Title	High-speed and low-power serial accumulator for serial/parallel multiplier
Author(s)	Meher, Manas Ranjan; Jong, Ching Chuen; Chang, Chip Hong
Citation	Meher, M. R., Jong, C. C., & Chang, C. H. (2008). High-speed and low-power serial accumulator for serial/parallel multiplier. IEEE Asia Pacific Conference on Circuits and Systems, pp.176-179, Macau, China.
Date	2008
URL	http://hdl.handle.net/10220/6353
Rights	

High-Speed and Low-Power Serial Accumulator for Serial/Parallel Multiplier

Manas Ranjan Meher, Ching-Chuen Jong, Chip-Hong Chang

Centre for Integrated Circuits & Systems (CICS)

School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore

***Abstract*—This paper presents a new approach to serial/parallel multiplier design by using parallel 1's counters to accumulate the binary partial product bits. The 1's in each column of the partial product matrix due to the serially input operands are accumulated using a serial T-flip flop (TFF) counter. Consequently, the column height is reduced from *N* to $\lfloor \log_2 N \rfloor + 1$. This logarithmic reduction results in a very small carry save adder (CSA) array or tree required before the two final summands are added up to obtain the final product. The counters can be clocked at very high frequency (around 1.5 GHz as dictated mainly by the TFF propagation delay) and the accumulation frequency is independent of the operand size. The proposed accumulation method achieves 33%, 38%, 43% gain in speed respectively for 31, 63, 127 operands accumulators and on average 42% reduction in power consumption over CSA based accumulation implemented in 0.18μm CMOS technology.**

I. INTRODUCTION

Accumulation is an integral part of many arithmetic circuits and digital signal processing (DSP) subsystems. The accumulation can be carried out serially or in parallel. Parallel accumulators (multi-operand adder) are popular for their lowest delay but are not suitable in an application where data is fetched serially to the system and the target chip has tight IO, power, area and testability constraints. Serial accumulator is a better choice in such cases due to its low hardware cost, reduced power consumption and reliability [1,3]. Serial accumulators are used beneficially in serial/parallel multipliers to accumulate the partial products. It is also a vital component in inner-product computation and matrix-vector multiplications commonly found in communications, testing, cryptography, etc. [2-6]. Fig. 1 shows the conventional ripple carry adder implementation of a serial accumulator. The critical path is shown in a dotted line. The operating frequency in this case is limited by the ripple carry adder which has a delay of $O(m)$, with m being the width of the operand. For higher frequency of operation, L. Dadda *et al.* proposed a carry save adder (CSA) implementation of pipelined adder as shown in Fig. 2 [7-8]. It has a delay of only one full adder (FA) and one D flip flop (DFF). In this paper we propose a new approach to serial accumulation using 1's counter which has a delay of only one TFF in each cycle of accumulation and dissipates low energy.

This paper is organized as follows. The proposed scheme is discussed in Section II. A counter-based serial/parallel multiplier is presented in Section III and the ASIC implementation results and performance comparison are given in Section IV. Finally, the conclusion is drawn in Section V.

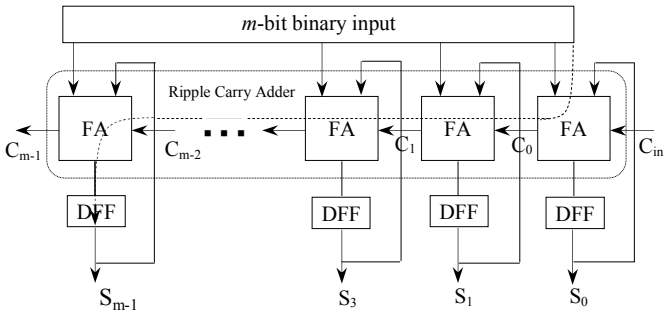

Fig. 1 Carry propagate adder based accumulator

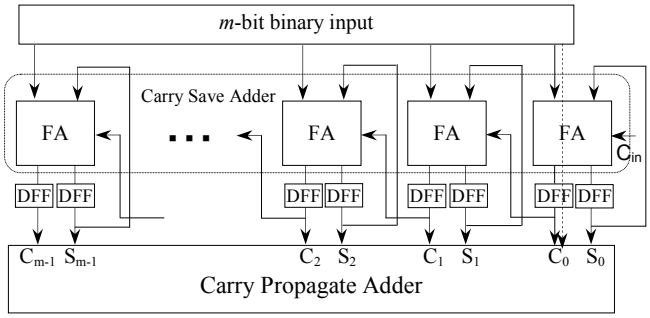

Fig. 2 Carry save adder based accumulator

II. PROPOSED COUNTER-BASED ACCUMULATOR

A typical accumulator is nothing but an adder which successively adds the current input with the value stored in its internal register. Generally, the adder used is either a simple ripple carry adder or a CSA for high-frequency operation. The delay of the former case is limited by the carry propagation chain and the latter requires two n-bit registers for the sum and carry vectors and an additional fast vector merged adder. In either case, the basic functional unit is a FA cell. Here, we suggest a new approach to serial accumulation by using binary ripple counter to reduce the power consumption without compromising the critical path delay.

978-1-4244-2342-2/08/$25.00 ©2008 IEEE.

An accumulation of n integers x_i for $i = 0, 1 \ldots n-1$ can be expressed mathematically as:

$$S = \sum_{i=0}^{n-1} x_i \tag{1}$$

For ease of exposition, let x_i be an unsigned integer represented by m binary weighted bits. (1) can be rewritten as:

$$S = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{m-1} x_i(j) \cdot 2^j \right) \tag{2}$$

where $x_i(j) \in \{0, 1\}$ is the j^{th} bit of the i^{th} operand and is associated with a positional weight 2^j. For each positional weight (0^{th} to m^{th} bit position), there exist a maximum of n binary bits. They can be added up by a simple binary ripple counter of width $k = \lfloor \log_2 n \rfloor + 1$ which can be specially designed to count the number of 1's in a column. Thus, an m-bit accumulator can be realized with m such dedicated binary counters to accumulate the 1's in each bit position in parallel. The *dependency graph* (DG) of such a scheme is shown in Fig. 3 for $m = 8$ and $n = 7$. The nodes in the DG represent a 1's counter. At the end of the n^{th} iteration or cycle each counter produces a k-bit binary weighted output. The least significant bit of the counter output is aligned with the column of bits accumulated by the same counter. By arranging all counter output bits of the same positional weight in the same column, the resulting dot matrix height has reduced logarithmically from n to k. The hardware architecture of an accumulator corresponding to the DG of Fig. 3 is shown in Fig. 4. The bits of the serially input operands are fed into their corresponding counters from column 0 to column 7. These counters execute independently and in parallel. In each cycle of accumulation, when a new operand is loaded, a column counter is incremented if a '1' is present in the bit position corresponding to that column. These counters can be clocked at very high frequency and all the operands will be accumulated to the counter outputs at the end of the n^{th} clock. The final outputs of the counters need to be further reduced to only two rows of partial products by a CSA array or a CSA tree (Wallace or Dadda's tree). A carry propagate adder is then used to obtain the final result. During the accumulation, the vector merging stage is disabled to significantly reduce the spurious switching in the adder network. Additional power minimization stems from the fact that the counter output will not toggle unless a '1' is present at its input. Hence, the probability of switching is reduced by half successively from the least significant bit to the most significant bit of the counter outputs.

The counters C in Fig. 4 are used to count the number of 1's in a column. Each of them is a simple TFF based ripple counter except that the T input of the first TFF is driven by the corresponding bit of the operand. The master clock is provided to the first TFF and all other TFFs are triggered by the preceding TFF outputs like a ripple counter. A typical 3-bit 1's counter is shown in Fig. 5. The clock input of this counter is synchronized with the input data rate and thus the operands can be accumulated with a maximum frequency defined by the setup time and propagation delay of a TFF.

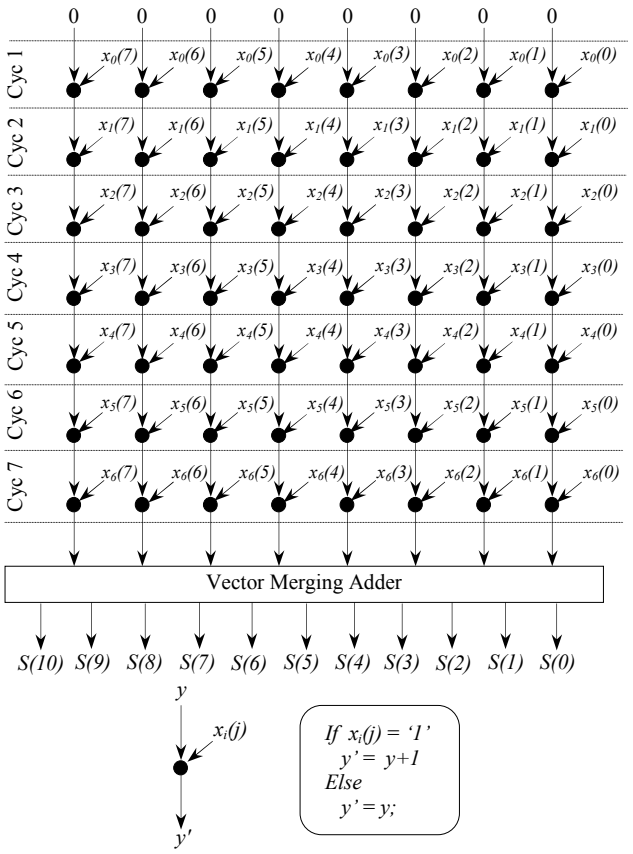

Fig. 3 Dependency graph for the proposed accumulator (7 operands and each operand has 8 bits)

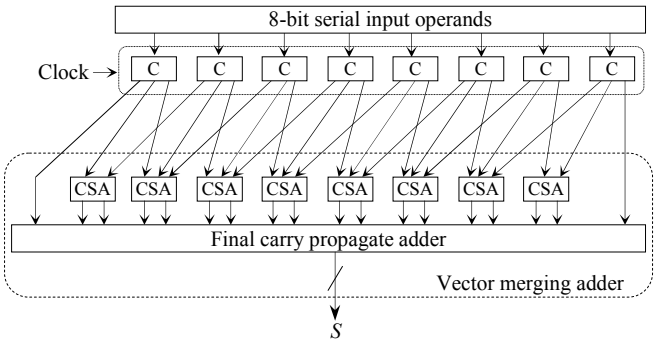

Fig. 4 Proposed counter-based accumulator architecture. (7 operands and each operand has 8 bits)

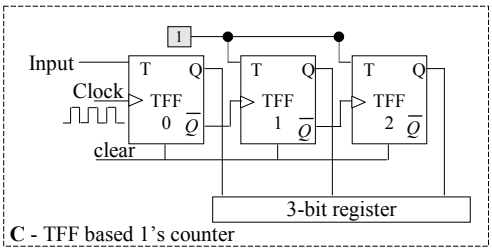

Fig. 5 A 3-bit 1's counter circuit

III. Design of a Fast Serial/Parallel Multiplier Using the Proposed Accumulator

The proposed accumulator can be used to design a fast serial/parallel binary multiplier. Let A = [a_{m-1}, a_{m-2}, ..., a_0] and B = [b_{n-1}, b_{n-2}, ... , b_0] be the two operands of word-length m and n, respectively. In the proposed method, the multiplier (A) bits are logically ANDed with the multiplicand (B) bits serially to produce the partial product matrix. The size of the resultant partial product matrix is $n\times(m+n-1)$, where n is the number of partial product rows due to the n multiplicand bits. To reduce the height of the partial product matrix, column to row transformation is performed by a bank of 1's counters or so called the accumulator. In this case, each counter counts the number of ones in a column of h ($h > 1$) partial product bits serially and outputs a row of partial product bits of length $\lfloor \log_2 h \rfloor + 1$, with the least significant bit aligned with the column. Since the column height of the partial product matrix increments from 1 to n and then reduces to 1, the length of the counters also vary accordingly. A dot matrix diagram of a 16×16-bit multiplication with the proposed counter based partial product reduction scheme is shown in Fig. 6.

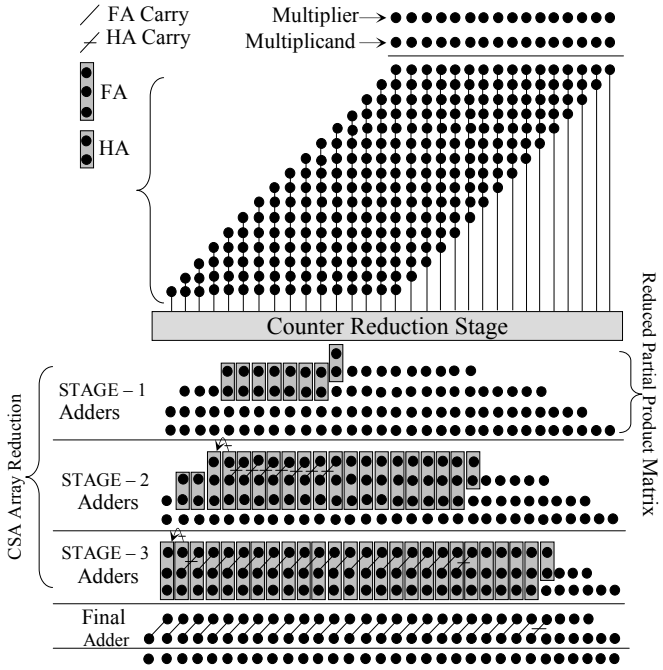

Fig. 6 Dot matrix diagram of a 16×16-bit counter-based multiplier

The detailed hardware architecture of a 7×7 bit multiplier is shown in Fig. 7. It consists of 13 AND gates and a 13-bit serial-in parallel-out shift-register (SIPOSR) to generate the partial products. One input of each AND gate is fed from the SIPOSR and the other input of all the AND gates are commonly fed by the multiplicand bits serially. On the first cycle, the 13-bit SIPOSR is loaded with 6 leading zero bits followed by the 7 multiplier bits. At the same time, the least significant bit of the multiplicand is fed into the common input line of the AND gates. Thereafter, the content of SIPOSR is shifted left by one bit and a zero is padded to its right. In addition, the next multiplicand bit is also fed to the common input of the AND array. The output of each AND gate is connected to a dedicated 1's counter, The size of the 1's counters, C0, C1, ..., C10, are not the same. One row of partial products is generated in each cycle. Thus, 7 cycles are required to count the number of 1's present in every column of partial products. It should be noted that the operating frequency of column-to-row transformation is independent of the operand width but the total latency is defined by the width of the multiplicand. At the beginning of the 7th clock cycle, the counter outputs hold the correct reduced form of the partial product matrix and are ready for further reduction by the parallel CSA array. Until the 7th clock the CSA array and the final carry propagate adder are disabled to avoid spurious transitions in the adder network to conserve power. Also, he switching probability of each counter output bit is reduced by half from $\frac{1}{2}$ for TFF 0 to $\frac{1}{2^n}$ for TFF $n-1$.

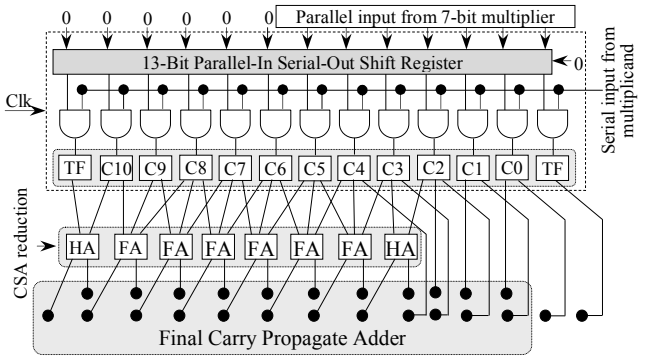

Fig. 7 Hardware Architecture of Proposed 7×7 bit Serial/Parallel Multiplier

IV. Implementation Results & Performance Evaluation

The proposed counter based accumulator is advantageous in speed and power consumption than the CSA based accumulator counterpart at the expense of a slightly higher hardware cost. The ripple carry adder based accumulator as shown in Fig. 1 should have a minimum period of $T_{DF} + mT_{FA}$, where T_{DF} and T_{FA} are the delays in the DFF and FA, respectively. The CSA based accumulator as shown in Fig. 2 is the best existing [8] scheme for serial accumulation with a FA and a FF in its critical path and its period is defined by $T_{DF} + T_{FA}$ if the delay associated with the final adder is excluded. The proposed counter based accumulator possesses a delay of T_{TFF}, where T_{TFF} is the propagation delay of a TFF. Thus, at least one FA delay is saved in each cycle of accumulation which becomes significant as the number of operands to be accumulated increase. The performance of the proposed accumulator is evaluated by implementing it in ASIC using the TSMC 0.18 μm CMOS technology. The top level design was synthesized by Synopsys Design Compiler to obtain the gate level netlist. The pre-layout synthesis results for CSA based and proposed accumulator is listed in Table I. The average dynamic power and energy consumption

were measured by Synopsys Power Compiler by inferring the switching activity information from 10,000 randomly generated operands at a frequency of 200 MHz. With the added net delay and fanout, the proposed accumulator saves around 1 *ns* in each cycle of accumulation compared to a CSA based accumulator. The counter based accumulator completes an accumulation process with around 33%, 38% and 43% less total delay than the CSA counterpart for 31, 63 and 127 operands accumulators respectively. In addition, the proposed accumulation method consumes 42% less power on average than the conventional CSA based accumulator.

TABLE I TOTAL DELAY, POWER AND ENERGY COMPARISON

No. of Operands	CSA Based Accumulation			Counter Based Accumulation		
	Delay (ns)	*Power (mW)*	*Energy (pJ)*	*Delay (ns)*	*Power (mW)*	*Energy (pJ)*
31	98.92	8.85	1633	66.65	5.10	970
63	170.26	8.89	3070	106.05	5.14	1791
127	313.94	8.94	5955	179.65	5.19	3478

Before we evaluate the performance of the proposed serial/parallel multiplier using the counter-based accumulation scheme, it is worth reviewing few of the best existing serial/parallel multipliers in the literature [9-11]. These multipliers are based on carry-save add-and-shift (CSAS) architecture. The clock period of the proposed method can be approximated as:

$$T_C = T_{AND} + T_{TFF} \tag{3}$$

where T_{AND} is the delay of an AND gate and T_{TFF} is the setup time of a T-flip flop. As per [11], the normalized delay of the basic logic elements in terms of the propagation delay of a NAND gate (T_{NAND}) is summarized in Table II. Thus, the total delay of an *n*-bit multiplier is approximated as follows:

$$\begin{aligned} T_{Total} &\approx (T_{AND} + T_{TFF})n + (\log_2 n + 1)\cdot T_{FA} \\ &\approx \{5.3n + (\log_2 n + 1)4.6\}T_{NAND} \end{aligned} \tag{4}$$

TABLE II DELAYS OF BASIC LOGIC MODULES [11]

Element	Delay
Two-input NAND	1.0
Two-input AND	1.5
Inverter	0.5
Two-input XOR	2.3
Flip-Flop	3.8
Full Adder	4.6

The final stage carry propagate adder (CPA) delay is not considered in the above comparison since, CPA can easily be pipelined with the counter stage or a fast parallel carry look ahead adder (CLA) may be used to minimize the delay of this stage. Table III summarizes the throughput rate (minimum clock period) and the total latency of various serial/parallel multipliers. From Tables II and III, it is evident that the proposed serial/parallel multiplication scheme has a higher throughput rate than other best existing methods.

TABLE III TOTAL LATENCY COMPARISONS

Method	Clock Delay	Number of cycles	Multiplication time
CSAS [10]	9.9 T_{NAND}	N+M	$(M+N)9.9T_{NANAD}$
FSP [10]	9.9 T_{NAND}	M	$(9.4M+(N-1)4.6)T_{NAND}$
DS-SPM [11]	8.8 T_{NAND}	(M+N)/2	$(4.4(M+N)+17.6)\ T_{NAND}$
Proposed	5.3 T_{NAND}	M	$(5.3M+(\log_2 M+1)4.6)T_{NAND}$

V. CONCLUSION

A novel approach to serial accumulation of unsigned binary numbers by a bank of high speed 1's counters is proposed in this paper. The pre-layout synthesis result shows that the proposed accumulation scheme is 33%, 38%, 43% faster respectively for 31, 63, 127 operands accumulators and 42% power efficient than its CSA counterpart. This idea has been extended to design a fast serial/parallel multiplier. By using a bank of high speed binary 1's counters, every *h* partial product bits in a column is compacted into only $\lfloor \log_2 h \rfloor + 1$ horizontal bits in a row. Consequently, the height of the partial product matrix is reduced from *N* to $\lfloor \log_2 N \rfloor + 1$. Our experimental results show that the proposed method computes a multiplication in less time than other existing serial/parallel multipliers. Besides the serial/parallel multiplier, the accumulator finds widespread applications in many multimedia signal processing functions where repeated additions are required.

REFERENCES

[1] P.E. Denielsson, "Serial-Parallel Convolvers", *IEEE Trans.* on Computers, pp.652-667, vol.33,1984.

[2] Yong Sin Kim and Sung Mo Kang, "A high Speed Low-Power Accumulator for Direct Digital Frequency Synthesizer", *Proc. IEEE International Symposium on Microwave*, pp. 502-505, June 2006.

[3] Valeriu Beiu, Snorre Aunet, Jabulani Nyathi, Robert R., and Walid Ibrahim, "Serial Addition: Locally Connected Architectures", *IEEE Trans. on Circuits & System-I*, pp.2564-2578.,vol. 54 (11), November 2007,

[4] Alizam Abbasfar, Dariush Divsalar, Kung Yao, "Accumulate-Repeat-Accumulate Codes", *IEEE Trans. on Communication*, pp.692-702, vol.55(4), April 2007.

[5] P. Karpodinis, D. Kagaris, D. Nikolos, "*Accumulator Based Test-Per-Scan BIST*", *Proc. IEEE International Symposium on On-Line Testing* (IOLTS'04), pp.193-198, July 2004.

[6] Sandeep Kumar, Tomas Wollinger and Christof Paar, "Optimum Digit Serial $GF(2^m)$ Multipliers for Curve-based Cryptography", *IEEE Trans. on Computers*, pp.1306-1311, vol.55(10), October 2006.

[7] Luigi Dadda, Vincenzo Piuri, "Pipelined Adders", *IEEE Trans. on Computers*, pp.348-356, vol.45(3), March 1996.

[8] Reto Zimmermann, *Lecture Notes on Computer Arithmetic: Principle, Architecture and VLSI Design*, March 1999.

[9] E.E. Swartzlander, Jr., "Quasi-Serial Multiplier", *IEEE Trans*. on Computers, pp.317-321, vol.22, 1973.

[10] R. Gnanasekaran, "A Fast Serial-Parallel Binary Multiplier", *IEEE Trans*. on Computers, pp.741-744, vol.34, 1985.

[11] H.I. Saleh *et al*, "Novel serial-parallel multipliers", *IEE Proc*. of Circuits, Devices & Systems, pp.183-189, vol.148(4), August 2001.