A Reconfigurable Multiplier Array For Video Image Processing Tasks, Suitable For Embedding In An FPGA Structure

Simon D. Haynes, Peter Y. K. Cheung

Abstract ¾ **This paper presents a design for a reconfigurable multiplier array. The multiplier is constructed using an array of 4 bit Flexible Array Blocks (FABs), which could be embedded within a conventional FPGA structure. The array can be configured to perform a number of 4*n* x 4*m* bit signed/unsigned binary multiplications. We have estimated that the FABs are about 35 times more efficient in area than the equivalent multiplier implemented using a conventional FPGA structure alone.**

I. Introduction

IT has been suggested that FPGAs are well suited for use as reconfigurable hardware to accelerate software in many applications [1]. Image/video processing tasks are particularly well suited to hardware acceleration, because of the inherent parallelism and data flow structure. Common to many image/video processing tasks is the need for intensive arithmetic operations such as multiplication and addition.

Whilst existing FPGA architectures are well suited to binary addition [2], configuring FPGAs for binary multiplication results in the available reconfigurable resources being used inefficiently [3]. Typically over 70% of the FPGA resources could be used solely for multiplication in some applications. The literature also suggests that hardware implemented on an FPGA requires as much as 100 times more die area, and will be about 10 times slower than the custom hardware equivalent [4].

One possible solution would be to embed custom multipliers into the FPGA structure. The difficulty with this is that inefficiencies will result if the size of the multiplier is not compatible with that of the algorithm. We suggest that a better solution is to use FPGAs with embedded *reconfigurable multiplier blocks*. In this paper we suggest a design for a reconfigurable 4 bit flexible array block (FAB), an array of which can be combined together to construct a multiplier which has speed comparable to that of a conventional signed array multiplier, with minimal extra cost in hardware required for reconfiguration. The multiplier can be configured to perform any 4*n* x 4*m* bit signed/unsigned binary multiplication.

Simon D. Haynes is with the Department of Electrical and Electronic Engineering, Imperial College of Science Technology and Medicine, London, England: E-mail: s.d.haynes@ic.ac.uk.

Peter Y. K. Cheung is with the Department of Electrical and Electronic Engineering, Imperial College of Science Technology and Medicine, London, England: E-mail: p.cheung@ic.ac.uk.

A. Organisation Of The Paper

Section B briefly discusses an existing reconfigurable scheme, and explains why most of the techniques for implementing fast multipliers are not well suited for a reconfigurable design. Section II presents our design for a flexible array block (FAB), and explains how an array of FABs can be used to create a 4*n* x 4*m* multiplier. Section III gives a proof of the functionality for the proposed design. The FAB scheme is compared to existing fixed size multipliers, the Hwang reconfigurable multiplier, and multipliers implemented using conventional FPGAs structures in section IV. A possible modification to the FAB scheme is discussed in section D, to allow simple implementation of multiplier accumulators, and FIR filters.

B. Previous Designs

Hwang has suggested constructing *Universal Multiplication Networks* using small (4 bit) *Programmable Additive Multiply* (PAM) modules[5]. Whilst simple in design, these networks have the drawback of slow multiplication times and a non-scaleable connection pattern, especially for large operand sizes.

Most other multiplier architectures are concerned with the multiplication of two multiplicands of fixed length. The techniques used for speeding up the multiplication are largely at the expense of regularity, such as Wallace Trees [6], or require some form of 'pre-processing' of the operands, e.g. Booth's Modified algorithm [7]. Neither of these styles of design are well suited to generalisation for multiplicands of variable size, thus making it difficult to create a reconfigurable multiplier based on these methods.

II. New Design

This section outlines the proposed new design for a reconfigurable multiplier based on 4 bit flexible array blocks .

A. The Design of the Flexible Array Block (FAB)

We propose the design shown in Fig. 2 for a 4x4 flexible array block (FAB). The block uses a modification of the array developed by Baugh-Wooley for two's complement multiplication [8]. The FAB consists of two parts: i) A multiplier array which reduces the four bit multiplication to two 5 bit numbers, and ii) an adder to produce the final output. The adder is unused unless the FAB is in the final column of overall multiplier array (i.e. it uses the most significant bit of the multiplicand *A*).

Each FAB is configured using the six configuration bits M_a, M_b, C_l, C_r, C_t, and C_b as shown in table TABLE I. The configuration of each FAB will be denoted by $FAB_{(Ma,Mb,Cl,Cr,Ct,Cb)}$

TABLE I

CONFIGURATION SETTINGS FOR THE FLEXIBLE MULTIPLIER BLOCK .

Bit	Meaning
Ma	High if A3 is the MSB of a signed number.
Mb	High if B3 is the MSB of a signed number.
Cl	High if A0 is not the LSB of the multiplicand A (The FAB is connected to the FAB on the left).
Cr	High if A3 is not the MSB of the multiplicand A (The FAB is connected to the FAB on the right).
Cb	High if B3 is not the MSB of the multiplicand B (The FAB is connected to the FAB on the bottom).
Ct	High if B0 is not the LSB of the multiplicand B (The FAB is connected to the FAB on the top).

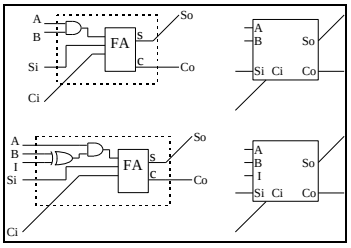

Fig. 1, The two basic units used in the FAB. (FA = Full Adder)

B. General Connection Scheme

Fig. 3 shows the proposed connection scheme for an embedded 4x4 array of FABs. The interconnect is both regular and scaleable, allowing simple VLSI implementation and expansion to larger array sizes. It is suggested that the interconnect is dedicated, thus leaving free the valuable reconfigurable interconnect resources of the FPGA. The array of FABs could either be placed physically together in one location of the FPGA, or distributed regularly throughout the FPGA structure.

Such an array of FABs can be configured to perform a number of multiplications, with multiplicands of varying sizes. For example, the array shown in Fig. 3 is capable of performing a single 16x16 bit multiplication, or 16 4x4 bit multiplication (unsigned, or two's complement in either case).

To configure the array, the six configuration bits of each FAB must be set appropriately, the correct multiplicands supplied to each FAB, and the output(s) taken from the appropriate place.

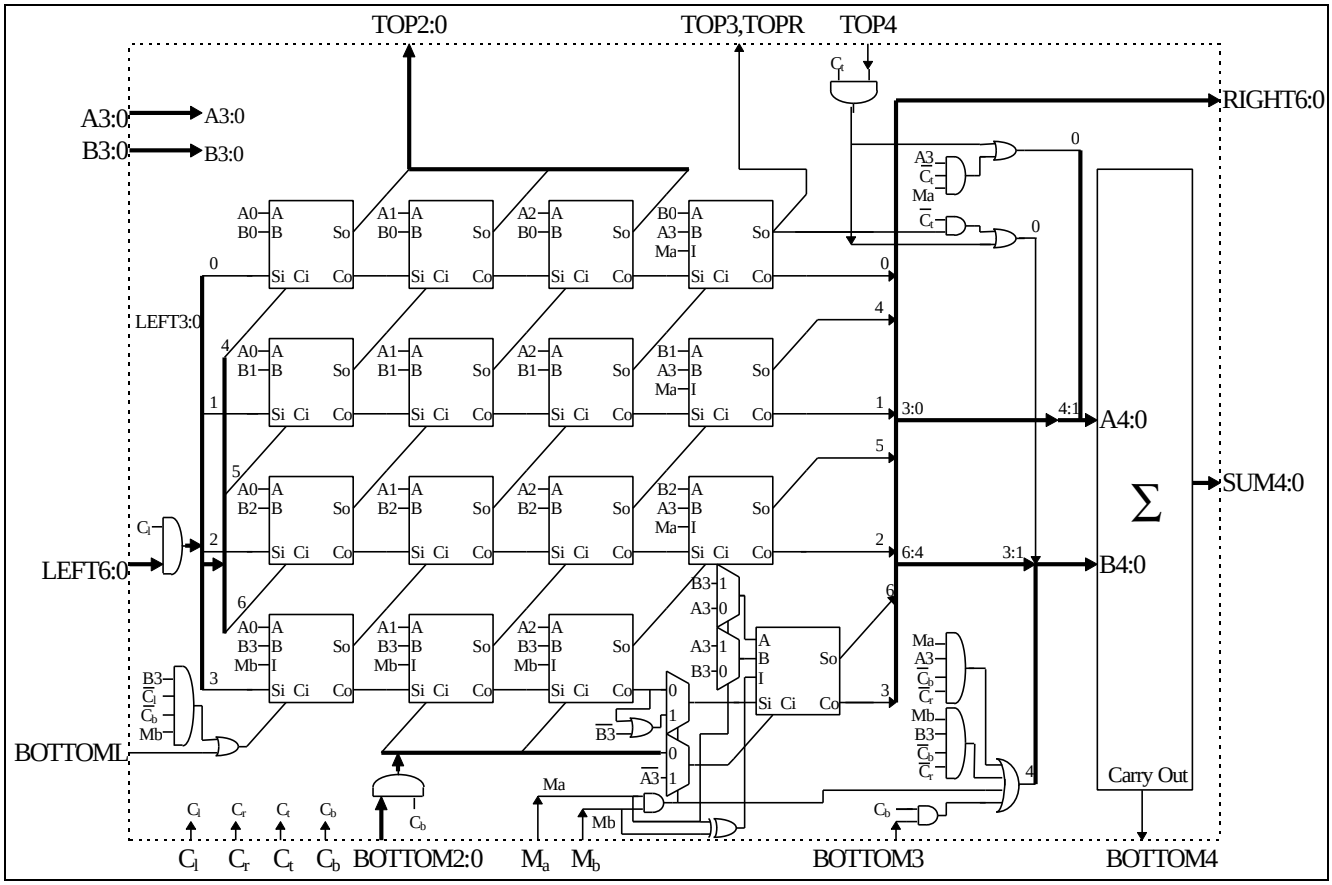

Fig. 2, Schematic for a 4x4 flexible array block (FAB)

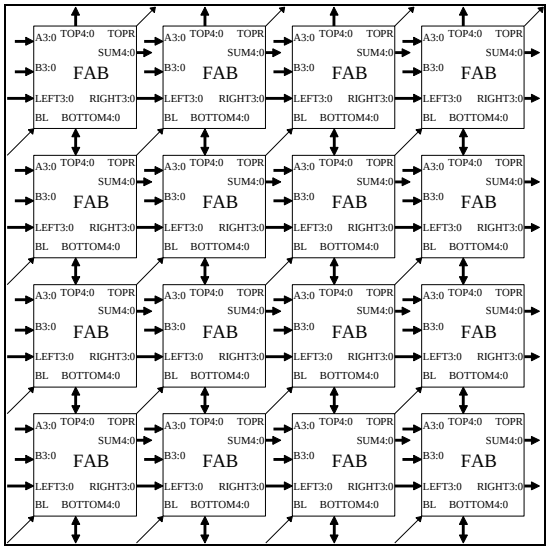

Fig. 3, The proposed connection scheme for a 4x4 array of FABs

C. Using FABs to Construct Multipliers

Fig. 4 shows how 4 FABs, as described above, can be configured to produce an unsigned 8x8 bit multiplier. Fig. 5 shows the configuration necessary for 6 FABs to generate an 8x12 bit two's complement multiplier.

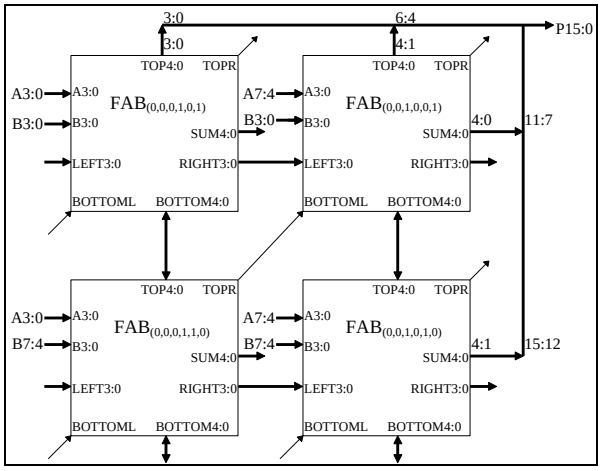

Fig. 4, An unsigned 8x8 multiplier using 4 FABs.

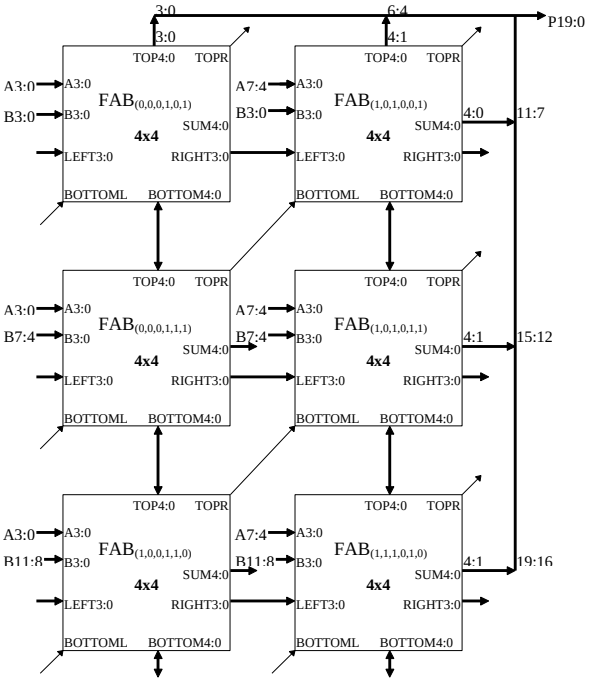

Fig. 5, A two's complement 8x12 bit signed multiplier, constructed using 6 FABs

III. Proof Of Multiplier

The multiplier as described in section II has three modes of operation: i) Both multiplicands are unsigned, ii) Both multiplicands are signed, and iii) One multiplicand is unsigned, the other signed.

The proof for each case will be considered in this section.

A. Both Multiplicands Are Unsigned

A and *B* are unsigned *n* and *m* bit binary numbers, respectively, as given by (1) (both *m* and *n* must be multiples of 4)

$$A = \sum_{i=0}^{n-1} 2^i * a_i \quad , \quad B = \sum_{j=0}^{m-1} 2^j * b_j \tag{1}$$

It can be easily shown that *A*B* is given by (2)

$$A * B = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} 2^{i+j} * a_i \cdot b_j \tag{2}$$

When the array of FABs has been correctly configured for a *n* by *m* bit unsigned multiplication, an array of elements equivalent to that shown in Fig. 6 exists.

The output of this array is given by (3)

$$Out = 2^n (Y^n + X^n) + \sum_{i=0}^{n-1} 2^i * y_0^{i+1} \tag{3}$$

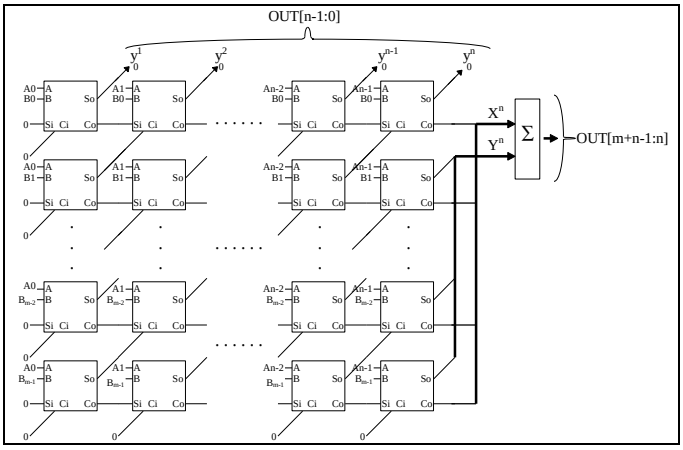

Fig. 6, The equivalent array for an *n* by *m* bit unsigned multiplier

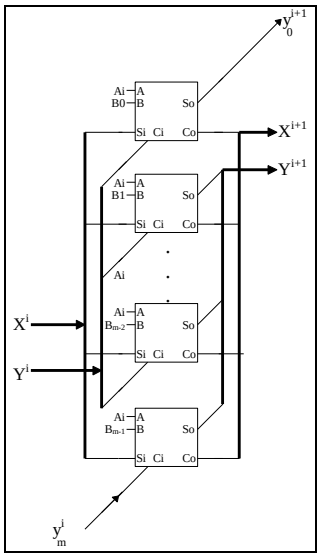

Fig. 7, One column of the multiplier array

Consider one column of the multiplier array, as shown in Fig. 7. The output of the column is given by (4). By using the appropriate boundary conditions it is easily shown that the output of the multiplier array is given by (5), which is equivalent to the required result, as derived in (2).

$$2(Y^{i+1}+X^{i+1})+y_0^{i+1}=X^i+Y^i+2^{m-1}y_m^{i-1}+\sum_{j=0}^{m-1}2^j*a_i\cdot b_j \qquad (4)$$

Boundary Conditions: $Y^0=X^0=y_m^i=0$

$$2^n(Y^n+X^n)+\sum_{i=0}^{n-1}2^i*y_0^{i+1}=\sum_{i=0}^{n-1}2^i*\sum_{j=0}^{m-1}2^j*a_i\cdot b_j \qquad (5)$$

B. When Both Multiplicands Are Two's Complement Signed Numbers.

A and *B* are both two's complement signed *n* and *m* bit binary numbers, respectively, as given by (6).

$$A=\sum_{i=0}^{n-2}\left[2^i*a_i\right]-2^{n-1}*a_{n-1}\ ,\ B=\sum_{j=0}^{m-2}\left[2^j*b_j\right]-2^{m-1}*b_{m-1} \qquad (6)$$

Therefore, it can be shown that *A*B* is given by

(7)

$$A*B=\left(\sum_{i=0}^{n-2}\sum_{j=0}^{m-2}2^{i+j}*a_i*b_j\right)-a_{n-1}*2^{n-1}*\left(\sum_{j=0}^{m-2}2^j*b_j\right)-b_{m-1}*2^{m-1}*\left(\sum_{i=0}^{n-2}2^i*a_i\right)+a_{n-1}*b_{m-1}*2^{m+n-2} \qquad (7)$$

When the array of FABs has been correctly configured for a *n* by *m* bit two's complement signed multiplication, an array of elements equivalent to that shown in Fig. 8 exists. The OR gate, present in Fig. 8, can be considered equivalent to an adder, because both inputs can never be high.

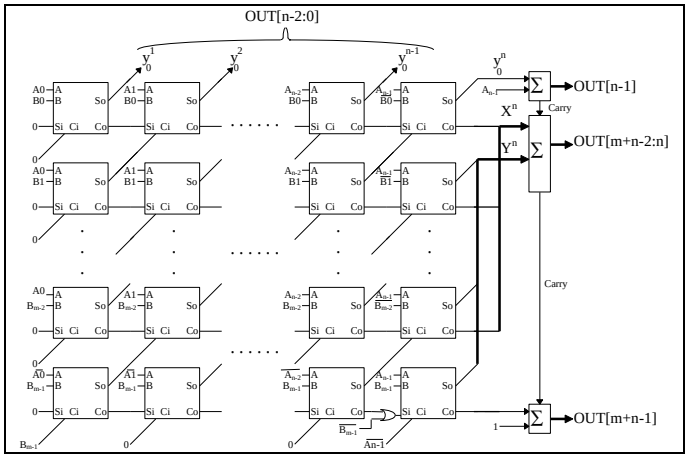

Fig. 8, The equivalent array for an *n* by *m* bit signed multiplier

The output of this array can be seen to be given by (8).

$$Out=2^n(Y^n+X^n)+\sum_{i=0}^{n-1}2^i*y_0^{i+1}+2^{m+n-1}+2^{n-1}a_{n-1} \qquad (8)$$

Consider the i^{th} column of this array as shown in Fig. 9. The output of this column is given by (9).

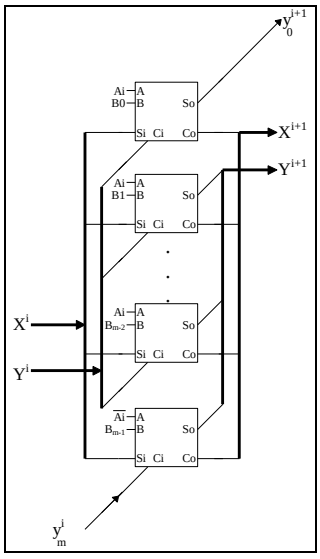

Fig. 9, One column of the multiplier array

$$\begin{aligned}&2(Y^{i+1}+X^{i+1})+y_0^{i+1}=X^i+Y^i+2^{m-1}y_m^{i-1}+\sum_{j=0}^{m-2}2^j*a_i\cdot b_j+2^{m-1}*\bar{a}_i\cdot b_{m-1},\ (i\neq n-1)\\&2(Y^n+X^n)+y_0^n=X^{n-1}+Y^{n-1}+\sum_{j=0}^{m-2}2^j*a_i\cdot\bar{b}_j+2^{m-1}*(a_{n-1}\cdot b_{m-1}+\bar{b}_{m-1}+\bar{a}_{n-1}),\ (i=n-1)\end{aligned} \qquad (9)$$

Using the appropriate boundary conditions, it is possible to calculate the output of the multiplier array as shown in (10).

$$\text{Boundary Conditions: } Y^0 = X^0 = 0, y_m^i = 0\ (i \neq 0),\ y_m^0 = b_{m-1}$$
$$\sum_{i=0}^{n-2}\sum_{j=0}^{m-2} 2^{i+j} a_i \cdot b_j + 2^{n-1}\sum_{j=0}^{m-2} 2^j a_{n-1} \cdot \bar{b}_j + 2^{m-1}\sum_{i=0}^{n-2} 2^j \bar{a}_i \cdot b_{m-1} + 2^{m-1} b_{m-1} + 3*2^{m+n-1}(a_{n-1}\ OR\ b_{m-1}) \quad (10)$$

This can be shown to be equivalent to (7)

C. One Multiplicand Unsigned, The Other A Two's Complement Signed Number.

A is an *n* bit two's complement signed binary number, and *B* is unsigned *m* bit binary number as given in (11).

$$A = \sum_{i=0}^{n-2}\left[2^i * a_i\right] - 2^{n-1} * a_{n-1}\ ,\ B = \sum_{j=0}^{m-1}\left[2^j * b_j\right] \quad (11)$$

Therefore, it can be shown that *A*B* is given by (12).

$$A*B = \left(\sum_{i=0}^{n-2}\sum_{j=0}^{m-2} 2^{i+j} * b_j * a_i\right) - a_{n-1} * 2^{n-1} * \left(\sum_{j=0}^{m-1} 2^j * b_j\right) \quad (12)$$

When the array of FABs has been correctly configured for a *n* by *m* bit multiplication, with *A* and *B* defined as in (11), an array of elements equivalent to that shown in Fig. 10 exists.

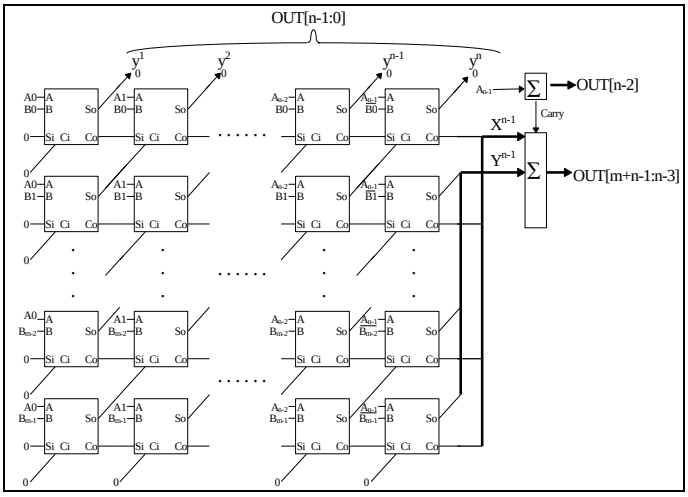

Fig. 10, The equivalent array for an *n* by *m* bit multiplier, *A* signed, *B* unsigned

The output of this array can be seen to be given by (13).

$$Out = 2^n(Y^n + X^n) + \sum_{i=0}^{n-1} 2^i * y_0^{i+1} \quad (13)$$

Consider the i^{th} column of this array as shown in Fig. 11. The output of this column is given by (14).

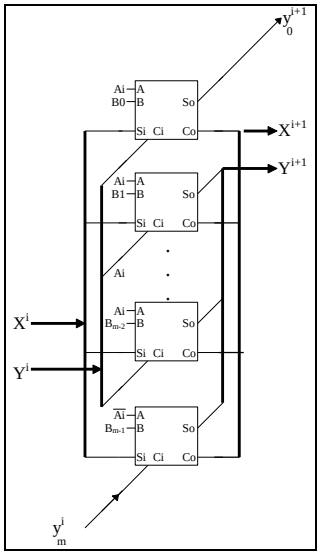

Fig. 11, One column of the multiplier array

$$\begin{aligned} 2(Y^{i+1} + X^{i+1}) + y_0^{j+1} &= X^i + Y^i + 2^{m-1} y_m^{j-1} + \sum_{j=0}^{m-1} 2^j * a_i \cdot b_j\ ,\ (i \neq n-1) \\ 2(Y^n + X^n) + y_0^n &= X^{n-1} + Y^{n-1} + 2^{m-1} y_m^{n-1} + \sum_{j=0}^{m-2} 2^j * a_i \cdot \bar{b}_j,\ (i = n-1) \end{aligned} \quad (14)$$

Using the appropriate boundary conditions, it is possible to calculate the output of the multiplier array as shown in (15).

$$\text{Boundary Conditions: } Y^0 = X^0 = y_m^i = 0$$
$$\sum_{i=0}^{n-2}\sum_{j=0}^{m-1} 2^{i+j} a_i \cdot b_j + 2^{n-1}\sum_{j=0}^{m-1} 2^j a_{n-1} \cdot \bar{b}_j + 3*2^{m+n-2} a_{n-1} \quad (15)$$

This can be shown to be equivalent to (12).

D.Summary of Proofs

This section has proved that the multiplier array will work correctly for unsigned, and signed two's complement numbers, when properly configured.

IV. Comparison with Existing Designs

The FABs are compared to the PAMs, as described by Hwang, in terms of speed and gate count. It is also compared to two fixed operand size schemes, the Baugh-Wooley Array, and the Wallace/Dadda to show that the cost of reconfigurability is small.

The FAB scheme is also compared to multipliers implemented using conventional FPGA structures, to show that there is a significant reduction in the die area required.

A.Speed of the FAB Scheme

Table II shows that the delay of the FABs is much better than the Hwang reconfigurable array, and is comparable to that of the Baugh-Wooley fixed size operand array. The delay of the PAMs is of $O(N^2)$, whilst the delay of the FABs is of O(N). The delay of the Wallace/Dadda multiplier is of $O(\log_2 N)$.

TABLE II

GATE DELAY FOR VARIOUS MULTIPLIER SCHEMES.

Scheme	Configurable	8 Bits	16 Bits	32 Bits	64 Bits
FABs	**yes**	**34**	**66**	**130**	**258**
Hwang's PAMs	**yes**	48	160	576	2 176
Baugh-Wooley	**no**	30	62	126	254
Wallace/Dadda	**no**	32	52	80	124

(Assuming that Full Adders are used to implement the final addition stage in all cases)

B.Gate Count for the FAB scheme

Table III shows that, in terms of the number of gates required, the FABs are slightly more costly (≈3%) than the existing Hwang reconfigurable scheme. The FAB scheme is more costly than either of the two fixed size operands schemes, with 50% more gates being required.

TABLE III

COMPARISON OF THE NUMBER OF GATES REQUIRED FOR VARIOUS MULTIPLIER SCHEMES.

Scheme	Configurable	8 Bits	16 Bits	32 Bits	64 Bits
FABs	**yes**	**664**	**2 656**	**10 624**	**42 496**
Hwang's PAMs	**yes**	644	2 576	10 304	41 216
Baugh-Wooley	**no**	413	1 701	6 965	28 245
Wallace/Dadda	**no**	379	1 763	7 603	31 571

(Assuming that Full Adders are used to implement the final addition stage in all cases)

C.Number of Interconnects Required Between FABs

The price which must be paid for the significant increase in speed when using FABs, compared to the PAMs is the 50% increase in the number of interconnects between the FABs, when compared with the PAMs, as shown in table TABLE IV.

TABLE IV

COMPARISON OF THE NUMBER OF INTERCONNECTS BETWEEN EACH BLOCK.

Scheme	Number of interconnects for a 4x4 reconfigurable block
FABs	**39**
Hwang's PAMs	26
Baugh-Wooley	N/A
Wallace/Dadda	N/A

(N.B. figures exclude the configuration control, and power)

D.Comparison with conventional FPGA structures

The FAB scheme is compared to multipliers implemented using Altera's FLEX10K, and Xilinx's 4000 series FPGAs, in terms of total die area required.

An estimate of transistor count has been chosen as the best metric for comparing the relative die area for each scheme. This is a reasonable assumption because most modern processes have a large number of metal layers, making routing less of a bottle neck. The number of transistors includes any required for the configuration, and routing of any cells used.

For all the considered schemes the amount of hardware required to implement an n x n bit multiplier proportional to n^2, for $n>4$ and n a multiple of 4.

A Xilinx 4000 FPGA requires, on average, $1.14*n^2$ CLBs (Configurable Logic Blocks) to implement an n x n bit multiplier (73 CLBs are need for an 8x8 multiplier [9]). For the Xilinx 4000 part each Cell (CLBs in this instance) requires 4 700 transistors. This figure includes all transistors required for configuration, routing, and CLB logic.

From extensive experimentation, we have found that the Altera Flex10K part requires, on average, $0.41*n^2$ LABs (Logic Array Blocks) to implement a n x n multiplier. We have estimated that each LAB requires about 13 700 transistors, for configuration, routing, and cell logic.

Using the FAB scheme, each 4x4 Cell requires approximately 2 600 transistors, with 0.0625 FABs being required per bit^2

Table III shows that the FAB scheme requires about 35 times less die area than either the FLEX10k, or 4000 FPGA.

TABLE V

COMPARISON OF THE DIE AREA REQUIRED FOR IMPLEMENTING MULTIPLIERS USING FPGAS AND THE FAB SCHEME

Implementation	Trans. Per Cell	Cells Per Bit^2	Trans. Per Bit^2	Relative Area
FABs	**2 600**	**0.0625**	**160**	**1**
Altera Flex10K	13 700	0.41	5600	35
Xilinx 4000	4 700	1.14	5400	34

V. EXTENSION OF THE UTILISATION OF THE FABS

The 5-bit adder in each FAB is only used if the FAB is as the last stage of the multiplier. Therefore if the size of either multiplicand is greater than 4 bits, there are a number of 'spare' adders available. With minimal cost in terms of extra hardware and multiplier performance, it would be possible to extend these adders to 8-bit adders, as shown by Fig. 12, and Fig. 13.

This would require 4 more configuration bits to sign extend the inputs to the adders, and 2 additional vertical connections between each FAB.

This small alteration would enable simple implementation of structures such as multiply-accumulators (MACs). and finite impulse response filters (FIRs), to be embedded within the multiplier array. Fig. 14 shows how an 8x8 MAC can be implemented using 4 FABs. The FABs should be connected as before.

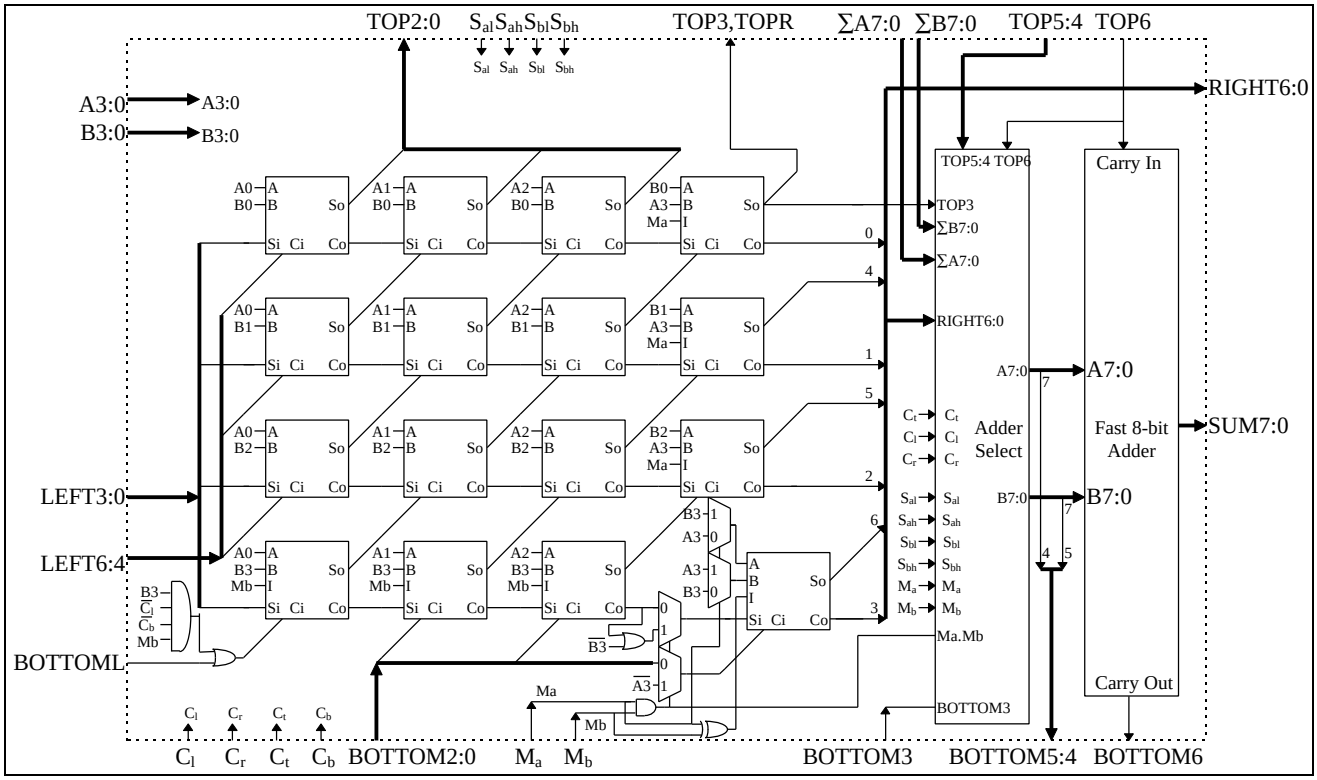

Fig. 12, Extension to the original FAB

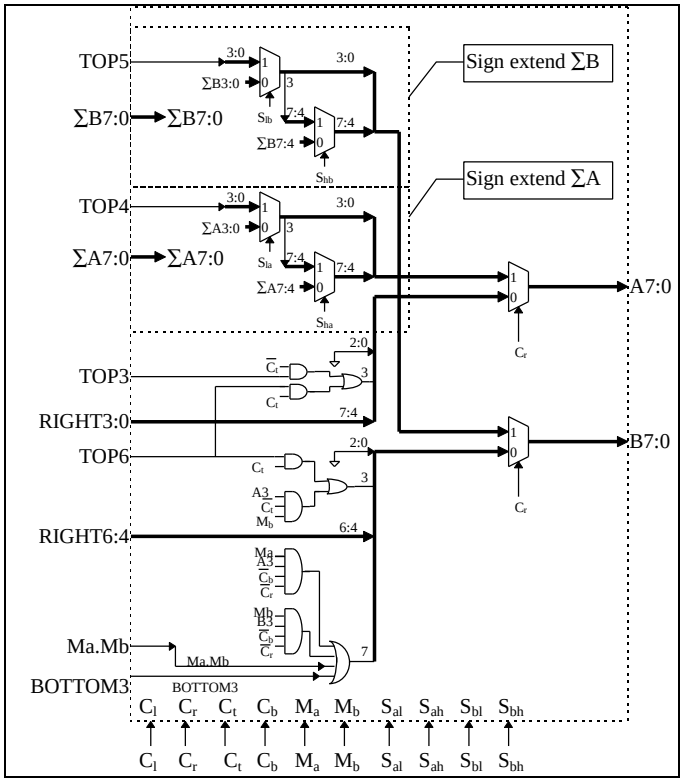

Fig. 13, The adder select block for Fig. 12

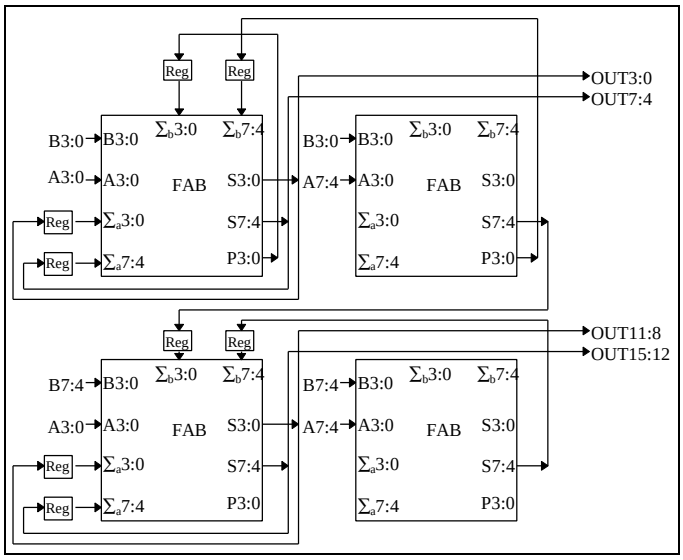

Fig. 14, an 8x8 MAC implemented using 4 extended FABs

VI. Conclusion

We have proposed the design for a new reconfigurable flexible multiplier which is considerably faster than existing reconfigurable multipliers reported previously in the literature. The speed improvements are gained at the cost of adding extra interconnects between the reconfigurable blocks. We estimate that this design is approximately 35 more efficient in terms of silicon area than using a conventional FPGA structure alone. It is likely that such a design would also yield considerable improvements in speed. Such a design should enable substantial savings of the 'standard' reconfigurable resources of an FPGA when used for image/video processing applications.

[1] P.M. Athanas, and A.L. Abbott, "*Real-time image processing on a custom computing platform*", IEEE Computer, Vol. 28, No. 2 , pp. 16-24, 1995
[2] Altera Corporation. "*Ripple-carry adders in FLEX 8000 devices*", Application Brief 118, ver. 2, May 1994
[3] O.T. Albaharna, P.Y.K. Cheung, and T.J. Clarke, "*On the viability of FPGA-based integrated coprocessors*", IEEE Symposium on FPGAs for Custom Computing Machines, April 17th - 19th 1996
[4] O.T. Albaharna, P.Y.K. Cheung, and T.J. Clarke, "*Area & time limitations of FPGA-based virtual hardware*", IEEE Proceedings International Conference on Computer Design: VLSI in computers and Processors, pp184-189, 1994
[5] K. Hwang, "*Computer arithmetic Principles, Architecture, and Design*", John Wiley & Sons, 1st edn., pp. 194-197, 1979
[6] C.S. Wallace, "*A suggestion for fast multipliers*", IEEE Trans. Electronic Computers, Vol. EC-13, pp. 14-17, Feb. 1964
[7] A.D. Booth, "*A signed binary multiplication technique*", Quarterly J. Mechan. Appl. Math., Vol. 4, Pt. 2, pp. 236-240, 1951
[8] A.R. Baugh, and B.A. Wooley, "*A two's complement parallel array multiplication algorithm*", IEEE Trans. Computers, Vol. C-22,No. 1-2, pp. 1045-1047, December 1973
[9] Xilinx, "*The Programmable Logic Data Book*", Version 1.02, Chapter 4, p. 7, June 1, 1996