FAST FPGA-BASED PIPELINED DIGIT-SERIAL/PARALLEL MULTIPLIERS

Javier Valls, Trini Sansaloni, Marcos M. Peiró, and Eduardo Boemo*

Departamento de Ingeniería Electrónica, Universidad Politécnica de Valencia,
Camino de Vera s/n, 46071 Valencia, Spain. E-mail: {jvalls, mpeiro, tmsansal}@eln.upv.es,

* Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid,
Ctra. Colmenar Km.15, 28049 Madrid, Spain. E-mail: eduardo.boemo@ii.uam.es

In this paper fast pipelined digit-serial/parallel multipliers are proposed. The conventional digit-serial/parallel multipliers and their pipelined versions are presented. Every structure has been implemented on FPGA and the results are given. These results have been analysed and it is detected that the pipelined ones do not have the throughput improvement expected because of a logic depth increment. As a consequence, a new structure based on the fast serial/parallel multiplier proposed in [1] has been developed. The new multipliers designed achieve better performance than the previous ones: their throughput is higher than it in the pipelined serial/parallel multipliers with nearly the same cost in area.

1. INTRODUCTION

Real-time signal processing hardware requires efficient multiplier units. However, each application needs different sample rates; from speech to image or radar a wide frequency range is covered. In several cases, it is senseless to use a bit-parallel circuit: it has an important cost in area and runs faster than the throughput required by the application. In these cases, the bit-serial [2] or digit-serial approach became an important alternative to efficiently implement custom DSP circuits. Furthermore, in FPGA-targeted applications, the serial stream of data matches better with the structure of such devices [3].

In digit-serial computation, data words of size W bits are partitioned into digits of size N bits (the digit-size, N, is divisor of the word-size, W) and are processed serially one digit at a time with the least significant digit first. A complete word is processed in P=W/N clock cycles and consecutive words follow each other continuously. The time of P cycles is named a sample period. In every digit-serial operator, it is necessary to add some control signals to indicate, for example, a new word entry. The digit-serial processors include parallel-serial and serial-parallel converters to process in digit format and to present the result in parallel one. The digit-serial operators are cascaded following the data-flow algorithm in a pipeline fashion. A detailed explanation of this kind of architectures can be found in [4], [5], [6], [7], [8], [9]. A set of digit-serial architectures can be designed by using different digit-sizes. Each element of the family has a specific size and throughput. Thus, it is possible to choose the digit-size that best suits the speed of the application, minimising the cost in terms of area.

In this paper a set of fast pipelined serial/parallel multipliers are proposed. All circuits compute the data-coefficient multiplication in two's complement representation. All the topologies have been implemented on an EPF10K50GC403-3 Altera FPGA for W=8 bits size (data and coeficients) using the default compilation options (automatic placement and routing).

This paper is organised in four parts: in Section 2 and 3 the digit-serial/parallel multipliers and the pipelined digit-serial/parallel ones are respectively explained, and the results of each one implementation on FPGA are given. In next section, a brief discussion of those results is made. In Section 5, more efficient pipelined digit-serial/parallel multipliers are proposed and the results of their implementation on FPGA are also given. Finally, the conclusions are presented.

2. SERIAL/PARALLEL MULTIPLIER

Serial/parallel multipliers use the shift-and-add algorithm to compute the multiplication of a coefficient, which is given in parallel form, by a data, which enters in the multiplier in serial style. Assuming coefficient, *C*, with W_c bits, and data, *X*, with W_d bits, are two's complement numbers, the equation of the serial/parallel multiplier can be written as:

$$Y = C \cdot X = C \cdot \left(-x_0 + \sum_{i=1}^{W_c-1} x_i 2^{-i} \right) = -C \cdot x_0 + \sum_{i=1}^{W_d-1} (C \cdot x_i) \cdot 2^{-i} \qquad [1]$$

the term $-C.x_0$ can be developed as:

$$-C \cdot x_0 = -\overline{c_0} \cdot x_0 + \sum_{i=1}^{W_c-1} \overline{c_i} \cdot x_0 \cdot 2^{-i} + x_0 \cdot 2^{-W_c+1} \qquad [2]$$

corresponding to the computational scheme shown in Fig. 1 for the case of both data and coefficient size of W=4 bits.

			c_0	c_1	c_2	c_3
		x	x_0	x_1	x_2	x_3
x_3c_0	x_3c_0	x_3c_0	x_3c_0	x_3c_1	x_3c_2	x_3c_3
x_2c_0	x_2c_0	x_2c_0	x_2c_1	x_2c_2	x_2c_3	
x_1c_0	x_1c_0	x_1c_1	x_1c_2	x_1c_3		
$x_0\overline{c_0}$	$x_0\overline{c_1}$	$x_0\overline{c_2}$	$x_0\overline{c_3}$			
			$+x_0$			
y_0	y_1	y_2	y_3	y_4	y_5	y_6

Fig. 1: Computational scheme

A bit-serial multiplier that maps this computational scheme is depicted in Fig. 2. In the first W_d clock cycles, the data-bits enter serially into the multiplier with the least significant bit (LSB) first. The low order product bits are computed and serially outputted with LSB first. When the last bit of the data (the sign bit) enters, the control signal SIGN is high, the coefficient is complemented to one and the sign bit, x_3, is latched to be fed, in the next cycle, into the final bit-serial adder. In this way, the two's complement is completed. In this clock cycle, the high order word of the product is stored in redundant format: the sum vector in the flip-flops (FFs) situated between two full adders (FAs), meanwhile the carry vector is stored in the FFs that

implement the carry feedback. To generate these bits with the right format, these two vectors have to be added by inserting W_c-1 zeros into the input. This multiplier has 2 clock cycles latency because of the final bit-serial adder stage, and needs W_d+W_c-1 clock cycles to compute a full precision result.

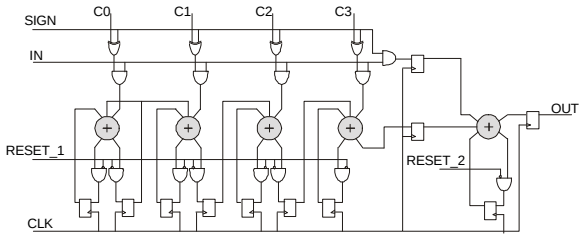

Fig. 2: Serial/parallel multiplier

The digit-serial version of the previous multiplier (Fig.2) with N=2 bits digit-size is shown in Fig. 3. Assuming that coefficient and data have the same size ($W_d=W_c=W$ bits), in the first W/N clock cycles the data-digits enter serially with least significant digit (LSD) first. The low order W/N digits of the product are outputted also with LSD first. When the last digit of the data enters (that which contains the sign bit) the control signal SIGN must be high. The coefficient is one's complemented and the sign bit, x_3, is latched to be fed into the final bit-serial adder in the next clock cycle. In this way, the two's complement can be calculated. At the same time, the high order word of the product is stored with redundant format. The high order product is produced by inserting W/N zeros into the input. As a result, the multiplier also has 2 clock cycles latency and needs 2W/N clock cycles to compute a full precision result.

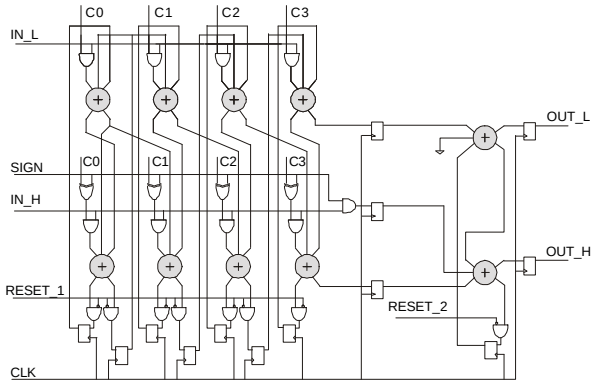

Fig. 3: Serial/parallel multiplier with 2-bits digit-size

TABLE I: Results of the FPGA implementation

Digit-size (bits)	Fclock (MHz)	Fsample (MHz)	Area (LEs)	Logic depth (LEs)
1	105.3	6.6	35	2
2	95.2	11.9	53	2
4	53.2	13.3	87	4
8	26.2	13.1	157	8

A family of these multipliers has been design by changing the digit-size. Each topology has been checked by implemented it on a FPGA. Main results for the different digit-sizes, are shown in Table I. The maximum clock frequency, the sample frequency, the occupied area in the chip and the logic depth of each multiplier are summarised.

3. PIPELINED SERIAL/PARALLEL MULTIPLIER

In this section, the pipelined version of the previous multipliers is presented. The goal is to maintain the throughput without adding extra clock cycles to inset zeros and, therefore, they compute a complete product in W/N cycles. It is achieved by manipulating the resulting product in double precision format [2], at the cost of using the double of output wires. Another singularity of these multipliers is that they compute the double of the product of the data by the coefficient (*2.X.C*), which is managed by inserting a zero in the LSB and ignoring the MSB.

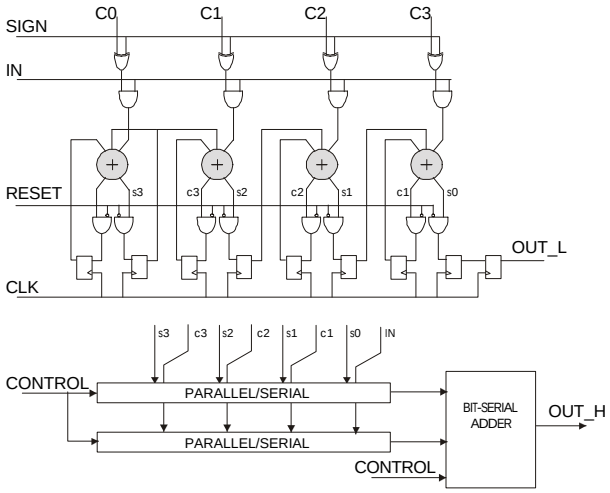

Fig. 4: Pipelined serial/parallel multiplier

The bit-serial pipelined serial/parallel multiplier is depicted in Fig. 4. During the first W clock cycles (those in which the bits of the data are inputted) it operates like the unpipelined version. During these cycles, the low order bits of the product are outputted through OUT_L. When the sign bit enters into the circuit, the CONTROL signal must be high. As a consequence, the sum, carry vector and the sign bit are stored in two parallel/serial converters (PSCs). The PSCs outputs are connected to a bit-serial adder, which computes the high order bits of the product. This multiplier computes at the same time the low order product of the current data and the high order product of the previous one.

The digit-serial version of the pipelined serial/parallel multiplier (Fig. 4) with N=2 bits digit-size is shown in Fig. 5. In the first W/N clock cycles, the data-digits enter serially with LSD first and the digits of the low order product are outputted, also with LSD first, via OUT_LX. When the digit which contains the sign bit (the last digit of each word), enters into the circuit, the CONTROL signal must be high, therefore, the sum and carry vector together with the sign bit are stored in two PSCs. The

PSCs outputs are connected to a digit-serial adder, which computes the high order bits of the product and whose digits are outputted by OUT_HX.

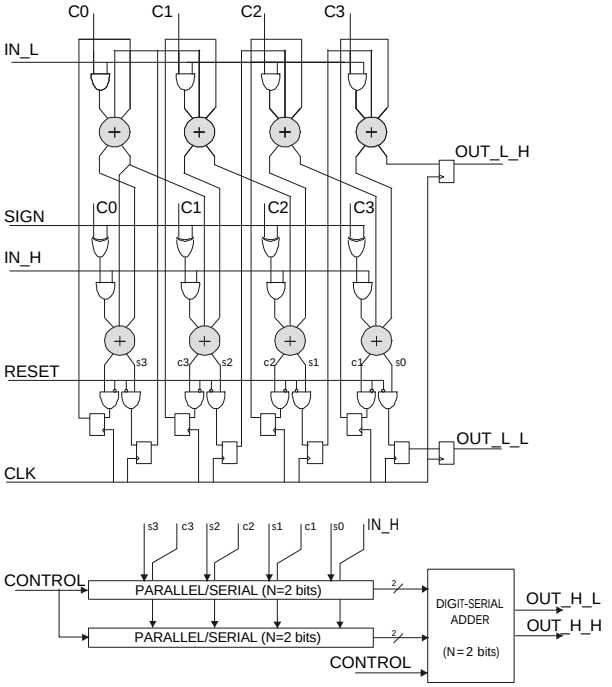

Fig. 5: Pipelined serial/parallel multiplier (N=2 bits)

A set of pipelined serial/parallel multipliers has been designed by changing the digit-size. Each circuit has been implemented on FPGA. The maximum clock frequency, the sample frequency, the occupied area and the logic depth of each multiplier are shown in Table II.

TABLE II: Results of the FPGA implementation

Digit-size (bits)	Fclock (MHz)	Fsample (MHz)	Area (LEs)	Logic depth (LEs)
1	57.1	7.1	64	3
2	63.7	15.9	80	3
4	44.2	22.1	111	5
8	25.5	25.5	187	9

4. DISCUSSION

In the abstract, the pipelined serial/parallel multiplier should double the throughput with respect to unpipelined one. However, in an actual implementation it does not occur, as can be seen by comparing table II and I. The throughput achieved by the pipelined multipliers is only a little higher than the corresponding to the unpipelined version, the increment varies form 107% using the bit-serial version to 194% with the bit-parallel one. This effect is caused by the structure of the FPGA selected: a matrix organisation in which each element is a k-input LUT (k=4 in the case of Altera). So, the circuits have to be divided into 4-input functions in order to be implemented. In Fig. 6 this division can be observed for the case of bit-serial multipliers. The serial/parallel multiplier uses 2 LEs logic depth and the pipelined one 3 LEs. This increment in the logic depth is caused by the inclusion of the PSCs in the serial/parallel multiplier, that needs 1 LE logic depth to be implemented. This increment of the logic depth in the pipelined multipliers causes the decrement of the throughput respect to the ideal case.

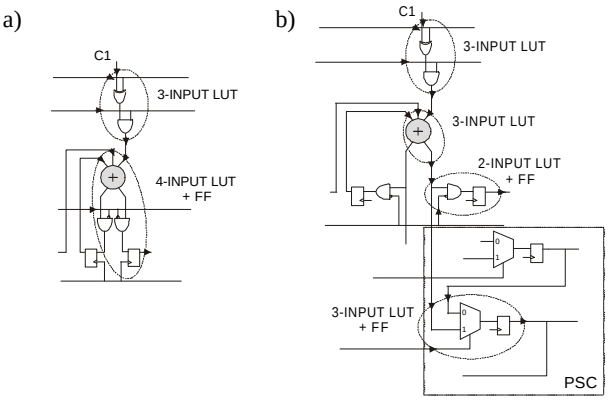

Fig. 6: Multipliers separated in LEs:
a) unpipelined; b) pipelined.

5. FAST PIPELINED SERIAL/PARALLEL MULTIPLIER

The novel multiplier structure presented in this paper is based in the fast serial/parallel multiplier proposed by R. Gnanasekaran in [1] and whose structure is shown in Fig. 7. The idea is to avoid W extra clock cycles to complete the computation by adding the sum and carry word, obtained after the first W cycles, with a bit-parallel adder. Thus, the multiplication time is improved if the addition of sum and carry word is performed with a bit-parallel adder instead of using W more clock cycles to propagate the carries.

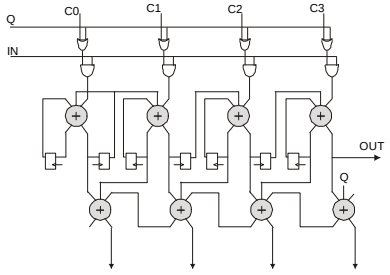

Fig. 7: Fast serial/parallel multiplier [1]

This idea can be exploited when the multipliers are mapped on FPGAs where their fast carry lines can be used to implement the ripple carry adders. The circuit depicted in Fig. 7 does not have better performance than the pipeline serial/parallel multiplier because ripple carry addition increases the logic depth. The way to improve its performance is to pipeline the circuit, but there will only be throughput increment if:

- the logic depth of the serial/parallel multiplier is not incremented
- the propagation delay in the bit-parallel adder is at least as short as it is in a circuit with minimum logic depth (2 LEs).

Furthermore, the new multiplier must give the result with the double precision format as the pipeline serial/parallel multiplier does and must be suitable for digit-serial processing.

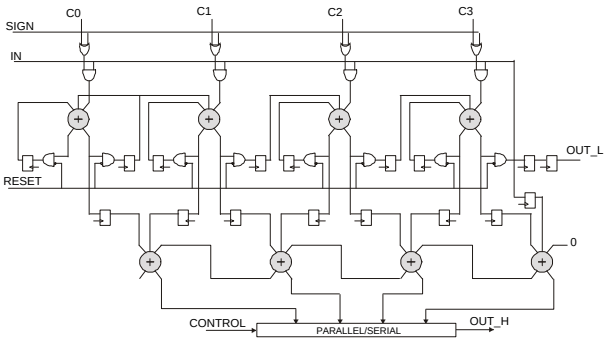

Fig. 8: Fast pipelined serial/parallel multiplier

This multiplier for the bit-serial version is shown in Fig. 8. To avoid incrementing the logic depth, load signal must not appear and only registers need to be included. Therefore, in each clock cycle, a ripple carry addition is performed. The parallel outputs of the adder are fed into a PSC in order to output the high order word of the multiplier with the right format. The adder outputs are captured into the PSC during the first bit of each word, that is, only once each W clock cycles. This operation is managed by the CONTROL signal.

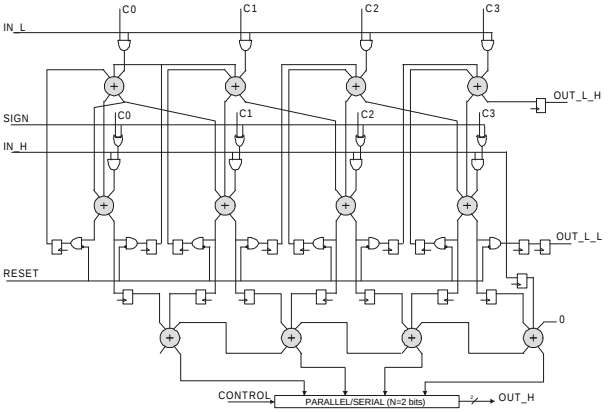

Fig. 9: Fast pipelined serial/parallel multiplier (N=2 bits)

The digit-serial version of this multiplier with 2-bits digit-size is shown in Fig. 9. The parallel outputs of the adder are fed into a PSC in order to give the high order word of the multiplier with the right digit-serial format. The CONTROL signal must be high during the first digit of each word (once each W/N cycles) and, as a consequence, the adder outputs will be captured into the PSC.

The results of the FPGA implementation of a set of these multipliers can be seen in Table III. The only difference in the implementation of this case with respect to the previous ones is that the FAST logic synthesis option has been used only to the bit-parallel adder (it allows the use of the carry chain lines).

As can be seen in Table III, the fast pipeline serial/parallel multipliers proposed in this paper achieve better performance than the previous versions. The speed improvement ranges from 156% using the bit-serial version to 115% with the bit-parallel one. The fast pipelined circuits have the same logic depth than the unpipelined serial/parallel ones. Therefore, they achieve higher throughput than the pipelined serial/parallel multipliers with nearly the same cost in area. If they are compared to the unpipelined multipliers, the throughput improvement varies from 169% for the bit-serial version, to 200% with the bit-parallel one. For example, the proposed multiplier with 2-bits digit-size is faster than the serial/parallel multiplier with 8-bits digit-size and it occupies a half area respect to it.

TABLE III: Results of the FPGA implementation

Digit-size (bits)	Fclock (MHz)	Fsample (MHz)	Area (LEs)	Logic depth (LEs)
1	89.3	11.2	64	2
2	87.7	21.9	81	2
4	48.6	24.3	112	4
8	28.8	28.8	169	8

6. CONCLUSIONS

A set of fast pipelined serial/parallel multipliers has been proposed. These structures reduce the logic depth with respect to the pipelined serial/parallel multiplier and have the same than in the unpipelined case. As a consequence, they are more efficient than the pipelined serial/parallel ones: they achieve higher throughput with nearly the same cost in area. Furthermore, they are closer to the abstract case: they manage nearly the double of throughput than the unpipelined ones.

REFERENCES

[1] R. Gnanasekaran, “A fast serial-parallel binary multiplier”, IEEE Trans. On Computers, Vol. C-34, NO. 8, August 1985

[2] P. Denyer and D. Renshaw, VLSI SIGNAL PROCESSING: A Bit-Serial Approach, Addison-Wesley, 1985.

[3] R. Petersen and B. Hutchings, “An Asssesment of the Suitability of FPGA-based Systems for Use in DSPs”, in Lecture Notes in Computer Science, nº975, pp.293-302, Springer-Verlag, Berlin: 1995.

[4] S.G. Smith and P.B. Denyer, Serial Data Computation, Kluwer Academic, Boston, MA, 1988.

[5] R. Harley and P.Corbet, “Digit-serial processing techniques”, IEEE Trans. On Circuits and Systems, Vol. 37, no. 6, pp. 707-719, June 1990.

[6] K.K. Parhi and C. Wang, “Digit-serial DSP architectures”, in Proc. of Int. Conf. On Application Specific Array Processors, pp. 341-351, September 1990.

[7] K.K. Parhi, “A systematic approach for design of digit-serial signal processing architectures”, IEEE Trans. Circuits and Systems, Vol. 38, pp.358-375, April 1991.

[8] R.I. Hartley and K.K. Parhi, Digit-Serial Computation, Kluwer Academic, Boston, MA, 1995.

[9] J. Valls, M.M. Peiro, T. Sansaloni and E. Boemo, “A study about FPGA-based digital filters”, 1998 IEEE Workshop on VLSI Signal Processing: Design and Implementation (SiPS’98), pp. 192-201, Boston, MA.