A Digit-Serial Structure for Reconfigurable Multipliers

Chakkapas Visavakul, Peter Y. K. Cheung, and Wayne Luk

Department of EEE, Imperial College, Exhibition Road, London SW7 2BT, UK
c.visavakul@ic.ac.uk, p.cheung@ic.ac.uk, wl@doc.ic.ac.uk

Abstract. This paper presents a design for combining reconfigurable multiplier array known as Flexible Array Blocks (FABs) and digit-serial techniques to implement arbitrary size multipliers with limited resources. Any $4M$x$4N$ bit multipliers can be implemented. In-depth evaluation of the tradeoff between resources and performance is presented. The resulting design is suitable for embedding in heterogeneous FPGA structures for fixed point DSP applications.

1 Introduction

FPGAs are commonly used in many DSP and video processing applications where multiplication forms one of the most common operations. Whilst existing FGPA architectures are optimised for binary addition, configuring FPGAs for binary multiplication is much less efficient. Haynes and Cheung reported the design of a reconfigurable multiplier array (known as Flexible Array Blocks or FAB) that can improve area efficiency by more than one order of magnitude [1]. A FAB is a 4x4 bit reconfigurable building block formed from a regular adder array structure that supports both signed and unsigned representations. Any $4M$x$4N$ bit parallel multiplier can be implemented by cascading together MxN FABs as a two dimensional array.

Although the proposed design improves area and time efficiency, it suffers a major limitation on flexibility. If a required multiplier cannot be fitted into the available on-chip FAB resources, it cannot be implemented. There are no possibility for trading off resources with multiplication time. This paper proposes a solution to this problem by exploiting digit-serial techniques to provide the necessary tradeoff between number of clock cycles needed to perform a multiplication and the number of FAB resources used. The resulting structure, known as DigiFAB, can be configured to implement any $4M$x$4N$ multiplier with any amount of FAB resources. An in depth quantitative study of the tradeoff between hardware and performance is also made.

This paper is organised as follows. In Section 2 a slight modification to the origin FAB structure is introduced. In section 3, the detail design of DigiFAB, a digit-serial version of the FAB structure, is described. An in depth evaluation of the the area-time tradeoff of DigiFAB is presented in Section 4. Section 5 concludes the paper.

2 Reduced Flexible Array Block

The original FAB structure [1] was designed as a 4x4 bit multiplier array configurable to form larger signed 2's complement or unsigned multipliers. This can be simplified to a Reduced Flexible Array Block, or RFAB (Figure 1), by removing the partial product adder which is only used if the FAB is located on the right-most column of the 2-dimensional array. A $4M$x$4N$ multiplier can be constructed by simply connected MxN reduced FABs together in a 2-dimensional array as shown in Figure 2[1].

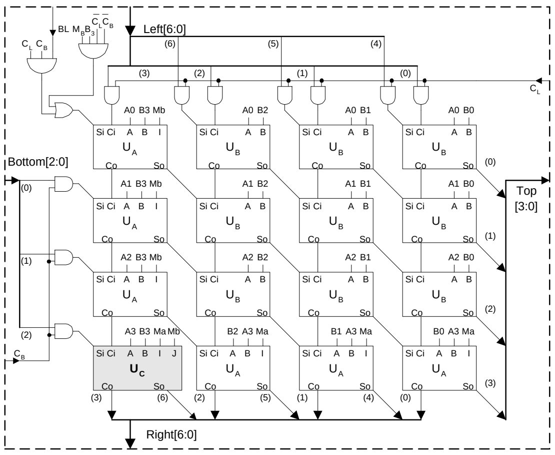

Fig. 1. Reduced Flexible Array Block (RFAB)

3 Digit-Serial Multiplier using FABs

Consider the case where instead of having the resource of MxN FABs on-chip, only KxL FABs are available[2]. To implement a 4Mx4N multiplier, the 2-dimensional FAB array can be divided into tiles of KxL FABs as shown in Figure 3. A digital-serial implementation can then be realised by mapping a single KxL FAB cluster to each tile on successive clock cycles. This can either be done row first or column first. In this way, tradeoff is made between area and performance (i.e. the number of clock cycles).

Unfortunately this tradeoff is not direct, nor is it simple. Additional registers, multiplexers and control logic have to be added to the KxL FAB cluster

[1] Hereafter the reduced form of FAB (RFAB) is used whenever FAB is mentioned.
[2] To simplify the discussion, we assume for the moment M/K and N/L are integers

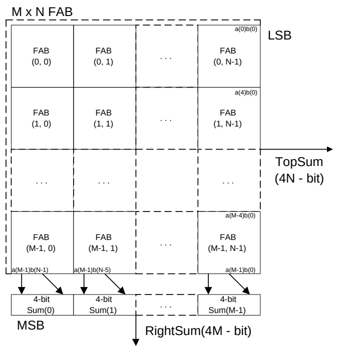

Fig. 2. $4M$x$4N$ bit multiplier implemented with MxN FABs

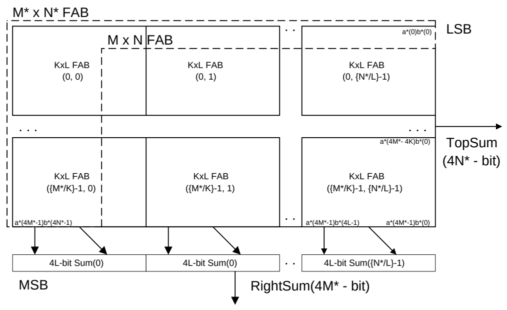

Fig. 3. $4M$x$4N$ bit multiplier implemented with tiles of KxL FABs

in order to store the partial results. Figure 4 shows the basic structure of Digi-FAB, a DIGIt-serial implemenation of multiplier using FABs. All the peripheral circuits surroudning the KxL FAB cluster are overheads due to the digit-serial implementation.

This is complicated further when M and N are not divisible by K and L respectively. In which case, the DigiFAB is configured to implement a $4M^*$x$4N^*$ multiplier where,

$$M^* = \lceil M/K \rceil \text{x} K$$
$$N^* = \lceil N/L \rceil \text{x} L$$

The number of extra registers needed to implement DigiFAB is given in Table 1.

FAB	Top Registers		Right Registers		Mux Registers	
	width	level	width	level	width	level
1x1	4	1	7	N	1	N
KxL	4K	1	8L-1	N*/L	1	N*/L

Table 1. Summary of registes usage in DigiFAB

3.1 Factoring of FAB clusters

Given the hardware resource of W FABs, there are many ways to factorise it such that $W = K$xL. For example 10 FABs can be employed to implement 4 different DigiFAB cluster configurations: 1x10, 2x5, 5x2 and 10x1. Table 2 shows the possible clustering arrangement for W= 9 to 12.

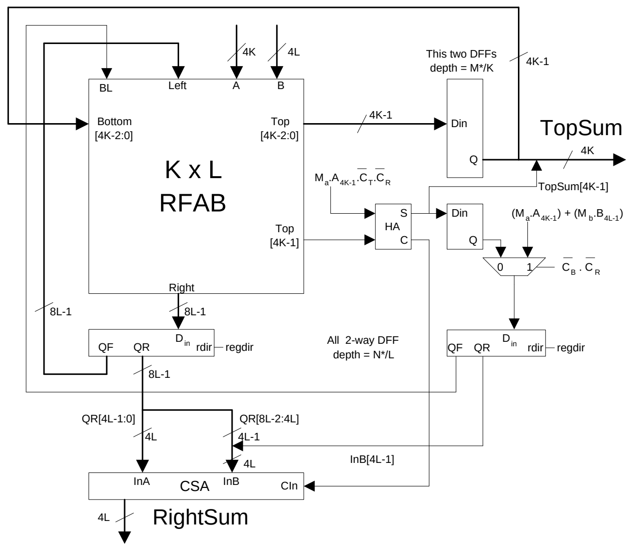

Fig. 4. DigiFAB structure

RFABs (X)	Factors	Possible Configurations (KxL)
12	1 2 2 3	1x12, 2x6, 3x4, 4x3, 6x2, 12x1
11	1 11	1x11, 11x1
10	1 2 5	1x10, 2x5, 5x2, 10x1
9	1 3 3	1x9, 3x3, 9x1

Table 2. Examples of possible clustering of 9 to 12 FABs

4 Evaluations and Results

In this section, the tradeoff between area and speed of DigiFAB is presented in two ways. Firstly DigiFAB using only 4x4 cluster is compared with a FAB-only implementation. Secondly the effect of using different cluster configurations on area and multiplication times is investigated for a range of FAB resources.

4.1 Area and Speed tradeoff for 4x4 cluster

Parametric models for the hardware complexity of the FAB-only and the DigiFAB implementations are found as:

$$\begin{aligned} Tr_{FAB} &= 836MN \\ Tr_{DigiFAB} &= 654KL + 88K + 96L + 272N + 29M^*/K + 21N^*/L + 256 \end{aligned}$$

The area utilization is measured in terms of number of transistors, and the circuits are mapped to the Alliance standard cell library [2].

Since area estimate for DigiFAB includes estimates for all the control circuitry necessary for its proper function, its model contains many more parameters when compared with the FAB-only implementation.

Figure 5 shows the transistor counts for implementing a NxN multiplier using DigiFAB with a single 4x4 cluster, and compares it to a FAB-only implementation. As expected the transistor count for the FAB-only implementation increases as N^2 while the DigiFAB size remains substantially constant. This is in spite of the extra registers required to store partial results and the additional overhead added by the control circuitry.

The timing model is also based on the Alliance CMOS cell library. The multiplication time is dependent on two factors: the worst-case time delay on the combinatorial circuit and the number of clock cycles required to complete a multiplication. The FAB-only implement requires only a single clock cycle to compute the product, but it contains long delay paths through both sum and carry chains. The DigiFAB, as with all digital-serial approaches [3], allows much shorter clock period to be used. However it also requires $\{\lceil M/K \rceil + 1\} * \lceil N/L \rceil$ cycles to complete a computation. The total computation time used for the evaluation is computed by multiplying the shortest clock period with the number of clock cycles required.

Figure 6 shows that computation time of the FAB-only design increases linearly with N, while the DigiFAB implementation has a step like computation time due to the quantized nature of the circuit.

4.2 Effects of Different Clustering Arrangements

So far only DigiFAB using 4x4 cluster of FAB cells has been considered. The same resource ($W = 16$) could also have been configured as 1x16, 16x1, 8x2, and 2x8. Which configuration would give the best area-speed tradeoff?

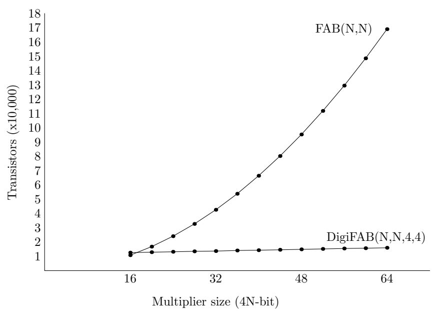

Fig. 5. Area comparison between FAB and DigiFAB with 16 RFABs

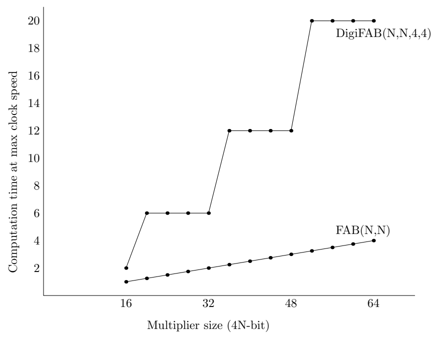

Fig. 6. Computation time comparison between FAB and DigiFAB with 16 RFABs

Figures 7 and 8 show respectively the relative area and compute time of all possible configurations for different values of W when compared with a FAB-only implementation of a $32x32$ bit multiplier. For example, if 16 FABs are used in a DigiFAB structure, only around 30% of hardware resources is required, but the compute time would increase by a factor of between 4 to 26, depending on the cluster arrangement. In other words, how W is factored into KxL hardly affects the amount of area saving, but can have significant effect on compute time. It can also be seen that having a cluster configuration that is essentially square provides the best arrangement (i.e K and L are as close as possible). In contrast the long-and-thin, and the short-and-fat configuration are worst (i.e. Wx1 and $1xW$).

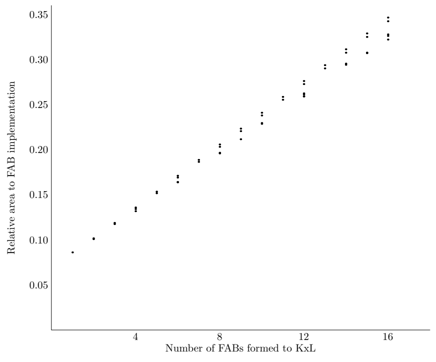

Fig. 7. Relative area to FAB-only implementation for building 32x32 bit multiplier using DigiFAB with 16 or less RFABs

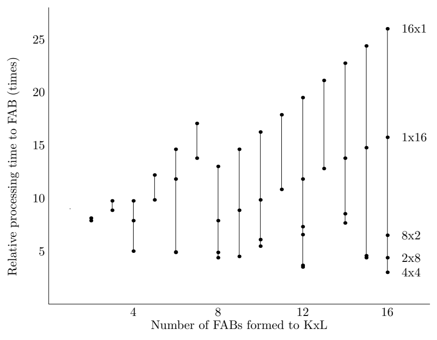

Fig. 8. Relative calculation time to FAB-only implementation for building 32x32 bit multiplier using DigiFAB with 16 or less RFABs

These two graphs can be combined to give a direct resource vs area vs compute time tradeoff for the different cluster configurations as shown in Figure 9. Each line represents a fixed amount of FAB resource. Every point on this graph is a possible solution to the problem of implementing a 32x32 bit unsigned/signed multiplier. Figure 10 shows a related tradeoff characteristic. Instead of showing the total compute time on the x-axis, this uses the maximum operating clock frequency relative to a FAB-only implementation. It can be seen that if DigiFAB is to be used in an embedded array with fast clocking, one would choose a small value for W.

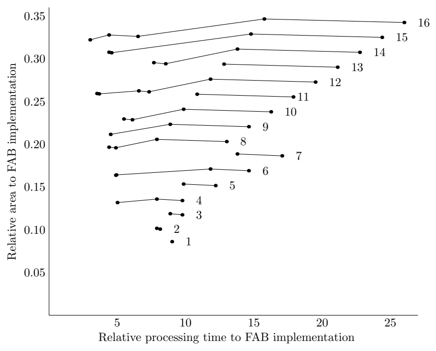

Fig. 9. Relative area VS relative computational time for different DigiFAB configurations

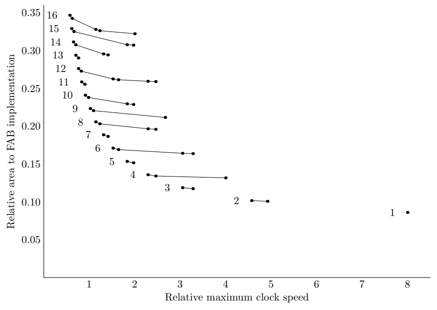

Fig. 10. Relative area VS maximum clock speed for different DigiFAB configurations

5 Conclusion

In this paper, a digit-serial version of the original FAB reconfigurable multiplier is presented. It has been shown that by exploiting digit-serial techniques, a flexible tradeoff between array resource, area and compute time is available. It has also been demonstrated that although a given resource could be configured in many different ways, the optimum clustering configuration is one which is essentially as square as possible. The DigiFAB design also allosw any size of multiplier to be implemented with any fixed amount of FAB resource.

Many questions remain unanswered. What is the appropriate mixture of FAB cells and conventional LUT-based logic cells? How should the routing to FABs be organised and how much? However, results from this paper suggest that digit-serial FABs may provide sufficient advantages to warrant efforts to be devoted to answering these questions.

References

1. Simon D. Haynes, Peter Y. K. Cheung, “A Reconfigurable Multiplier Array For Video Image Processing Tasks, Suitable For Embedding In An FPGA Structure”, in *Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 226-234, 1998.
2. A. Greiner, F. Pêcheux, “ALLIANCE: A complete set of CAD tools for teaching VLSI design”, 3^{rd} *Eurochip Workshop on VLSI Design Training*, Grenoble, pp. 230–237, 1992.
3. Yun-Nan Chang, Janardhan H. Satyanarayanan, Keshap K. Parhi, “Systematic Design of High-Speed and Low-Power Digit-Serial Multipliers”, *IEEE Trans Circuits and Syst.–II Analog and Digital Signal Processing*, vol. 45, no. 12, December 1998.