Volume : 3 | Issue : 5 | May 2014 • ISSN No 2277 - 8179

Fpga Implementation of Variable-Latency Speculating Booth Multiplier (VLSBM)

Engineering

KEYWORDS : Adaptive carry estimation, error compensation, FPGA, speculating multiplier.

Bency John

PG Scholar, Department of Electronics and Communication, VVIT, Calicut University, Kerala, India.

ABSTRACT

Data hazards cause major pipeline performance degradation for data-intensive computing processes. To improve the performance of the pipeline efficiency, a high-speed VLSBM is proposed. This is done by successively performing a speculating and correcting phase. To decrease the critical path, the VLSBM partial products are divided into the (n-z)-bit least significant part (LSP) and the (n+z)-bit most significant part (MSP). The estimation function predicts the carry to the MSP, thereby permitting independent calculation of the partial-product accumulation of parts. If a carry prediction is true, the data dependence is hidden and the correcting phase is bypassed, thereby make sure that the potential speed-up of the pipelined data path. When a miss-carry prediction occurs, the speculation is flushed and the correcting phase is performed to obtain the exact multiplication. Verilog HDL is used as a description language and Xilinx ISE sim as simulation tool for implementation. Xilinx Spartan 3 XC3S200 Field Programmable Gate Array (FPGA) was chosen as a Hardware Platform for the System Implementation.

I. Introduction

Multipliers are key components of various computing and digital signal processing applications [1]. The latency for this parallel multiplication is normally divided into three blocks. Primary block is the generation of partial products. Second block is the Wallace tree section, which is a reduction-tree process that simplifies partial products into two rows and the last block is the addition of the result by a carry-propagation adder (CPA) [2]–[6]. Because all partial product bits within each column are added in parallel, the Wallace tree compression is superior during the second block. To get the reduced critical path, the parallel multiplier CPA, a carry-look ahead adder (CLA) has been frequently applied.

Pipelining is a good method to alleviate the drawback from aggressive logic- and gate-level timing optimizations while maintaining sufficient throughput. If it has poor pipeline efficiency, the throughput benefit of the pipelined data path is eventually undermined by data hazards [1]. So it is possible to hide data dependence by employing a speculative functional unit (SFU). The SFU is an arithmetic unit that gives a predictor for the carry signal [7]. This SFU predicts the carry of one or more cells in the design without waiting for the true carry propagation. Conversely, when inaccurate carry predictions occur, error detection logic and the associated error recovery are needed to generate to get the real result.

To improve the addition reliability, a variable-latency speculating adder (VLSA) is proposed to done the integrating error detection logic into the almost correct adder (ACA). This achieved a critical path almost same as that of a conventional CLA, and an error recovery process (i.e., the original CLA) that reproduced the true addition. The ACA is faster than the VLSA, but it is less reliable. So, they are both potentially unsuitable for the fast and reliable parallel multiplier CPA.

Based on a probabilistic result-speculation technique, another design method is proposed, i.e. the design for an efficient variable-latency speculating multiplier (VLSM). Fig. 1 shows the block diagram of the VLSM design. This multiplier concept consists of two operating phases. First one is the speculating phase and next is the correcting phase. To lessen the critical path, the multiplier partial products are split into two parts. 1) the -bit most significant parts (MSP) and 2) the bit least significant parts (LSP). During the speculating phase, the partial-product accumulations of the LSP and the MSP are executed in parallel and independently. The carry estimation function is generated a carry to the -bit MSP, while without waiting the true carry from the LSP.

So, the speculating output of the VLSM is

$$\textstyle\sum_{n+z}(MSP) + 2^{n-z}\sigma_z + \sum_{n-z}(LSP) \qquad \ldots\ldots (1)$$

which can be given to the successive instruction for further processing. Here in (1) is the partial-product addition of the –bit MSP and is the addition of the -bit LSP minus the carry. If the carry is true, then the corresponding correcting phase is not considered and the data dependence is hidden; thus, the multiplier cycle is saved. If an inaccurate carry prediction occurs, the speculation is flushed and the exact multiplication is obtained by adding, the true carry, and during the correcting phase. Compared to the VLSA, the error recovery of the proposed VLSM is simple; thus, correcting the miss-carry-prediction does not subside the overall performance of the VLSM.

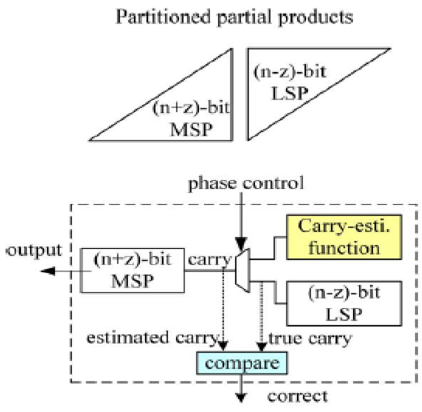


Fig. 1: Proposed variable-length speculating multiplier (VLSM)

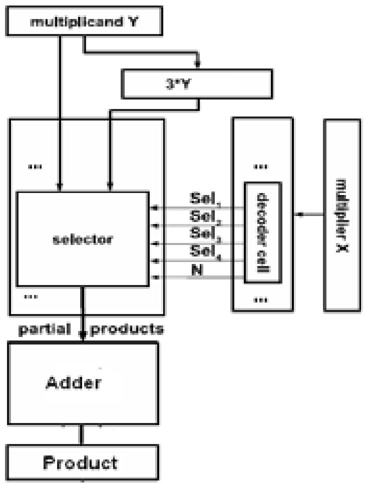


Fig 2: Booth 3 multiplier.

A generator that creates a smaller number of partial products will allow the partial product summation to be faster and use less hardware. These circuits include a Partial-Product-Generator (PPG) and adders. A recoding scheme introduced by Booth reduces the number of partial products by about a factor of two. The multiplier is partitioned into overlapping groups of 4 bits, and each group is decoded to select a single partial product as per the selection table. Each partial product could be from the set {±0, ±M, ±2M, ±3M, ±4M}. All multiples with the exception of 3M are easily obtained by simple shifting and complementing of the multiplicand.

TABLE I PARTIAL PRODUCT SELECTION TABLE

Partial Product Selection Table			
Multiplier Bits	Selection	Multiplier Bits	Selection
0000	+ 0	1000	-4 x Multiplicand
0001	+ Multiplicand	1001	-3 x Multiplicand
0010	+ Multiplicand	1010	-3 x Multiplicand
0011	+2 x Multiplicand	1011	-2 x Multiplicand
0100	+2 x Multiplicand	1100	-2 x Multiplicand
0101	+3 x Multiplicand	1101	-Multiplicand
0110	+3 x Multiplicand	1110	-Multiplicand
0111	+4 x Multiplicand	1111	-0

The remaining section of this paper is organized as follows. Based on conditional probability theory, Section II briefly introduces carry estimation schemes for the fixed-width Baugh-Wooley multiplier[20] and the modified Booth multipliers. Based on the generalized carry estimation schemes, Section III details the optimized VLSBM design. Section IV presents simulation results. Lastly section V presents conclusion.

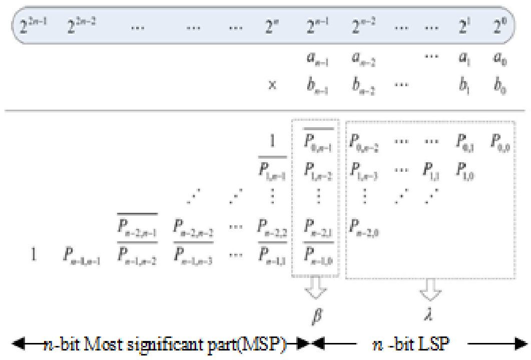
II GENERALIZED CARRY ESTIMATIONS FOR -PT FIXED-WIDTH MULTIPLIERS

The Baugh-Wooley multiplication algorithm is applied frequently to design regular multipliers by handling sign bits of the 2's-complement integers efficiently. The two integers A and B are given,

$$A = -a_{n-1}2^{n-1} + \sum_{j=0}^{n-2} a_j 2^j$$

$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \quad \text{.... (2)}$$

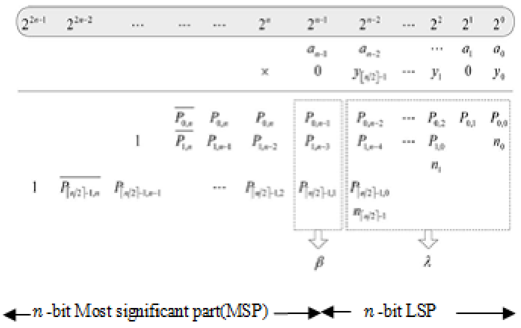
Where ,є



2^{2n-1} 2^{2n-2} ... 2^n 2^{n-1} 2^{n-2} ... 2^1 2^0
a_{n-1} a_{n-2} ... a_1 a_0
× b_{n-1} b_{n-2} ... b_1 b_0
β
λ
n-bit Most significant part(MSP)
n -bit LSP

Fig. 3. Partial products of -bit A B Baugh-Wolley multiplier

Fig. 3 shows the partial products of Baugh-Wooley multiplier, where ≡,and is the complement of . Note that the -bit MSP and the -bit LSP divide the partial products into two groups. β and α (Fig. 3) are defined by the additions of the most significant column and the remaining -bit LSP columns, respectively. For getting more idea, this study defines =and =then β=.



2^{2n-1} 2^{2n-2} ... 2^n 2^{n-1} 2^{n-2} ... 2^2 2^1 2^0
β
λ
n -bit Most significant part(MSP)
n -bit LSP

Fig. 4. Partial products of -bit A B Modified Booth multiplier

To reduce the critical path by subsiding the number of rows of partial products, the Booth's multiplication algorithm is developed. Table I shows the partial product selection table for booth-3 encoder. Fig. 4 shows the reduced partial products of the modified Booth multiplier. Therefore, the results of the direct-truncation (DT), post-truncation (PT), or full-precision (FP) multipliers are expressed as

$$(A \times B)_{DT} = \sum_n (MSP)$$

$$\begin{aligned}(A \times B)_{PT} &= \sum_n (MSP) + [2^{n-1}\beta + \lambda]_r \\ &= \sum_n (MSP) + 2^n[\frac{\beta}{2} + 2^{-n}\lambda]_r \\ &= \sum_n (MSP) + 2^n\sigma\end{aligned}$$

$$(A \times B)_{FP} = \textstyle\sum_n (MSP) + 2^n\sigma + \sum_n (LSP) \quad \text{.... (3)}$$

where σ is the carry-in to and operator rounds to the next integer. Without disseminating the true σ from LSP to MSP, the DT multiplier is fast and economical, although it causes a significant loss in precision.

Fig. 5 shows the -bit fixed-width -PT multiplier, which defines 1 ≤ ≤ and is for designing an efficient VLSM. By definition, the -PT multiplier adds the most significant columns of the partial products, and subsequently rounds or truncates to the -bit result. α, β and λ (Fig. 5) are the additions of the most significant columns, the -th most significant column, and the remaining columns of the -bit LSP, respectively.

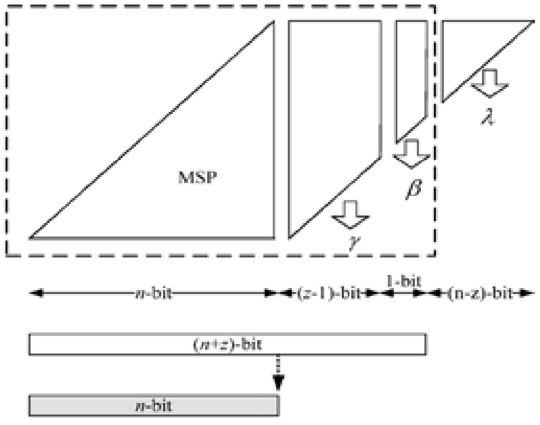


Fig. 5 The -bit fixed-width -PT multiplier

With α, β and λ the -bit -PT multiplication can be expressed as

$$(A \times B)_{z-PT} = \textstyle\sum_n (MSP) + [\gamma + 2^{n-1}\beta + \lambda]_r$$

$$= \textstyle\sum_n (MSP) + 2^n[2^{-n}\gamma + \sigma_z]_r \quad \text{.... (4)}$$

Volume : 3 | Issue : 5 | May 2014 • ISSN No 2277 - 8179

Where denotes the carry to for the -PT multiplier.

III. VARIABLE-LATENCY SPECULATING BOOTH 3 MULTIPLIER

Fig. 6 shows the comprehensive micro architecture of the proposed 16 16 variable-latency speculating two-stage pipelined modified Booth 3 multiplier (VLSBM), which comprises the following two operating stages: 1) Stage-1 is the speculating phase of the VLSBM and 2) Stage-2 performs the error recovery. To decrease the critical path, the VLSBM partial products are divided independently into the -bit MSP and the -bit LSP. As shown in Fig. 7, the “” is applied for the differentiation to signify the speculation, and the “” represents the-bit correct multiplication of the VLSBM. Without waiting the from the-bit LSP, and can be calculated in parallel during the speculating phase with the estimated λ ().

1) Error Detection and Error Recovery: The error recovery of “” is separated by the following two parts in the proposed VLSBM: 1) the primary -bit compensation for and 2) the successive -bit compensation for . As shown in Fig. 6, the error correction of each part is performed in stages to decrease the critical path.

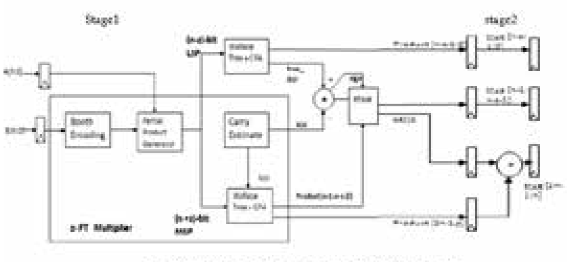


Fig 6. The proposed speculating 2-stage pipelined modified Booth 3 multiplier

Because is always true, at the end of the Stage1 the result is . It is readily proven that can be obtained by subtracting the estimated λ() from true λ(z), and then adding it to . Consequently, to verify the estimation, an adder and subsequent subtractor are applied (Fig. 6). The sign of the difference (true λ() estimated λ()) is stored by the “sign” bit for further use. The following process is based on the true value of the sign bit.

(a) If sign=0 :

This means that an under-estimated (or correct) has been predicted. If an estimation is inaccurate, (true λ()λ()) is the compensation that must be added to speculative for error correction. After the first -bit compensation for the in the speculating phase, the carry of the error-compensation adder is recoded (Fig. 6). The following procedure is based on the truth value of the recorded carry bit.

(i) If carry=0 : This means that no carry is required for further compensation of during the correcting phase. Therefore, the “correct” bit is set to 1 and .

(ii) If carry=1: This means that a non-zero carry should be added to during the correcting phase. Therefore, the “correct” bit is set to 0.

(b) If sign=1:

This means that for λ(z), an over-estimated prediction has been considered, which leads to (true λ(z)λ(z)) < 0. The difference must be calculated, and then subsequently subtracted from during the compensation stage error correction. At the end of Stage1, the carry of the error-correction adder is recoded to calculate the following procedure.

(i) If carry=0 : This means that no carry is required for further correction of during the correcting phase. Therefore, the “correct” bit is set to 1 and .

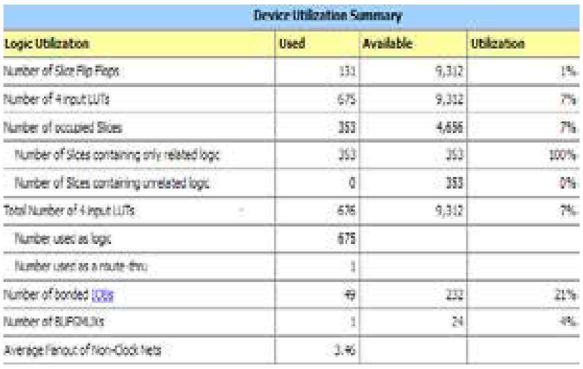
(ii) If carry=1 : This means that a non-zero carry must be added to during the correcting phase. Therefore, the “correct” bit is set to 0.

IV SIMULATION RESULTS

The behavior of the design is described in Verilog. The Verilog code is simulated using Xilinx ISE sim Simulation Tool. Xilinx Synthesis Technology (XST) tool is selected for synthesis.

Device utilization:

TABLE II DEVICE UTILIZATION OF SPARTAN XC3S500E



Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	131	9,312	1%
Number of 4 input LUTs	675	9,312	7%
Number of occupied Slices	353	4,656	7%
Number of Slices containing only related logic	353	353	100%
Number of Slices containing unrelated logic	0	353	0%
Total Number of 4 input LUTs	676	9,312	7%
Number used as logic	675		
Number used as a route-thru	1		
Number of bonded IOBs	49	232	21%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	3.46		

For example, the inputs are taken as 25 and 34, then the simulation results is shown in Fig 7. The actual carry generated is “0”, but the estimated carry is “1”.So as discussed the difference is calculated and the difference is subtracted in the correction phase.

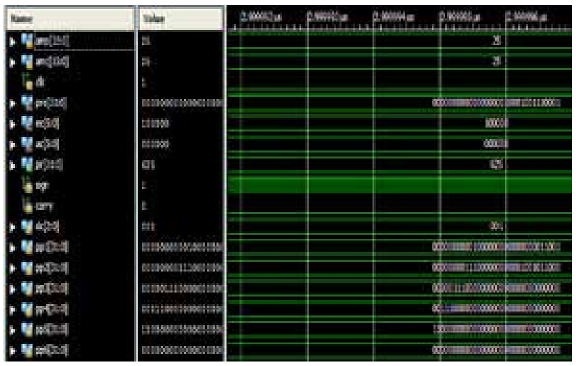


Fig 7.simulation result

V CONCLUSION

Data hazards cause major performance degradation in the pipelined data path of data-intensive computing processes. To improve the performance of the pipeline efficiency, this study implemented an optimized VLSBM. Without waiting for the true carry propagation, the VLSBM employs a conditional- probability carry estimator for the carry signal. Similar to the branch predictor used in microprocessors, if a true carry prediction occurs, the multiplier-stall cycles caused by data dependence can be hidden.

The future direction of this research will consider additional number of pipeline stages. Devising an efficient control and error recovery mechanism for compensating for miss carry predictions without adversely impacting the overall performance are the most critical challenges.

REFERENCE

[1] K. K. Parhi, VLSI Digital Signal Processing Systems. NewYork,NY, | USA: Wiley, 1999. | [2] J. Fadavi-Ardekani, " Booth encoded multiplier generator using optimized | Wallace trees," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 1, | no. 2, pp. 120–125, Jun. 1993. | [3] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier | design," IEEE Trans. Computers, vol. 49, no. 7, pp. 692–701, Jul. 2000. | [4] Q. Li, G. Liang, and A. Bermak, "A high-speed 32-bit signed/unsigned | pipelined multiplier," in IEEE Int. Symp. Electronic Design, Test & | Applications, DELTA, 2010, pp. 207–211. | [5] R. K. Yu and G. B. Zyner, "167 MHz radix-4 floating point multiplier," in | Proc. 12th Symp. Computer Arithmetic, 1995, pp. 149–154. | [6] G. Even and P.-M. Seidel, "A comparison of three rounding algorithms for | IEEE floating-point multiplication," IEEE Trans. Computers, vol. 49, no | 7, pp. 638–650, Jul. 2000. | [7] A. A. Del Barrio, S. O. Memik, M. C. Molina, J.M.Mendias, and R. | Hermida, "A distributed controller for managing speculative functional | units in high level synthesis," IEEE Trans. Comput.-Aided Des. Integr. | Circuits Syst., vol. 30, no. 3, pp. 350–363, March 2011. | [8] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative | addition: A new paradigm for arithmetic circuit design," in Proc. Des., | Automat. Test Eur., 2008, pp. 1250–1255. | [9] A. Cilardo, "A new speculative addition architecture suitable for two's | complement operation," in Proc. Des., Automat. Test Eur., 2009, pp. 664– | 669. | [10] Y. C. Lim, "Single-precision multiplier with reduced circuit complexity | for signal processing applications," IEEE Trans. Computers, vol. 41, no. | 10, pp. 1333–1336, Oct. 1992. | [11] M. J. Shulte and E. E. Swartzlander, Jr., "Truncated multiplication with | correction constant," VLSI Signal Processing, vol. VI, pp. 388–396, | 1993. | [12] S. S. Kidambi, F. El-Guibaly, and A. Antonious, "Area-efficient | multipliers for digital signal processing applications," IEEE Trans. | Circuits Syst. II: Analog and Digital Signal Processing, vol. 43, no. 2, | pp. 90–95, Feb. 1996. | [13] J.-M. Jou, S.-R. Kuang, and R.-D. Chen, "Design of low-error fixedwidth | multipliers for DSP applications," IEEE Trans. Circuits Syst. II: Analog | and Digital Signal Processing, vol. 46, no. 6, pp. 836–842, Jun. 1999. | [14] L.-D. Van, S.-S. Wang, and W.-S. Feng, "Design of the lower error | fixed-width multiplier and its application," IEEE Trans. Circuits Syst. II: | Analog and Digital Signal Processing, vol. 47, no. 10, pp. 1112–1118, | Oct. 2000. | [15] S.-J. Jou, M.-H. Tsai, and Y.-L. Tsao, "Low-error reduced-width Booth | multipliers for DSP applications," IEEE Trans. Circuits, Syst I, vol. 50, | no. 11, pp. 1470–1474, Nov. 2003. | [16] K. J. Cho,K. C. Lee, J. G. Chung, andK.K. Parhi, "Design of low-error | fixed-width modified Booth multiplier," IEEE Trans. Very Large Scale | Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004. | [17] L.-D. Van and C.-C.Yang, "Generalized low-error area-efficient fixed | width multiplies," IEEE Trans. Circuits Systems I: Regular Papers, vol. | 52, no. 8, pp. 1608–1619, Aug. 2005. | [18] T. B. Juang and S. F. Hsiao, "Low-error carry-free fixed-width | multipliers with low-cost compensation circuits," IEEE Trans. Circuits | Syst. II, vol. 52, no. 6, pp. 299–303, Jun. 2005. | [19] A. G. M. Strollo, N. Petra, and D. De Caro, "Dual-tree error | compensation for high performance fixed-width multipliers," IEEE Trans. | Circuits Syst. II, vol. 52, no. 8, pp. 501–507, Aug. 2005. | [20] Y.-C. Liao, H.-C. Chang, and C.-W. Liu, "Carry estimation for two's | complement fixed-width multipliers," in Workshop on Signal Processing | Systems (SiPS), Banff, Canada, Oct. 2006, pp. 345–350. | [21] S.-R. Kuang and J.-P. Wang, "Low-error configurable truncated | multipliers for multiply-accumulate applications," Electron. Lett., vol. 42, | no. 16, pp. 904–905, Aug. 3, 2006. | [22] H.-A. Huang, Y.-C. Liao, and H.-C. Chang, "A self-compensation Fixed | width Booth multiplier and its 128-point FFT applications," in Proc. | IEEE Int. Symp. Circuits Syst., 2006, pp. 3538–3541. | [23] M. A. Song, L. D. Van, and S. Y. Kuo, "Adaptive low-error fixedwidth | Booth multipliers," IEICE Trans. Fundam., vol. E90-A, no. 6, pp. 1180– | 1187, Jun. 2007. | [24] J.-H. Tu and L.-D. Van, "Power-efficient pipelined reconfigurable fixed- | width Baugh-Wooley multipliers," IEEE Trans. Comput., vol. 58, no. 10, | pp. 1346–1355, Oct. 2009. | [25] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, | "Truncated binary multipliers with variable correction and minimum | mean square error," IEEE Trans. Circuits Syst. I, vol. 57, no. 6, pp. | 1312–1325, Jun. 2010. | [26] C.-H. Chang and R. K. Satzoda, "A low error and high performance | | multiplexer-based truncated multiplier," IEEE Trans. Very Large Scale | Integr. (VLSI) Syst., vol. 18, no. 12, pp. 1767–1771, Dec. 2010. | [27] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, | "Design of fixed-widthmultipliers with linear compensation function," | IEEE Trans. Circuits Syst. I, vol. 58, no. 5, pp. 947–960, May 2011. [28] | C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, "A probabilistic | estimation bias circuit for fixed-width Booth multiplier and its DCT | applications," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 4, | pp. 215–219, Apr. 2011. |