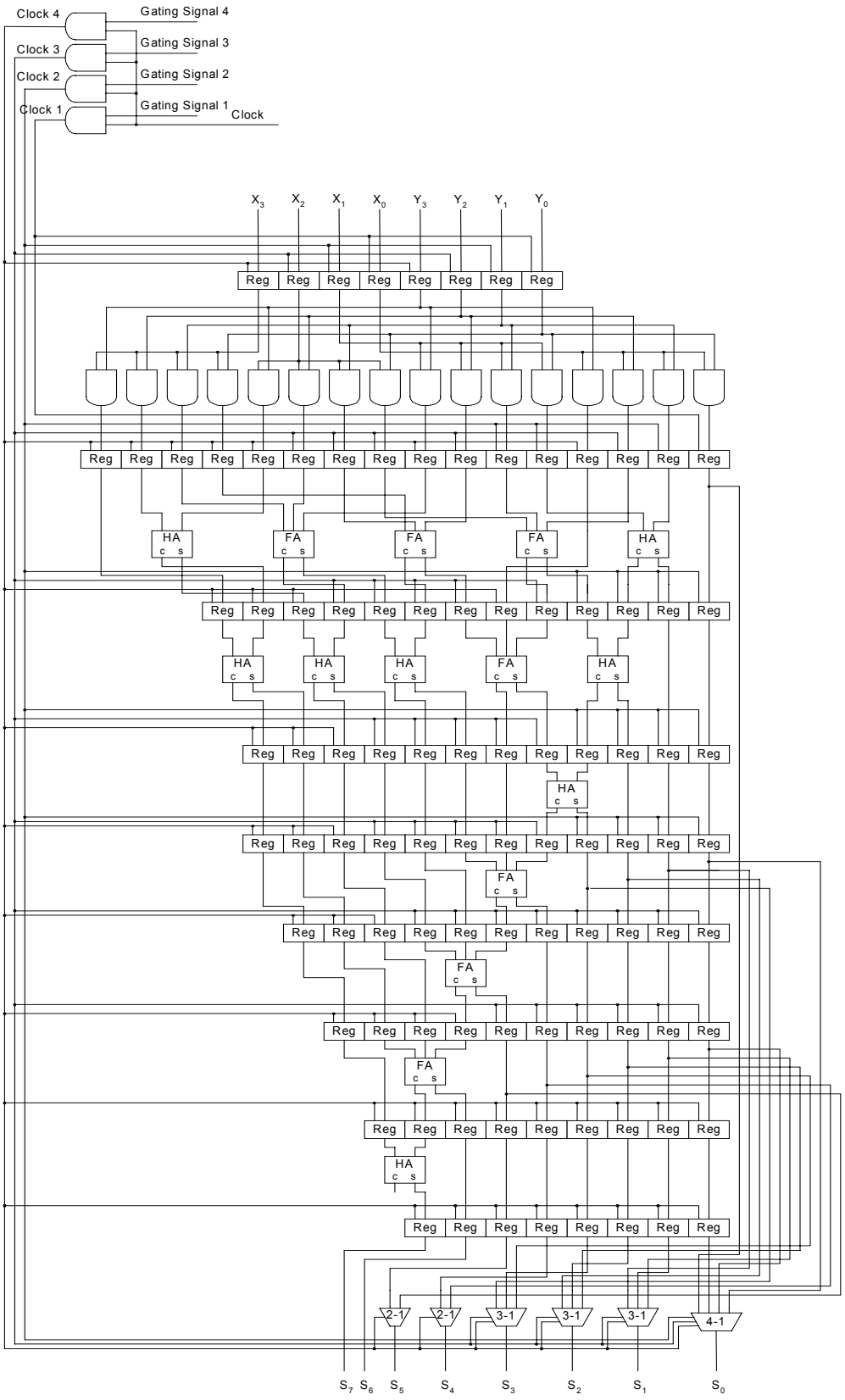
Clock 4
Gating Signal 4
Clock 3
Gating Signal 3
Clock 2
Gating Signal 2
Clock 1
Gating Signal 1
Clock
X3 X2 X1 X0 Y3 Y2 Y1 Y0
Reg
HA
FA
c s
2-1
3-1
4-1
S7 S6 S5 S4 S3 S2 S1 S0

Fig. 6

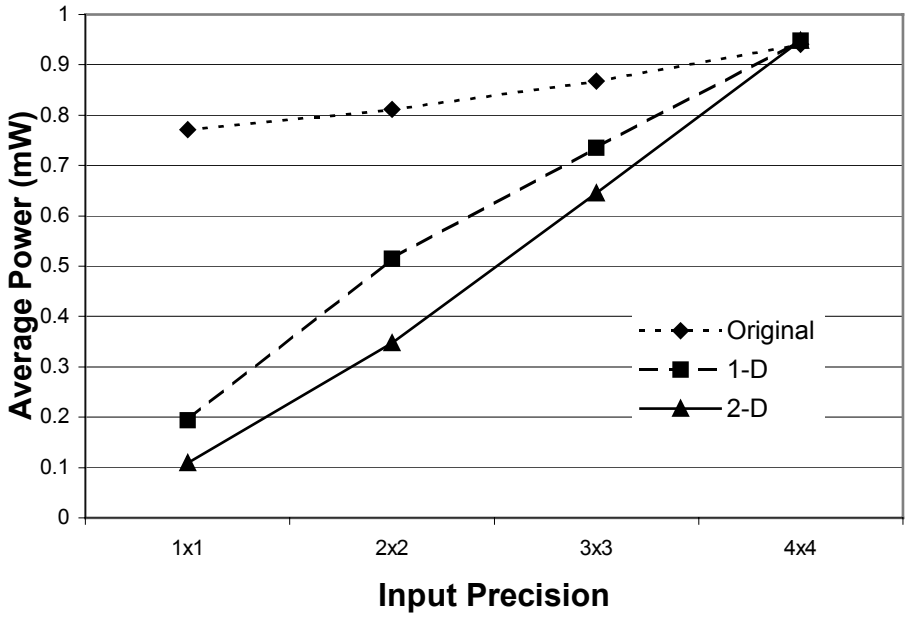
Average Power (mW)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1x1
2x2
3x3
4x4
Input Precision
Original
1-D
2-D

Fig. 7

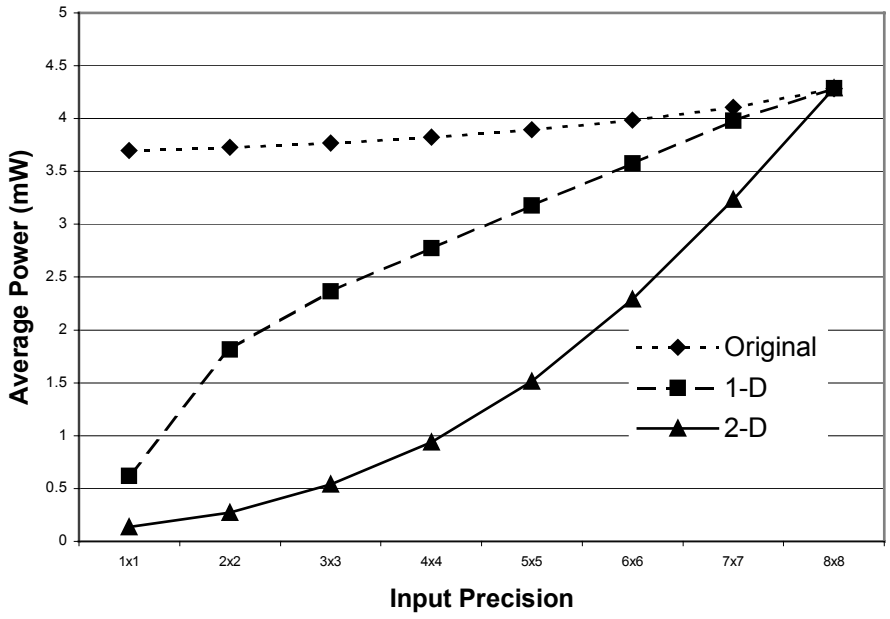
Average Power (mW)
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1x1
2x2
3x3
4x4
5x5
6x6
7x7
8x8
Input Precision
Original
1-D
2-D

Fig. 8

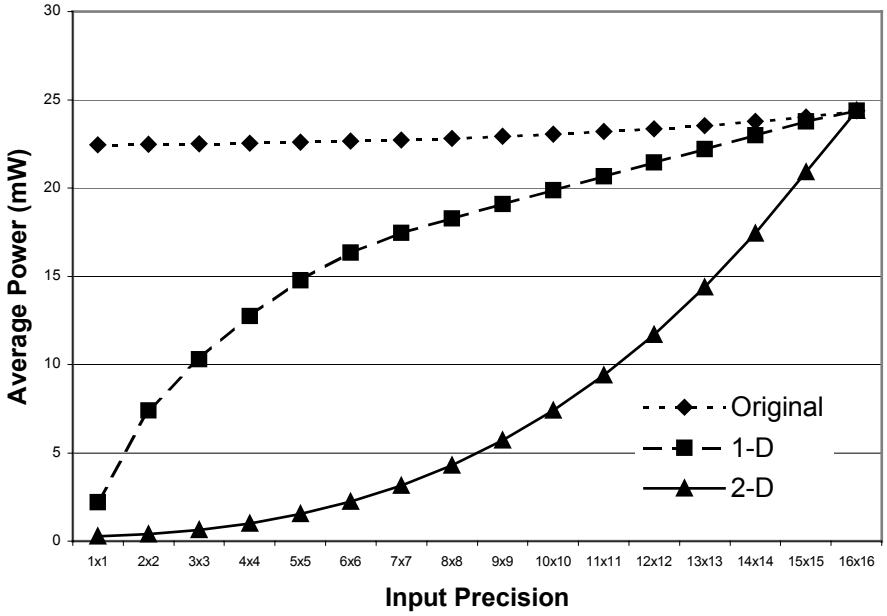
Average Power (mW)
30
25
20
15
10
5
0
1x1
2x2
3x3
4x4
5x5
6x6
7x7
8x8
9x9
10x10
11x11
12x12
13x13
14x14
15x15
16x16
Input Precision
Original
1-D
2-D

Fig. 9

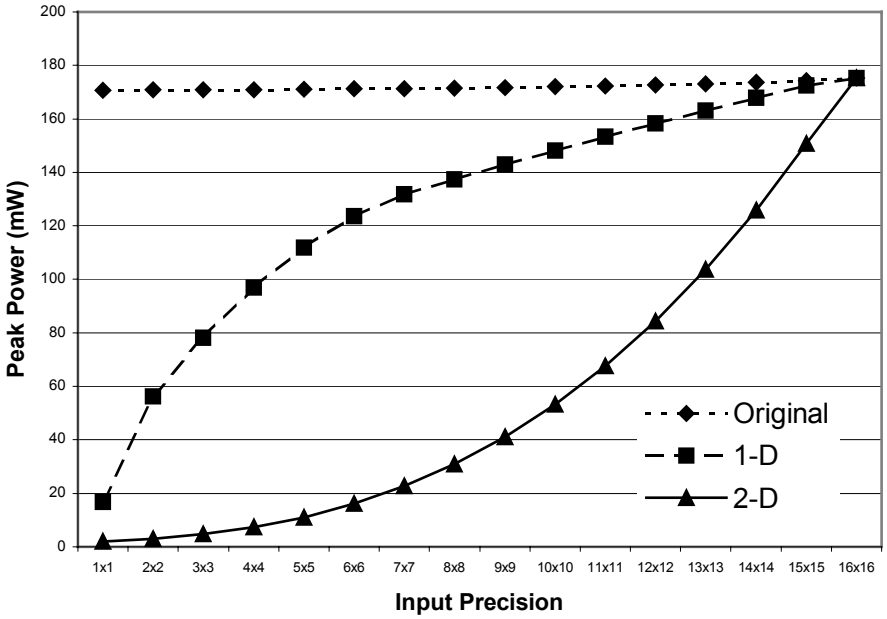
Peak Power (mW)
200
180
160
140
120
100
80
60
40
20
0
1x1
2x2
3x3
4x4
5x5
6x6
7x7
8x8
9x9
10x10
11x11
12x12
13x13
14x14
15x15
16x16
Input Precision
Original
1-D
2-D

Fig. 10

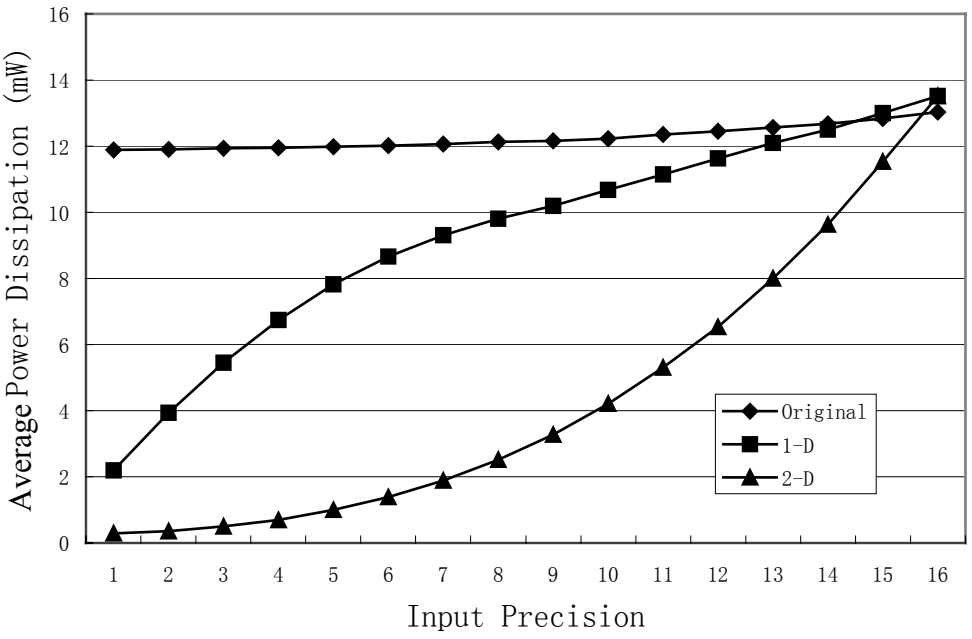
Average Power Dissipation (mW)
Input Precision
Original
1-D
2-D

Fig. 11

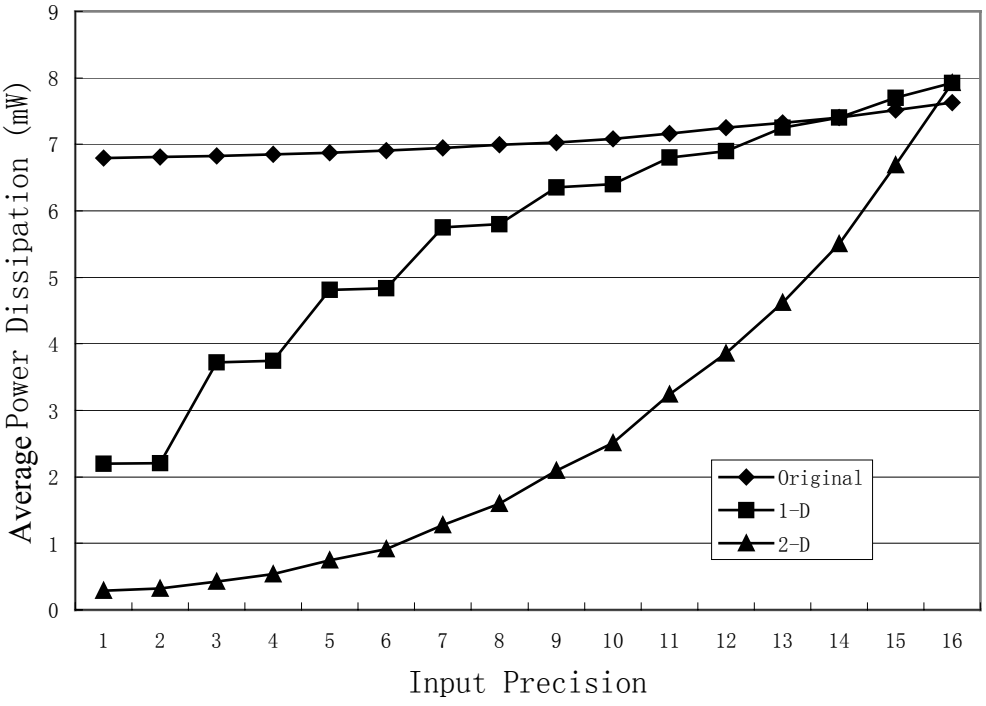
Average Power Dissipation (mW)
Input Precision
Original
1-D
2-D

Fig. 12

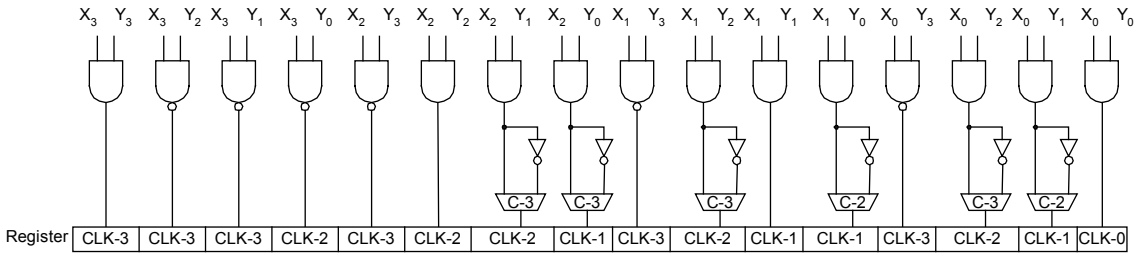
X3 Y3 X3 Y2 X3 Y1 X3 Y0 X2 Y3 X2 Y2 X2 Y1 X2 Y0 X1 Y3 X1 Y2 X1 Y1 X1 Y0 X0 Y3 X0 Y2 X0 Y1 X0 Y0
C-3
C-3
C-3
C-2
C-3
C-2
Register
CLK-3 CLK-3 CLK-3 CLK-2 CLK-3 CLK-2 CLK-2 CLK-1 CLK-3 CLK-2 CLK-1 CLK-1 CLK-3 CLK-2 CLK-1 CLK-0

Fig. 13

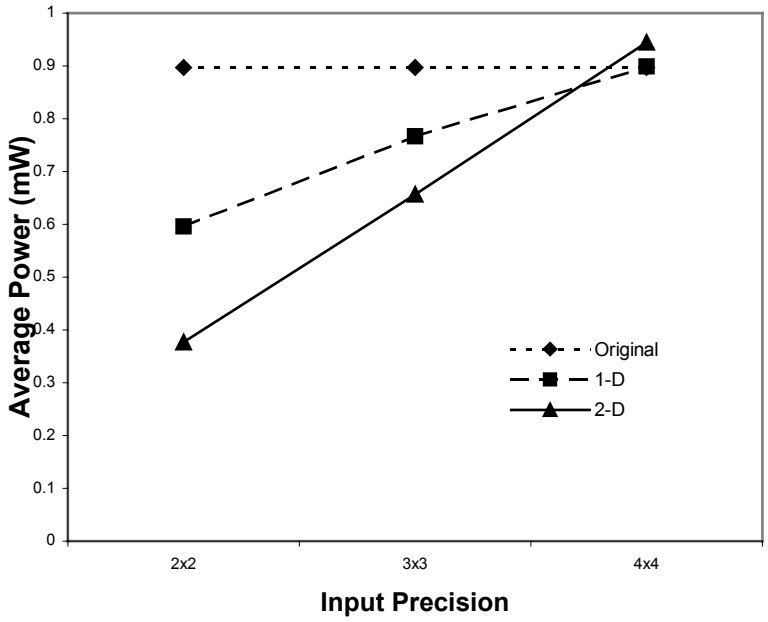
Average Power (mW)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2x2
3x3
4x4
Input Precision
Original
1-D
2-D

Fig. 14

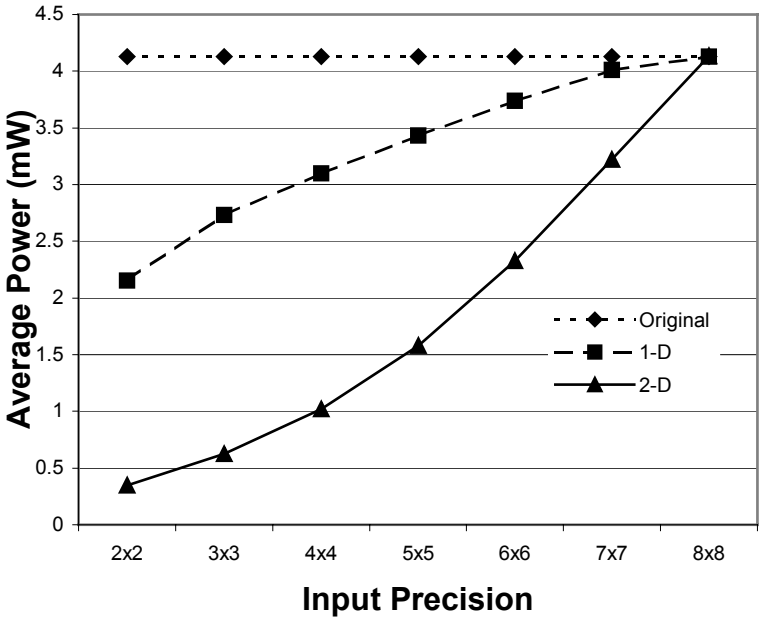
Average Power (mW)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
2x2
3x3
4x4
5x5
6x6
7x7
8x8
Input Precision
Original
1-D
2-D

Fig. 15

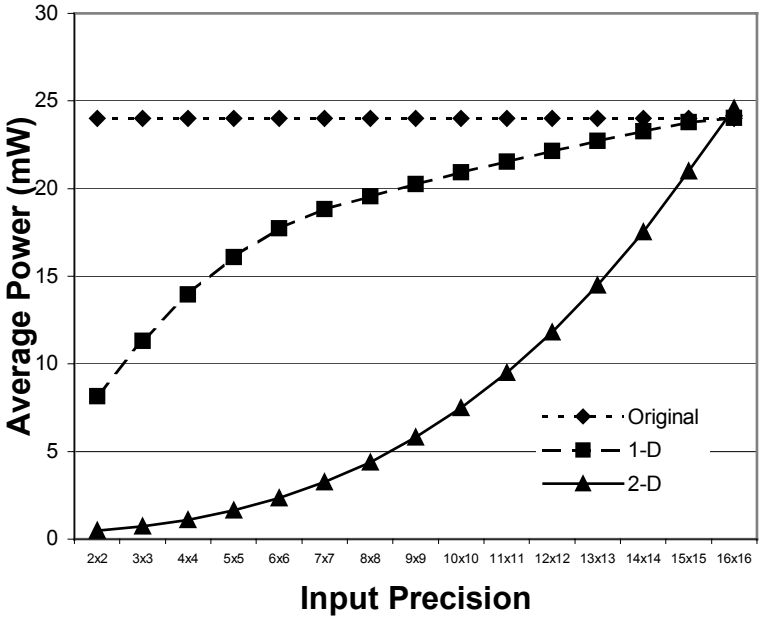
Average Power (mW)
30
25
20
15
10
5
0
2x2
3x3
4x4
5x5
6x6
7x7
8x8
9x9
10x10
11x11
12x12
13x13
14x14
15x15
16x16
Input Precision
Original
1-D
2-D

Fig. 16

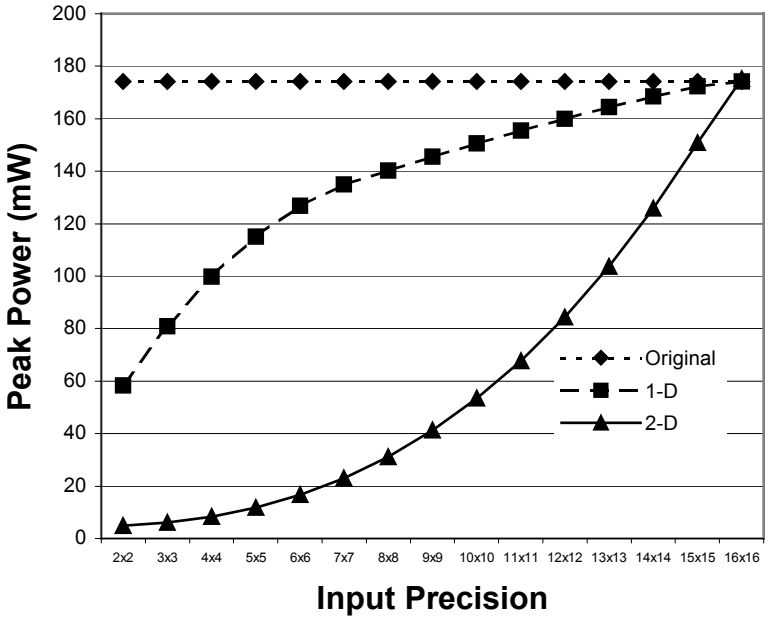
Peak Power (mW)
200
180
160
140
120
100
80
60
40
20
0
2x2
3x3
4x4
5x5
6x6
7x7
8x8
9x9
10x10
11x11
12x12
13x13
14x14
15x15
16x16
Input Precision
Original
1-D
2-D

Fig. 17

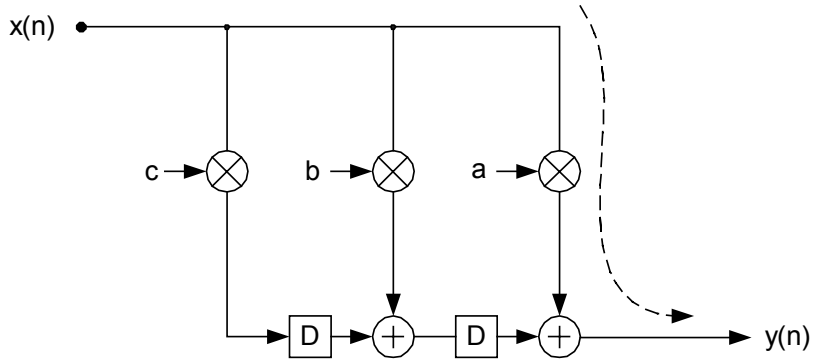
x(n)
c
b
a
D
D
y(n)

Fig. 18

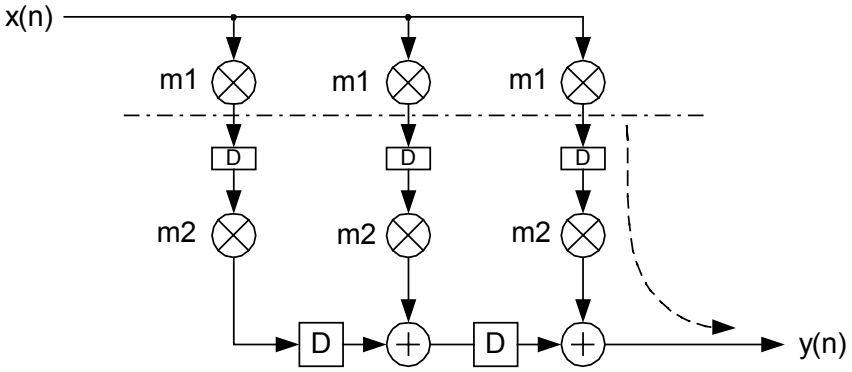
x(n)
m1
m1
m1
D
D
D
m2
m2
m2
D
D
y(n)

Fig. 19

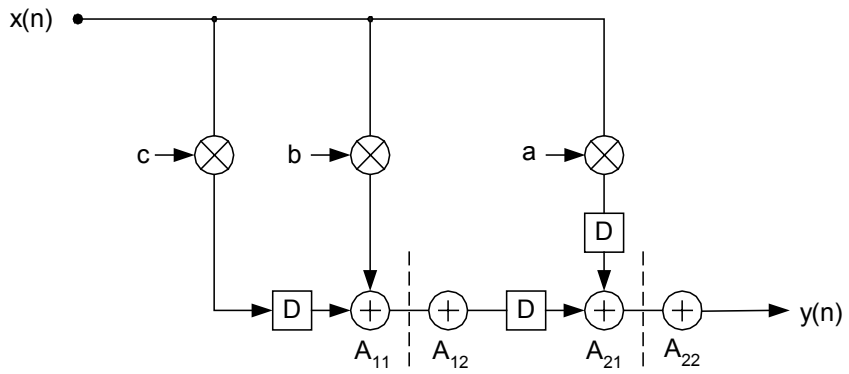

Fig. 20

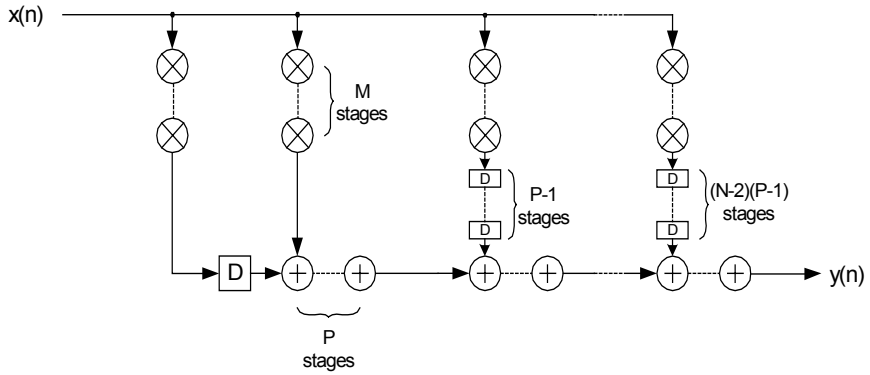

Fig. 21

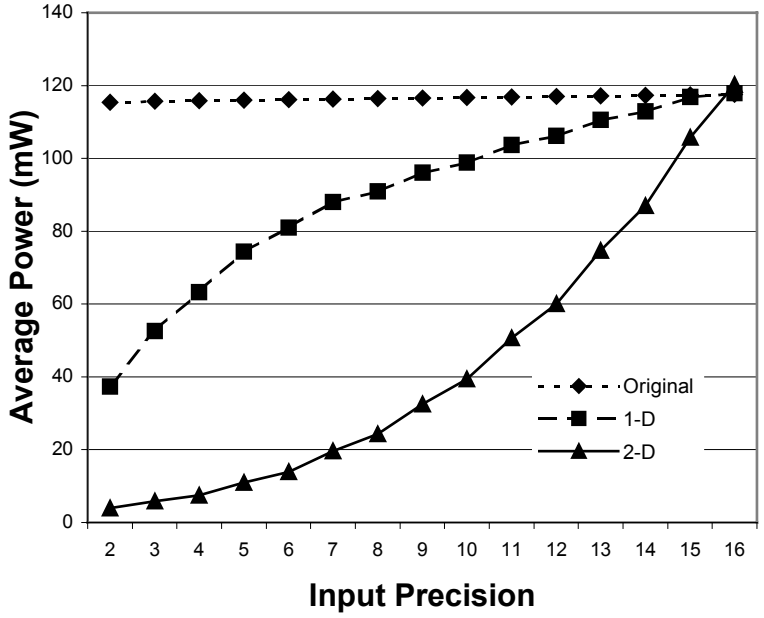
Average Power (mW)
140
120
100
80
60
40
20
0
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input Precision
Original
1-D
2-D

Fig. 22

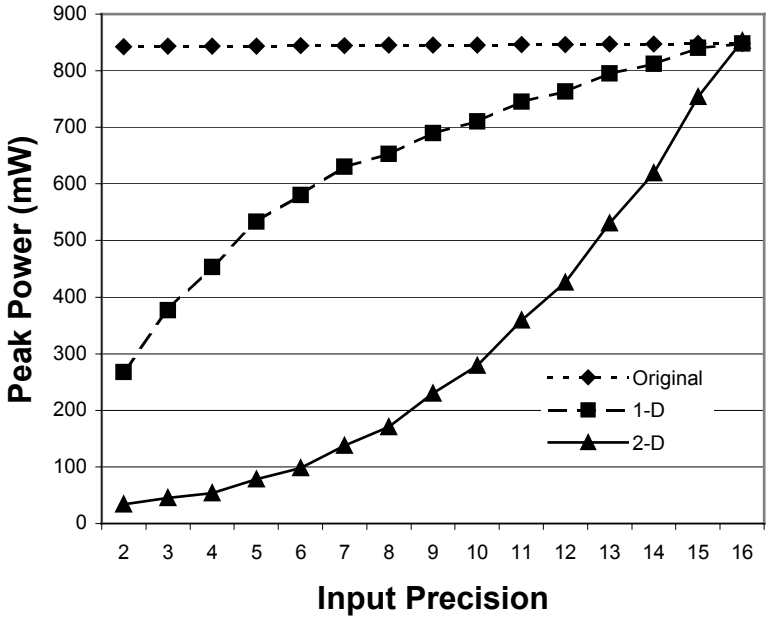
Peak Power (mW)
900
800
700
600
500
400
300
200
100
0
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input Precision
Original
1-D
2-D

Fig. 23

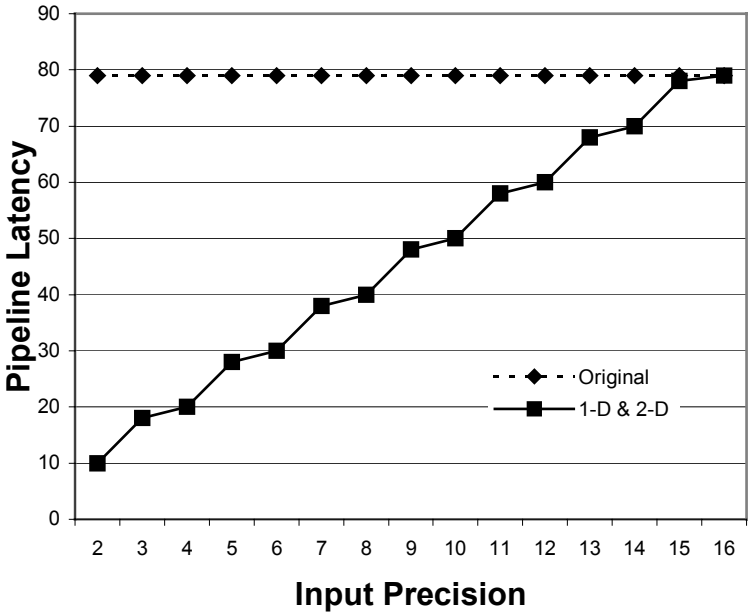
Pipeline Latency
90
80
70
60
50
40
30
20
10
0
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Input Precision
Original
1-D & 2-D

Fig. 24

Multiplier length		4-bit	8-bit	16-bit
Under equal input precision probability	Average power saving			
	1-D vs. Original	29.5%	27.8%	25.7%
	2-D vs. Original	39.6%	57.8%	66.2%
	2-D vs. 1D	14.3%	41.5%	54.4%
	Peak power saving			
	1-D vs. Original	31.0%	28.7%	26.1%
	2-D vs. Original	43.4%	60.0%	67.3%
	2-D vs. 1D	18.0%	43.9%	55.8%
	Latency reduction			
	1-D vs. Original	36.1%	39.1%	44.1%
	2-D vs. Original	36.1%	39.1%	44.1%
Overhead	1-D	0.01%	0.02%	0.03%
	2-D	0.01%	0.02%	0.03%

Table 1

Number of pipeline stages		8	16	32
Under equal input precision probability	Average power saving			
	1-D vs. Original	20.8%	21.0%	21.1%
	2-D vs. Original	62.5%	64.0%	65.0%
	2-D vs. 1D	52.6%	54.5%	55.6%

Table 2

Multiplier length		4-bit	8-bit	16-bit
Under equal input precision probability	Average power saving			
	1-D vs. Original	16.0%	19.4%	21.1%
	2-D vs. Original	26.5%	55.3%	65.0%
	2-D vs. 1D	12.5%	44.6%	55.6%
	Peak power saving			
	1-D vs. Original	17.5%	20.6%	21.7%
	2-D vs. Original	28.4%	55.6%	65.4%
	2-D vs. 1D	13.1%	44.0%	55.8%
	Latency reduction			
	1-D vs. Original	36.1%	39.1%	44.1%
	2-D vs. Original	36.1%	39.1%	44.1%
Overhead	1-D	0.15%	0.03%	0.01%
	2-D	0.52%	0.10%	0.23%

Table 3