High-Performance Left-to-Right Array Multiplier Design

Zhijun Huang and Miloš D. Ercegovac
Computer Science Department
University of California Los Angeles
Los Angeles, CA 90095
{*zjhuang, milos*}*@cs.ucla.edu*

Abstract

We propose a split array multiplier organized in a left-to-right leapfrog (LRLF) structure with reduced delay compared to conventional array multipliers. Moreover, the proposed design shows equivalent performance as tree multipliers for $n \leq 32$. An efficient radix-4 recoding logic generates the partial products in a left-to-right order. The partial products are split into upper and lower groups. Each group is reduced using [3:2] adders with optimized signal flows and the carry-save results from two groups are combined using a [4:2] adder. The final product is obtained with a prefix adder optimized to match the non-uniform arrival profile of the inputs. Layout experiments indicate that upper/lower split multipliers have slightly less area and power than optimized tree multipliers while keeping the same delay for $n \leq 32$.

1. Introduction

The three steps of parallel multiplication are denoted as recoding and partial product (PP) generation (PPG), PP reduction (PPR), and final carry-propagate addition (CPA). Based on the approaches to PPR, multipliers are usually classified into: (i) linear array multipliers with logic delay proportional to n, and (ii) tree multipliers with delay proportional to $log(n)$ [12]. The tree reduction treats PP bits either in rows or in columns. Although tree multipliers have the shortest logic delay in the PPR step, they have irregular layout with complicated interconnects. On the other hand, array multipliers have larger delay but offer regular layout and simpler interconnects. As interconnects become important in deep submicron design [22], architectures with regular layout and simple interconnects are desirable. Irregular layouts with complicated interconnects not only demand more physical design effort but also introduce significant interconnect delay and make noise a problem due to several types of wiring capacitance [1, 22].

Modern multiplier designs use [4:2] adders [14] to reduce the PPR logic delay and regularize the layout. To improve regularity and compact layout, regularly structured tree (RST) with recurring blocks [6] and rectangular-styled tree by folding [8] were proposed, at the expense of more complicated interconnects. In [15], three dimensional minimization (TDM) algorithm was developed to design adders of the maximal possible size with optimized signal connections, which further shortened the PPR path by $1 \sim 2$ XOR delays. However, the resulting structure has more complex layout than a [4:2]-adder based tree. In [10], multiplication was divided recursively into smaller multiplications to increase layout regularity and scalability, which essentially resulted in a hierarchical tree structure.

In linear array multiplier design, the even/odd split structure [9] was proposed to reduce both delay and power of conventional right-to-left (R-L) linear array structures. In [13], a *leapfrog* structure was proposed to take advantage of the delay imbalances in adders. in [4], a left-to-right (L-R) carry-free (LRCF) array multiplier was proposed where the final CPA step to produce the MS bits of the product was avoided by using on-the-fly conversion in parallel with the linear reduction. In [2], this LRCF approach was extended to produce $2n$-bit product. It was also discovered that glitches in L-R reduction arrays were smaller than in the conventional R-L arrays, especially for data with large dynamic range [5, 20, 7].

To further reduce the delay of array multipliers while maintaining their regular layout and simple interconnect, this paper proposes split array LRLF (SALRLF) multipliers that combine the advantages of splitting, L-R computation, and leapfrog structure. Two types of splitting are considered: even/odd and upper/lower. Each step of SALRLF is optimized with the primary objective of delay reduction and the sec-

ondary objective of power reduction. Logic-level analysis as well as physical layout with guided floorplanning are conducted to compare SALRLF with tree multipliers.

In the following, the multiplicand $X = -x_{n-1}2^{n-1} + \sum_{j=0}^{n-2} x_j 2^j$ and the multiplier $Y = -y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i$ are integers in the two's-complement form with n being even to simplify description. For logic-level analysis, the delay of a 2-input XOR2 gate, T_{XOR2}, is used as the unit delay. The delay of two-level a complex gate such as AOI22 (AND2-NOR2) is equivalent to T_{XOR2}.

2. Partial Product Generation

Radix-4 recoding is used to reduce the number of PPs to half. After comparing common recoders, we developed a version *neg/two/one-nf* ("nf" for *neg-first*) shown in Fig. 1. The negation operation is done before the selection between $1X$ and $2X$ so that two_i and one_i set PP_i to zero regardless of neg_i for "-0". To generate additional '1' for negative PP_i, a correction bit $c_i = y_{2i+1}(y_{2i}y_{2i-1})'$ is used.

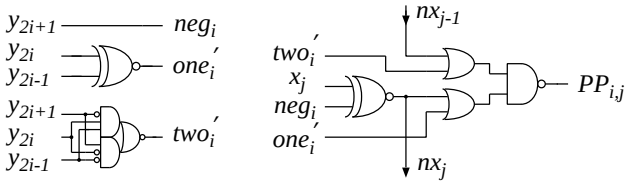


Figure 1: *neg/two/one-nf* generator.

Due to shifting, each PP has a 0 between $PP_{i+1,0}$ and c_i. To have a more regular LSB part of each PP, $PP_{i,0}$ is added with c_i bit in advance [18]. The $PP_{i,0}^{(new)}$ and $c_i^{(new)}$ are described as:

$$PP_{i,0}^{(new)} = x_0 \cdot (y_{2i} \oplus y_{2i-1}) \quad (1)$$

$$c_i^{(new)} = y_{2i+1}y'_{2i}y'_{2i-1} + y_{2i+1}x'_0(y_{2i} \oplus y_{2i-1}) \quad (2)$$

Both $c_i^{(new)}$ and $P_{i,0}^{(new)}$ are obtained no later than other PP bits. The generated PP bit-array is arranged in MSB-first or L-R manner as shown in Fig. 2. The grey circles are $PP_{i,0}^{(new)}$ and the white circles are $c_i^{(new)}$.

3. Partial Product Reduction

The delay gap between tree multipliers and array multipliers is mainly due to the linear PPR structure in conventional array multipliers. To improve the speed of array multipliers, parallelism is introduced in PPR. In addition, different adder types and the signal flow between them also have impact on delay, area, and power.

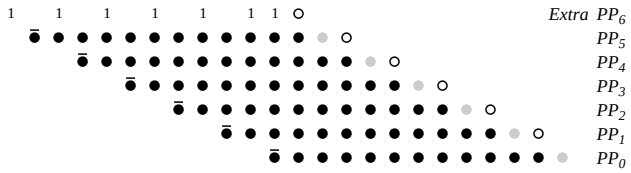


Figure 2: MSB-first radix-4 PP bit array (n=12).

3.1. L-R leapfrog (LRLF) structure

To exploit the delay difference between carry and sum signals in adders, the sum signals in the *leapfrog* [13] structure for R-L array multipliers skip over alternate rows. Because all the carry signals propagate through the entire array, the MSBs of final PPR vectors arrive at the same (latest) time. This unfortunately prevents optimization of the final CPA that is possible in tree multipliers. To allow final CPA optimization in linear array multipliers, we combine L-R computation and leapfrog structure resulting in a new L-R leapfrog (LRLF) array multiplier scheme. A LRLF multiplier for PP array of Fig. 2 is shown in Fig. 3. The dashed lines are carries and solid lines are sum signals. Each adder symbol represents either a FA if all three inputs are variables or a HA if one of three inputs is constant.

The power and delay characteristics of LRLF multiplier have been reported in [7]. Here we optimize the [4:2] adder design according to input arrival profiles. The basic [4:2] adder module, M42, is shown in Fig. 4. The delay from any input A, B, C, and D to output Sum or $Cout$ is $3T_{XOR2}$. Each M42 actually has five inputs because there is one intermediate signal Tin. In the $(n-3)$-bit [4:2] adder for LRLF in Fig. 3, more than half 5-input M42s have one or more zero inputs, which can be simplified to have smaller delay. For M42s with one zero input, the simplification is as follows. Assume the four non-zero inputs of a simplified M42 are A, B, C, and D with arrival time (α) relation $\alpha_A \le \alpha_B \le \alpha_C \le \alpha_D$. The order is arbitrary and does not affect the discussion here because all inputs are functionally equivalent. According to input arrival profiles, two designs with different Sum logic are developed: M42L (linear-Sum) in Fig. 5a and M42T (tree-Sum) in 5b. The arrival times of $Tout$ and

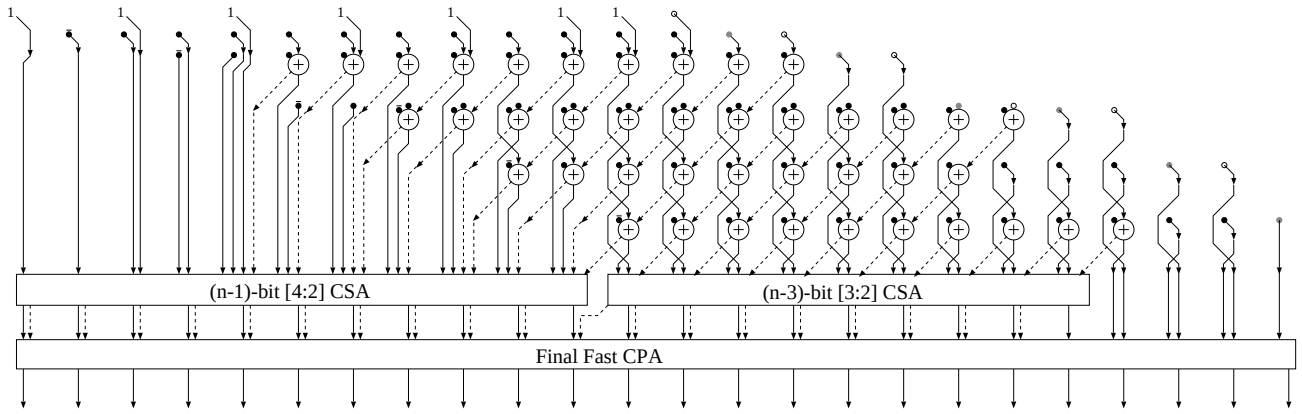


Figure 3: A LRLF array multiplier (n=12).

Cout are

$$\alpha_{Tout} = \alpha_B + T_{AND2} \quad (3)$$

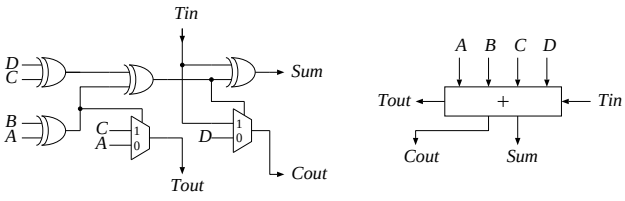
$$\alpha_{Cout} = max(\alpha_B + T_{XOR2}, \alpha_D) + T_{AO222} \quad (4)$$

which are smaller than those in M42. In M42L, Sum arrives at

$$\alpha_{linear-Sum} = max(\alpha_{BCX}, \alpha_D) + T_{XOR2} \quad (5)$$

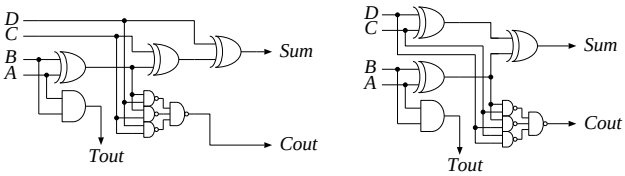
where $\alpha_{BCX} = max(\alpha_B + 2T_{XOR2}, \alpha_C + T_{XOR2})$. In M42T,

$$\alpha_{tree-Sum} = \alpha_D + 2T_{XOR2} \quad (6)$$



(a) MUX-based Design (b) Symbol

Figure 4: Basic [4:2] adder module M42.



(a) M42L: linear-*Sum* (b) M42T: tree-*Sum*

Figure 5: Two simplified M42 designs.

The $(n-1)$-bit [4:2] adder in LRLF is designed from LSB to MSB as follows. For each bit, sort five inputs A, B, C, D, and E (one of them being Tin) according to arrival time and assume that $\alpha_E \leq \alpha_A \leq \alpha_B \leq \alpha_C \leq \alpha_D$. If no input is 0, M42 of Fig. 4 is used. To minimize delay, five inputs E, A, B, C, and D are connected to pin A, B, C, D, and Tin in that order. If two or more inputs are 0s, the adder is reduced to be a FA or HA. The signal flow optimization of FAs will be addressed in Section 3.3. If only one input is 0, it must be E because a constant arrives at the earliest time. Simplified 4-input M42 is used in this case. Four inputs A, B, C, and D are connected to pin A, B, C, and D in that order. M42L is used for faster Sum if $\alpha_{linear-Sum} < \alpha_{tree-Sum}$. Otherwise, M42T is used. The output $Tout$ is fed into the next bit position and the process is repeated. With this optimization, many outputs of the [4:2] adder become available one T_{XOR2} earlier.

3.2. Split array LRLF (SALRLF) structure

The PPR delay of an LRLF array multiplier is about $\lceil \frac{n}{2} \rceil T_{XOR2}$ while that of an $n \times n$-bit radix-4 tree multiplier is $3(\lceil \log_2(\frac{n}{4}) \rceil) T_{XOR2}$. The delay of LRLF is not comparable with that of tree multipliers when $n > 16$. To reduce PPR delay, certain level of parallelism is necessary.

One approach is to split the PP bit array into even PPs and odd PPs, as shown in Fig. 6. In each split part, PPs are shifted four bits each row and reduced into two vectors using a LRLF structure. The final vectors from even and odd parts are merged by a $(2n-3)$-bit [4:2] adder. This algorithm is named even/odd LRLF (EOLRLF).

Another approach is to split the PP bit array into upper PPs and lower PPs, as shown in Fig. 7. In each part, PPs are shifted two bits each row and reduced into two vectors using LRLF. The final vectors from upper and lower parts are merged by a [4:2] adder. To

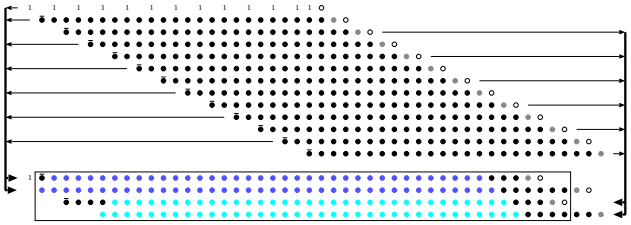


Figure 6: EOLRLF array multiplier (n=24).

reduce the size of this adder, the highest carry bit from the right-side [3:2] CSA in LRLF is fed into the left-side [4:2] CSA as Tin instead of being a bit of the final CPA input. In this way, only a $(n+2)$-bit [4:2] adder is required. This algorithm is named upper/lower LRLF (ULLRLF).

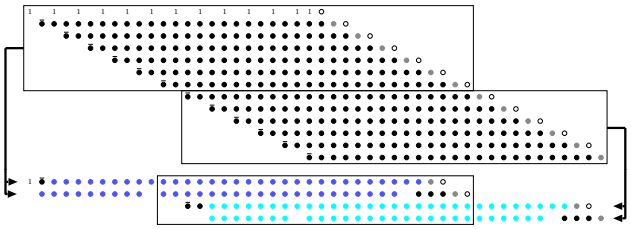


Figure 7: ULLRLF array multiplier (n=24).

The PPR delay of SALRLF is about $(\lceil \frac{n}{4} + 3 \rceil \sim \lceil \frac{n}{4} + 4 \rceil) T_{XOR2}$, depending on the type of adders used. For $n \leq 32$, the delay is $< 11 \sim 12$ while the best result of a tree multiplier is ≤ 9. Further splitting of the PP array reduces the layout regularity and will not be considered. Instead, optimization of FAs and final CPA as well as floorplanning will be used to narrow the remaining gap. In EOLRLF, the arrival profile of PPR final vectors has fewer latest-arriving bits than that in tree multipliers. Fig. 8 shows the PPGR delay profiles in a 32×32-bit TDM multiplier, an EOLRLF, and a ULLRLF. The number of latest-arriving bits in EOLRLF is 5 while this number is 8 in TDM. The bit delay distribution in EOLRLF is also more regular. Most bit groups in EOLRLF have 4-5 bits. But the group size varies a lot in TDM. The final adder design could exploit these better-shaped arrival profiles in EOLRLF to reduce delay. Compared with EOLRLF, ULLRLF has two main advantages. First, the shifting distance between PPs in each upper/lower part is 2 positions instead of 4, which leads to simpler interconnects. Second, the final [4:2] adder in ULLRLF is only $(n+2)$-bit in contrast to $(2n-3)$-bit in EOLRLF. On the other hand, URLRLF has a worse arrival profile than EOLRLF. However, such a profile only leads to just one T_{AO21} delay, which will be explained in Section 4. Our detailed layout experiments indicate that EOLRLF is worse than URLRLF in all measurements. Therefore, we choose ULLRLF in the following discussion.

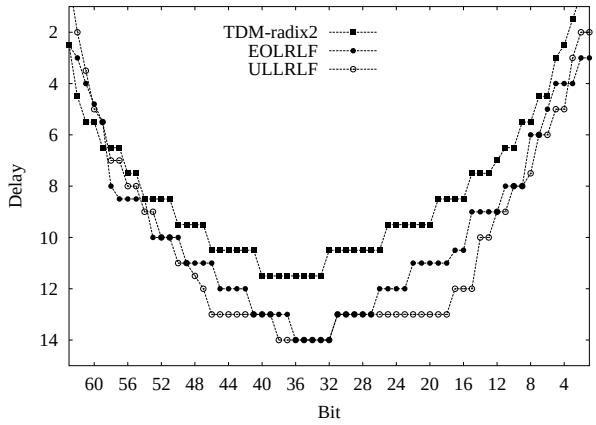
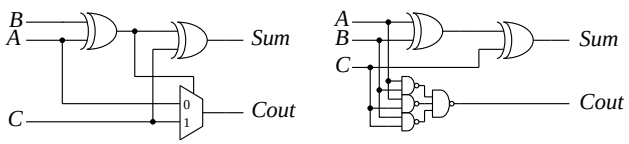


Figure 8: PPGR delay profiles (n=32).

3.3. Optimization of FAs

In array multipliers, the basic components for PPR are full adders (FA). Two common FA structures, FA-MUX and FA-ND3, are shown in Fig. 9. Compared with FA-ND3, FA-MUX typically has smaller area even if pass transistors are not used. Since FA is the most used element in array multipliers, smaller FA would lead to smaller overall area, which is also helpful in the reduction of power consumption and interconnect delay. As to logic delay, however, FA-NAND3 is better than FA-MUX because the delay from all inputs to $Cout$ is T_{AO222} ($T_{AO222} \approx T_{XOR2}$).



(a) FA-MUX (b) FA-ND3

Figure 9: Two FA structures.

Because of the different characteristics of FA inputs, it is possible to optimize signal flow with respect to propagation delay. This technique has been applied in TDM tree multipliers [15]. In addition to delay, signal flow optimization affects power [7]. Assume the three input signals to FA are Ain, Sin, Cin. These input signals are sorted according to their arrival times. We assume that the α relationship is $\alpha_{Ain} \leq \alpha_{Cin} \leq \alpha_{Sin}$. The order is arbitrary since the inputs are functionally equivalent. In FA-ND3, Sin is connected to pin

C. There is no restriction on the connections between $Ain(Bin)$ and pin $A(B)$ unless transistor-level difference between A and B is considered. In FA-MUX, Sin is also connected to pin C. Between Ain and Bin, the signal with less switching activity is connected to pin A for power saving because pin B has less load capacitance and is used for the one with higher switching activity. Since PP bits arrive at the earliest time and never change after PPG, they are connected to A pins. This signal flow optimization technique is named *CSSC* to reflect the interchange of *sum* and *carry* signals. In the experiments section, we show the delay effects of FA selection and CSSC optimization in LRLF and ULLRLF array multipliers.

4. Final Adder

Final adders are optimized to match the non-uniform input arrival profiles. The optimal final adder for tree multipliers is CSMA based design [16]. Efficient design of on-the-fly converter for L-R array multipliers also corresponds to a multi-level carry-select (CSEL) or conditional-sum (CSUM) adders [11]. In [19], generalized earliest-first (GEF) algorithm was proposed to design CSUM for arbitrary input arrival profile. The similarity between CSUM and prefix adder (PFA) is also shown in [19] where PFA is called CLA.

We followed the GEF algorithm and chose PFA for final addition because the PFA operators, AO21 and AND2, are simpler than the basic CSUM operators – a pair of MUX21. Two lists, $Plist$ and $Tlist$, are maintained in GEF. All (G, A) signal pairs are initially put into $Plist$ and sorted according to arrival times. The earliest pairs are then moved to $Tlist$. Adjacent bit pairs in $Tlist$ are retrieved and merged from left to right. The merged pairs are put back into $Plist$. The iteration continues until the generation of the MSB carry bit. Other carry bits are generated using existing (G, A) bits. A PFA example for a hill-shaped arrival profile is shown in Fig. 10. Black nodes in PFA are computation cells and white nodes have no logic or only buffers. In the original GEF, the merging is conducted from from right to left. Because of different input-output delays in operator '$\bullet$', the left-to-right merging in $Tlist$ leads to $0.5T_{XOR2}$ delay improvement.

Let W_{max} be the largest number of adjacent signals that arrive at the same time. If these W_{max} signals are also the latest arriving signals in a hill-shaped arrival profile, the delay of PFA for such a profile can be estimated as

$$T_{PFA} = (\log_2(W_{max}) + 2)T_{AO21} + T_{XOR2} \quad (7)$$

which is not directly related to the adder width $2n$. A small W_{max} would lead to a small T_{PFA}. However, the difference in T_{PFA} is just one T_{AO21} for most schemes in our study because of the logarithmic relationship. One T_{AO21} delay could be further eliminated from T_{PFA} if carry-select adders are used for the final stages of the left part in hill-shaped arrival profiles [3].

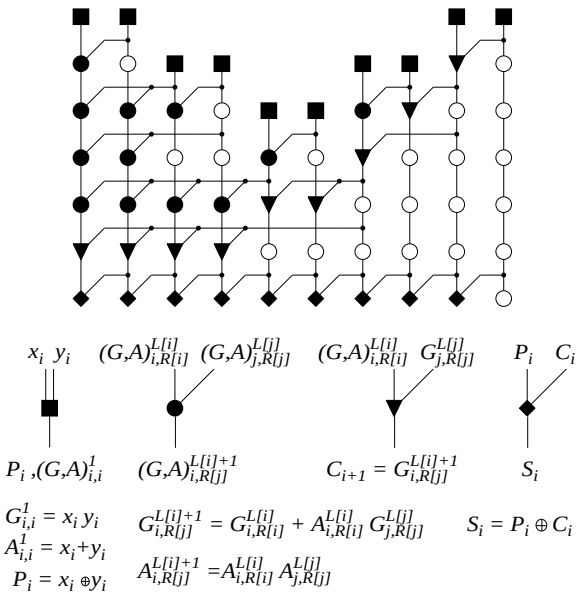


Figure 10: A PFA example.

5. Experiments

To compare the proposed ULLRLF with tree multipliers, logic-level delay analysis is first conducted. Actual VHDL implementation and physical layout are then performed on Synopsys and Cadence design platforms.

5.1. Delay comparison at logic level

VHDL generation programs for both LRLF and ULLRLF algorithms have been written with the flexibility of FA selection and signal flow optimization. The comparison results at logic level without wiring effects are normalized to T_{XOR2} and listed in Table 1. T_{GR} is the delay of PPG and PPR. T_A is the delay of the final adder. For LRLF, the use of FA-ND3 rather than FA-MUX reduces PPR delay and the overall delay by 1 T_{XOR2}. CSSC reduces the delay by 1 T_{XOR2} except for 48-bit LRLF-ND3 where the reduction is 2. For ULLRLF, FA-ND3 reduces one T_{XOR2} in PPR, but not the overall delay. CSSC only reduces PPR delay in ULLRLF-MUX by 0.5 and also has no effect

on the overall delay. This is because varying FAs and applying CSSC change the input arrival profiles of the final adder and affect T_A by up to 1 T_{XOR2}. Even if there is little delay advantage, however, it is still useful to apply CSSC for power reduction [7]. We have also noticed that CSSC and FA-ND3 could help EOLRLF achieve $0.5 \sim 2$ less logic delay than ULLRLF. However, ULLRLF outperforms EOLRLF after layout because of smaller area and simpler wiring. Finally, it is worthwhile to note that T_A does have little relation with the adder width as explained in Eq. 7.

Using the results from Table 1, we now compare the delays of LRLF/ULLRLF with tree multipliers. Radix-2 and radix-4 TDM schemes [15][18] are chosen because they are the best tree multipliers to our knowledge. In addition, tree multipliers based on [4:2] and [3:2] CSAs are also used for comparison as they have more regular structures. To avoid the delay due to the extra row $PP[n/2]$ in radix-4 two's-complement multipliers, the reduction of 9 PPs from $PP[n/2-8]$ to $PP[n/2]$ is based on a [9:4] adder with only $3T_{XOR2}$ delay, as illustrated in Fig. 11. The $3T_{XOR2}$ delay is achieved as follows. All FAs except the right most one in the shaded [3:2] CSA are simplified into HAs with half delay as they have constant inputs. Inputs of the second-level [3:2] adders are properly optimized so that each FA has one input arriving at least T_{XOR2} later than the other two inputs. This late input is connected to pin C of FAs to ensure one T_{XOR2} delay. To distinguish from other tree multipliers, the radix-4 tree multiplier using this special [9:4] adder is named *tree*9*to*4.

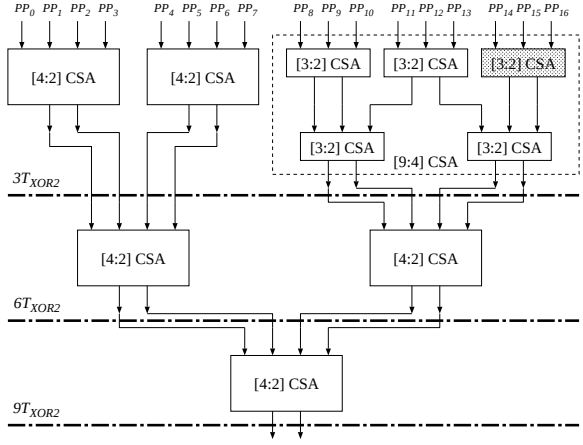


Figure 11: Tree PPR with $9T_{XOR2}$ delay.

The logic delay comparison results are given in Table 2. The blank boxes with '–' are because T_{GR}s or delay profiles from PPR are not available from literature. The original TDM-radix4 data in [18] are normalized to our measurement base. It is shown that the radix-4 tree multipliers based on our [9:4] adder design have almost the same T_{PPGR} as TDM schemes. For $n \leq 32$, ULLRLFs have $0.5 \sim 1.5T_{XOR2}$ more delay than tree multipliers. For larger precisions, ULLRLF shows 23% more gate delay for 48×48-bit multiplication and 28% more for 54×54-bit multiplication.

5.2. Simulation with physical layout

For more realistic evaluation, structural VHDL designs are compiled and mapped into Artisan TSMC $0.18\mu m$ 1.8-Volt standard-cell library [23] using Synopsys Design Compiler. For a fair comparison of different schemes, [3:2] FA cells in the library are not used because there is no [4:2] adder cells. Buffers are inserted automatically by Design Compiler. Two radix-4 schemes for 24×24-bit and 32×32-bit multiplication are compared: tree9to4 and ULLRLF-MUX-CSSC. tree9to4 has the similar delay as TDM but is more regular. MUX-CSSC based designs are chosen because it has smaller area and CSSC is good for power. CSSC is also applied in tree9to4. Standard-cell based automatic layout is first conducted using Cadence Silicon Ensemble. Interconnect parameters are extracted from layout and back-annotated into Synopsys tools for delay and power calculation. Power consumption is measured at 100MHz with 500 pseudo-random data. The results are shown in Table 3. For 24-bit, URLRLF is better than tree9to4 in area, delay, and power, with up to 7% improvement. For 32-bit, URLRLF has 3% less area and 10% less power than tree9to4 while keeping similar delay.

Table 3: Comparison after automatic layout

Schemes	Area(μm^2)	Delay(ns)	Pwr(mW)
tree9to4-24b	56,126	6.01	26.27
ULLRLF-24b	54,014	5.88	24.59
tree9to4-32b	108,324	7.18	45.12
ULLRLF-32b	105,380	7.25	40.65

We have also experimented layout with guided floorplanning for 32×32-bit multipliers. The floorplan of tree9to4 is shown in Fig. 12, which is based on H-tree for symmetry and regularity [17]. The row utilization rate has to be relaxed to 63% from 70% in automatic layout for routability. In addition, all blocks have to be assigned to specific regions for delay reduction. The floorplan of ULLRLF is shown in Fig. 13. The left-side [4:2] CSA in each part is distributed into PPR rows. The row utilization remains at 70%. Regions are assigned for two big upper/lower blocks, final

Table 1: The delay effects of FA type and CSSC in LRLF/ULLRLF

Schemes	$n = 24$			$n = 32$			$n = 48$		
	T_{GR}	T_A	Total	T_{GR}	T_A	Total	T_{GR}	T_A	Total
LRLF-MUX	15	5	20	19	5	24	27	5	32
LRLF-MUX-CSSC	14	5	19	18	5	23	26	5	31
LRLF-ND3	14	5	19	18	5	23	26	5	31
LRLF-ND3-CSSC	13	5	18	17	5	22	24.5	4.5	29
ULLRLF-MUX	12	5	17	14	6	20	18	6	24
ULLRLF-MUX-CSSC	11.5	5.5	17	13.5	6.5	20	17.5	6.5	24
ULLRLF-ND3	11	6	17	13	6	19	17	7	24
ULLRLF-ND3-CSSC	11	6	17	13	6	19	17	7	24

Table 2: Delay comparison of tree multipliers and LRLF/ULLRLF

Schemes	$n = 24$			$n = 32$			$n = 48$			$n = 54$		
	T_{GR}	T_A	Total	T_{GR}	T_A	Total	T_{GR}	T_A	Total	T_{GR}	T_A	Total
TDM-radix2	10.5	–	–	11.5	6	17.5	13.5	6	19.5	13.5	6	19.5
TDM-radix4	11	5.5	16.5	12	6	18	–	–	–	–	–	–
Tree9to4	10	6	16	11	7	18	13	7	20	14	7	21
LRLF	13	5	18	17	5	22	24.5	4.5	29	26.5	4.5	31
ULLRLF	11	6	17	13	6	19	17	7	24	18	7	25

[4:2] adder, and CPA. The results are shown in Table 4. The delay is improved by 4% from automatic layout. For ULLRLF, there is no cost in area for this delay improvement. For tree9to4, the area increases 10%. After layout with guided floorplanning, ULLRLF and tree9to4 has similar delay while tree9to4 has 15% more area and 9% more power.

Table 4: Comparison after guided layout

Schemes	Area(μm^2)	Delay(ns)	Pwr(mW)
tree9to4-32b	120,958	6.90	45.53
ULLRLF-32b	105,380	6.99	41.72

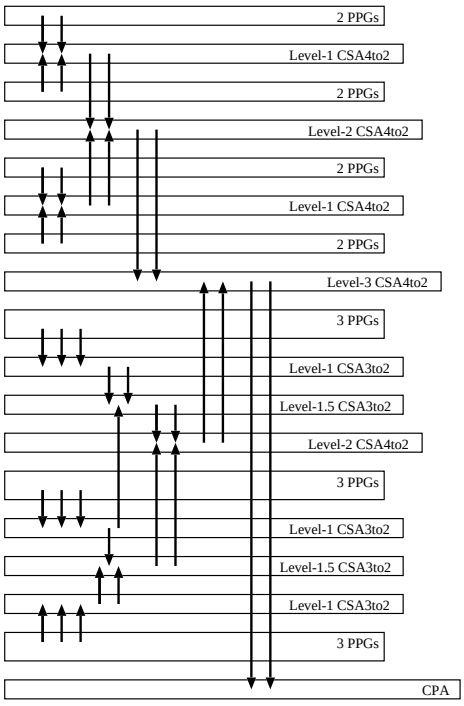


Figure 12: Floorplan of tree9to4 (n=32).

6. Conclusions

We have studied left-to-right split array multiplier schemes EOLRLF/ULLRLF. An efficient radix-4 recoding logic generates the partial products in a left-to-right order. These partial products are split into upper/lower or even/odd groups. These two groups are reduced in parallel using the L-R leapfrog structure with optimized adder modules and signal flows. Results from the two groups are merged using a [4:2] adder. The final adder is a prefix adder optimized to match non-uniform input arrival profile. We find that upper/lower splitting outperforms even/odd splitting after layout although even/odd splitting is a little better in gate delay at logic level. Layout experiments for $n = 24$ and $n = 32$ indicate that ULLRLF multipliers have slightly less area and power than optimized tree multipliers while keeping similar delay. We con-

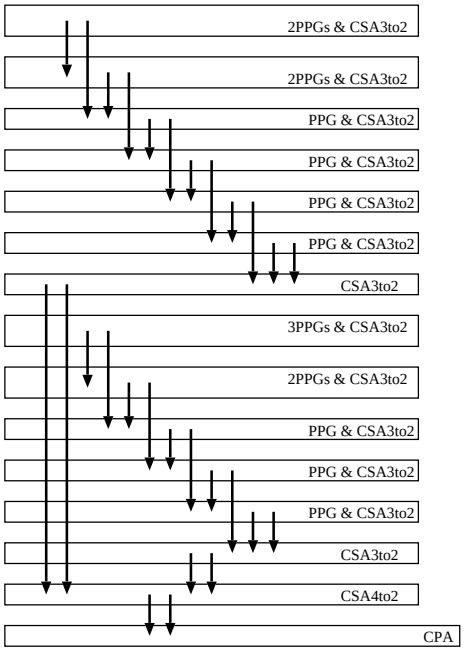


Figure 13: Floorplan of ULLRLF (n=32).

clude that ULLRLF array multipliers and tree multipliers are similar in major performance characteristics for $n \leq 32$ if standard-cell based automatic layout is conducted.

Acknowledgments

The authors would like to thank Dr. Wen-Chang Yeh and Dr. Paul Stelling for their help. This work has been supported in part by Raytheon Company, Fujitsu Laboratories of America, and the State of California MICRO program.

References

[1] K.C. Bickerstaff, E.E. Swartzlander, Jr., and M.J. Schulte, "Analysis of column compression multipliers," in *Proc. 15th IEEE Symp. Computer Arithmetic*, pp.33-29, 2001.

[2] L. Ciminiera and P. Montuschi, "Carry-save Multiplication Schemes Without Final Addition," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 1050-1055, Sept. 1996.

[3] Y. Choi and Earl E. Swartzlander, Jr., "Design of a hybrid prefix adder for non-uniform input arrival times," in *Proc. SPIE 2002 Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, July 2002.

[4] M.D. Ercegovac and T. Lang, "Fast multiplication without carry-propagate addition," *IEEE Trans. Comput.*, vol.39, no.11, pp.1385-1390, Nov. 1990.

[5] A. Goldovsky, *et. al.*, "Design and implementation of a 16 by 16 low-power two's complement multiplier," in *Proc. 2000 IEEE Int. Symp. Circuits and Systems*, vol.5, pp.345-348, 2000.

[6] G. Goto, *et. al.*, "A 54*54-b regularly structured tree multiplier," *IEEE J. Solid-State Circuits*, vol.27, no.9, pp.1229-1236, Sept. 1992.

[7] Z. Huang and M.D. Ercegovac, "Low power array multiplier design by topology optimization," in *Proc. SPIE 2002 Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, July 2002.

[8] N. Itoh, *et. al.*, "A 600-MHz 54*54-bit multiplier with rectangular-styled Wallace tree," *IEEE J. Solid-State Circuits*, vol.36, no.2, p.249-257, Feb. 2001.

[9] J. Iwamura, *et. al.*, "A high speed and low power CMOS/SOS multiplier-accumulator," *Microelectronics Journal*, vol.14, no.6, pp.49-57, Nov.-Dec. 1983.

[10] J. Kim and E.E. Swartzlander, Jr., "Improving the recursive multiplier," in *Proc. 34th Asilomar Conf. Signals, Systems and Computers*, pp.1320-1324, Nov. 2000.

[11] R.K. Kolagotla, H.R. Srinivas, and G.F. Burns, "VLSI implementation of a 200-MHz 16*16 left-to-right carry-free multiplier in 0.35 mu m CMOS technology for next-generation DSPs," in *Proc. IEEE 1997 Custom Integrated Circuits Conf.*, pp.469-472, May 1997.

[12] I. Koren, *Computer Arithmetic Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[13] S.S. Mahant-Shetti, P.T. Balsara, and C. Lemonds, "High performance low power array multiplier using temporal tiling," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol.7, no.1, p.121-124, March 1999.

[14] M. Nagamatsu, *et. al.*, "A 15-ns 32*32-b CMOS multiplier with an improved parallel structure," *IEEE J. Solid-State Circuits*, vol.25, pp.494-497, Apr. 1990.

[15] V.G. Oklobdzija, D. Villeger, and S.S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Trans. Comput.*, vol.45, no.3, pp.294-306, March 1996.

[16] P.F. Stelling and V.G. Oklobdzija, "Designing optimal hybrid final adders in a parallel multiplier using conditional sum blocks," in *Proc. 15th IMACS World Congress Scientific Computation, Modeling, and Applied Math.*, Aug. 1997.

[17] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, Inc., 1983.

[18] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. Comput.*, vol.49, no.7, pp.692-701, July 2000.

[19] W.-C. Yeh, *Arithmetic Module Design and its Application to FFT*. Ph.D. dissertation, National Chiao-Tung University, 2001.

[20] Z. Yu, L. Wasserman, and A.N. Willson, Jr. "A painless way to reduce power dissipation by over 18% in Booth-encoded carry-save array multipliers for DSP," in *2000 IEEE Workshop on SiGNAL PROCESSING SYSTEMS*, pp.571-580, Oct. 2000.

[21] R. Zimmermann, *Binary Adder Architectures for Cell-Based VLSI and their Synthesis*. Ph.D. dissertation, Swiss Federal Institute of Technology, Zurich, 1997.

[22] *International Technology Roadmap for Semiconductors – Interconnect*, 2001 Edition.

[23] *TSMC* 0.18μm *Process 1.8-Volt SAGE-X Standard Cell Library Databook*. Artisan Components, Inc., Oct. 2001.