A Bit-Serial Cell for Reconfigurable Hardware

Mitchell J. Myjak and José G. Delgado-Frias
School of Electrical Engineering and Computer Science
Washington State University
Pullman, Washington 99164-2752 USA
Email: {mmyjak, jdelgado}@eecs.wsu.edu

***Abstract*— This paper introduces a novel bit-serial cell for reconfigurable hardware used to perform digital signal processing. The cell contains an array of 4-bit lookup tables, or "elements", that can operate in two modes. In memory mode, the elements behave as a random-access memory. In mathematics mode, the elements perform operations such as multiply-accumulate, addition, and shifting in bit-serial fashion. To calculate m-bit functions, the cell requires $(m+1)$ elements and $(2m+1)$ clock cycles. Layout simulations in 180-nm CMOS demonstrate that the serial clock frequency approaches 2 GHz. Compared to a parallel implementation with the same functionality, the cell has lower throughput but substantially smaller area.**

I. Introduction

Reconfigurable hardware has become a well-accepted option for implementing complex processing stages without incurring the development costs of custom integrated circuits [1]. In particular, digital signal processing (DSP) greatly benefits from the performance acceleration over microprocessor-based implementations [2]. However, DSP is computationally intensive: many algorithms apply a number of multiplications and additions to a large data set. Taking advantage of this parallelism, researchers have proposed new coarse-grain reconfigurable architectures for DSP [3]. Such devices typically contain an array of cells that perform 16-bit or 32-bit operations. This approach contrasts with the 1-bit granularity of a field-programmable gate array (FPGA).

Designing reconfigurable hardware for DSP involves basic tradeoffs between performance, area, and flexibility. Coarse-grain cells can execute calculations quickly but offer limited functionality. Fine-grain cells can be combined into many different modules but require a complex interconnection structure. To solve this dilemma, we have proposed a two-level architecture in which each cell contains a number of smaller reconfigurable units, called "elements" [4]. The elements within a cell need only assume a few structures to implement all basic DSP operations, including multiplication, addition, control logic, and memory access. Hence, the design offers less overhead than fine-grain devices, while retaining much of the low-level flexibility.

In the original two-level architecture, the cell performed word-length operations using a small matrix of elements. However, computing binary arithmetic in a bit-serial fashion greatly reduces the circuit complexity—particularly for multiplication, which forms the bottleneck of most DSP algorithms. Bit-serial designs also require fewer data lines and hence fewer interconnection resources on reconfigurable hardware. Recognizing these benefits, researchers have proposed new reconfigurable architectures that perform bit-serial computations. Both fine-grain [5], [6] and coarse-grain [7] devices appear in the literature. The drawback of these designs is that the entire device runs off a single clock signal. Aside from the problem of clock distribution, the interconnection structures within the critical path limit the maximum clock frequency and hence the processing speed.

This paper introduces a novel bit-serial cell for reconfigurable DSP hardware. As described in Section II, the design uses the two-level scheme developed previously to minimize interconnection structures within the cell. Section III demonstrates that the VLSI implementation achieves high clock frequency and compares the simulated performance with the parallel architecture. Finally, Section IV concludes the paper.

II. Design

The bit-serial cell performs n-bit operations using an array of $n+1$ reconfigurable elements. Each element is simply a 16×2-bit random-access memory. The array of elements can only assume two structures: one optimized for bit-parallel memory access and the other optimized for bit-serial arithmetic. The remainder of this section describes the two modes of operation and demonstrates how the design can implement several common operations. For the purposes of this discussion, assume that $n = 4$.

A. Memory Mode

Fig. 1 depicts the cell in memory mode. The elements implement a random-access memory with address $a_{3:0}$, input data $i_{4:0}$, and output data $q_{4:0}$. The memory is organized into two banks of 16 words each; each element stores a 1-bit slice of each bank. The $op_{2:0}$ input specifies the operation performed on the memory. One unique feature of the design is the ability to read from one bank and write to the other bank simultaneously. This capability helps multi-stage DSP algorithms work with large data sets. For example, the Fast Fourier Transform (FFT) can load a pair of samples from bank 0 and store the previous results in bank 1. The next stage can then read from bank 1 and write to bank 0.

Besides storing intermediate data, memory mode is useful for specifying read-only lookup tables and implementing logic functions with up to five inputs. Additionally, all cells default to memory mode during reconfiguration so that the system can write new data into the elements.

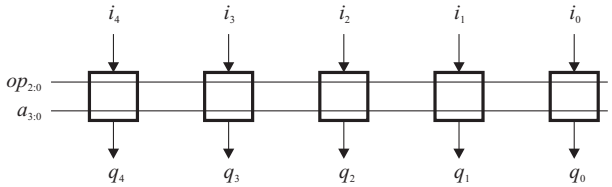
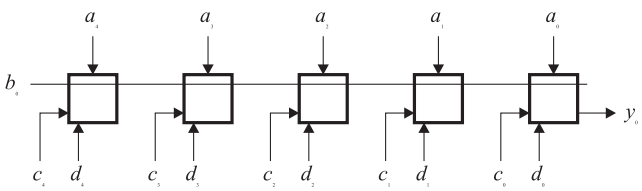
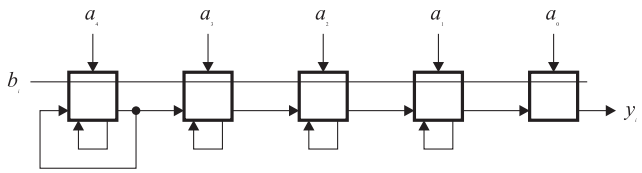


Fig. 1. Memory mode.



(a) Initialization.



(b) Execution (i = 1, 2, ..., 8).

Fig. 2. Mathematics mode.

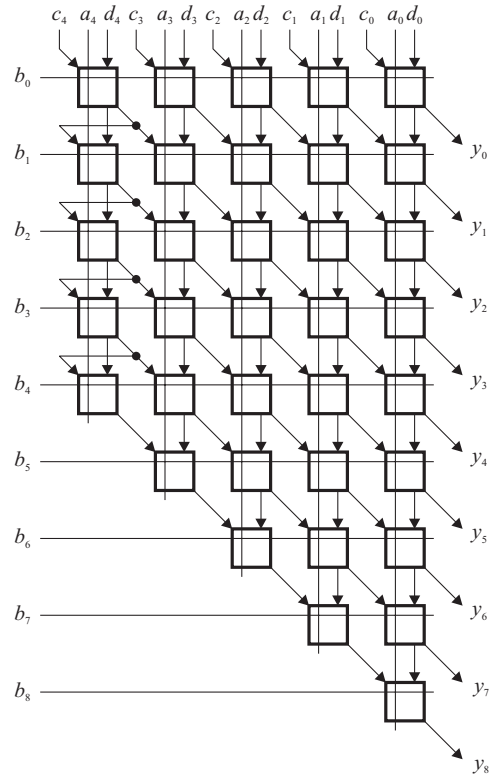


Fig. 3. Expanded design of cell in mathematics mode.

B. Mathematics Mode

Fig. 2 illustrates the set of elements in mathematics mode. The elements now operate in a bit-serial fashion with inputs $a_{4:0}$, b_i, $c_{4:0}$, $d_{4:0}$, and output y_i. Here, b_i and y_i are in bit-serial format; all other terms are in parallel format. Each computation consists of an initialization phase and an execution phase. During initialization, the cell applies all four inputs to the elements to generate the first bit of the output, y_0. During execution, the cell takes away the $c_{4:0}$ and $d_{4:0}$ inputs and connects the elements in a chain. The elements run for nine cycles to produce the remaining bits of the output, $y_{1:8}$.

To clarify how the cell operates in mathematics mode, Fig. 3 expands the chain of elements into an equivalent parallel model. The memory inside each element acts as a lookup table with four input bits and two output bits. The outputs feed back into the inputs in the next cycle. The system loads the contents of the lookup tables with the desired function during reconfiguration. For almost all DSP operations, each element in the cell contains the same lookup table, simplifying the reconfiguration process.

It is no coincidence that the parallel model of mathematics mode closely resembles a carry-save multiplier [8]. In fact, the structure can perform 4-bit multiply-accumulates (MAC) of the form

$$y_{7:0} = (a_{3:0} \times b_{3:0}) + c_{3:0} + d_{3:0}. \quad (1)$$

This function allows the reconfigurable device to implement $4n$-bit multipliers with an $n \times n$ array of cells [9]. At first, it would seem that only four elements would be necessary to compute a 4-bit MAC. However, using an additional element allows modules to work with data in unsigned and two's-complement format. The extra bit on the inputs and outputs prevents ambiguity between the two number representations.

The cell in mathematics mode can compute many other functions besides multiplication. Clearly, addition and subtraction are special cases of (1) with $b_{3:0}$ set to 1 or -1. The structure can also implement a bit shifter. Referring to Fig. 3, $b_{8:0}$ carries the input data and $a_{4:0}$ specifies the number of places to shift using one-hot encoding. The column with $a = 1$ copies the value of b diagonally right toward the output. The remaining columns simply pass the data in the same direction. With appropriate values of $a_{4:0}$, the cell can copy any 5-bit substring of $b_{8:0}$ to the upper significant bits of the output, $y_{8:4}$. The extra bit y_8 can be zeroed if desired.

III. Implementation

This section complements the discussion of the bit-serial cell by describing its transistor-level implementation. Essentially, the entire design consists of several lookup tables with some glue logic. This strategy leads to a simple and compact implementation that can be optimized for high performance and low power consumption. The following discussion focuses on the operation of each element in memory mode and mathematics mode, and then compares the simulated performance to the original parallel architecture.

A. Memory Mode

In memory mode, the elements act as a random-access memory. Each of the two banks is organized as a 4×4 array

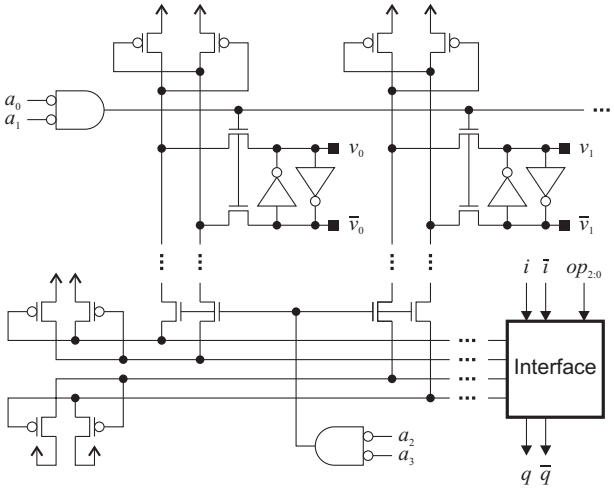


Fig. 4. Implementation of memory mode.

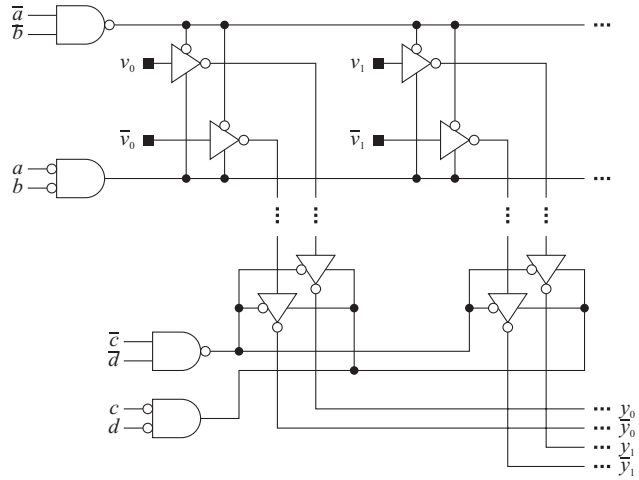


Fig. 5. Implementation of mathematics mode.

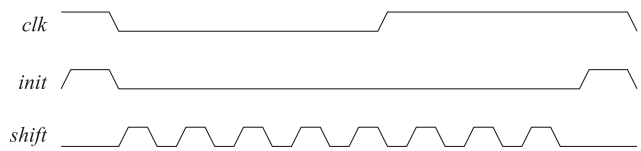


Fig. 6. Clock waveforms for mathematics mode.

of latches with some simple glue logic. The lower two bits of the address $a_{3:0}$ enable one row, and the upper two bits enable one column. The $op_{2:0}$ input specifies the operation performed on each bank. Fig. 4 depicts the datapath in memory mode.

Suppose the element needs to read from bank 0 and write to bank 1. The element begins by performing a read operation on both banks. A row decoder enables the latches in row 0, and a column decoder connects the data lines in column 0 to the outputs. The outputs of each bank are represented as two complementary lines. The n-transistors in each selected latch discharge one of the lines to ground. External cross-coupled p-transistors raise the other line to V_{DD}.

The output data connects to an interface module controlled by the $op_{2:0}$ input. After allowing some time for the read operation to complete, the interface module copies the value read from bank 0 to the q and $\overline{q}$ outputs. Next, the module drives the i and $\overline{i}$ inputs back into the selected latch in bank 1. The n-transistors in the datapath now run in reverse, overwriting the value stored in v_1 and $\overline{v_1}$. The read and write operations are synchronized to the clock signals used in mathematics mode, described next.

B. Mathematics Mode

To achieve maximum performance, the element uses a separate datapath for mathematics mode. Fig. 5 illustrates this implementation. Now all input and output bits are represented as two complementary lines. The a and b inputs control a row decoder, and the c and d inputs control a column decoder. The decoders drive a series of gated inverters that copy the values stored in the selected latches to the y_0 and y_1 outputs. Bank 0 determines y_0, and bank 1 determines y_1.

During the initialization phase of mathematics mode, the current inputs and previous outputs are collected in parallel form. The reconfigurable device takes advantage of this fact by pipelining the datapath into n-bit portions. Each pipeline stage encompasses one operation of the cell, or one data transfer across the interconnection structure. This paper assumes that the overall clock rate depends only on the cell, as the interconnection structure can always transmit data in parallel. To control the pipeline stages, the device distributes a master clock signal clk to every cell. Each cell must divide the clock period into an initialization phase and $2n+1$ serial processing stages. Fig. 6 depicts the resulting control signals $init$ and $shift$. Note that the initialization phase occurs at the end of the global clock cycle, so that each operation spans part of two cycles. This feature allows clk to be slowed down for testing.

The clocking scheme used in the proposed cell achieves higher performance than other bit-serial reconfigurable architectures. The frequency of the serial clock $shift$ does not depend on the propagation delay though complex interconnection structures, but rather the latency of the elements in mathematics mode. To increase performance further, the cell uses differential control signals that drive dual n-type and p-type transmission gates. As described in [10], the clock generator contains a dual ring oscillator with cross-coupled p-transistors on the outputs to ensure that the waveforms are symmetric. Generating the control signals locally avoids many of the clock distribution problems shared by other designs.

C. Simulations

To evaluate the functionality and performance of the proposed implementation, we performed layout simulations in 180-nm technology with all parasitic capacitances. The results for mathematics mode appear in Fig. 7. The cell has been configured to act as a 4-bit MAC unit. In the simulation, the elements perform the following calculation:

$$(01111 \times 01010) + 01010 + 01010 = 010101010. \qquad (2)$$

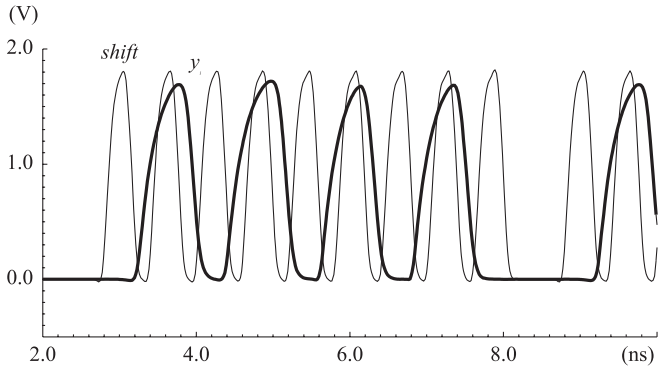


Fig. 7. Simulation of mathematics mode.

TABLE I
PERFORMANCE COMPARISON

Parameter	4-bit Parallel	4-bit Serial	8-bit Serial
Number of elements	16	5	9
Granularity	4-bit	4-bit	8-bit
Estimated area	0.040 mm^2	0.017 mm^2	0.033 mm^2
Memory per cell	512 bits	128 bits	256 bits
Configuration (math)	512 bits	38 bits	38 bits
Clock period	4 ns	6 ns	12 ns

These values represent a worst-case situation, since the bit-serial output y_i alternates between logic 0 and logic 1. As shown, the elements produce the correct output value. The shift signal $shift$ runs at a frequency of approximately 2 GHz; the master clock clk has a period of 6 ns.

D. Performance Comparison

The proposed bit-serial cell offers a reasonable alternative to the parallel design we developed previously in [4]. Table I compares the performance, estimated area, and overhead of the two approaches. The original cell used a 4×4 matrix of elements to compute 4-bit functions. Like the bit-serial scheme, the elements could also implement a random-access memory. The cell could store 128 words in memory mode (although two words were accessed simultaneously). However, the system had to program the lookup table inside each element before running in mathematics mode. An optimized implementation of the parallel design in 180-nm technology ran with a period of 4 ns.

Table I lists two alternatives of the bit-serial design. The first uses five elements to compute 4-bit functions, and the second uses nine elements to compute 8-bit functions. Both cells require fewer elements than the original design and thus save area. However, DSP algorithms that need large memory modules would need more cells to implement the same address space. The bit-serial designs also require fewer configuration bits in mathematics mode. Since the elements almost always specify the same lookup table, the system can write to every element simultaneously. The parallel design does offer a higher clock rate, making this alternative better for applications that cannot trade off performance. Notice, however, that a given die size could contain more bit-serial cells, offering designers more opportunities for parallelization.

IV. CONCLUSION

In this paper, we have introduced a novel bit-serial cell for coarse-grain reconfigurable architectures. The cell uses an array of $(n+1)$ elements to perform n-bit computations. The extra element allows the cell to operate with unsigned and two's-complement data formats. The cell can operate in only two modes: one optimized for memory access and the other for binary arithmetic. Together, the two modes allow the cell to perform the entire spectrum of operations required by DSP algorithms with low hardware complexity. We presented a transistor-level implementation of the cell as well as layout simulations that verify the performance of the scheme. The serial clock reaches a frequency of approximately 2 GHz in 180-nm technology.

ACKNOWLEDGMENT

M. Myjak is supported by a graduate fellowship from the U.S. Department of Homeland Security (DHS). The DHS Scholarship and Fellowship Program is administered by the Oak Ridge Institute for Science and Education (ORISE) for DHS through an interagency agreement with the U.S. Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORISE.

REFERENCES

[1] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, Jun. 2002, pp. 171–210.

[2] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: a survey," in *Programmable Digital Signal Processors*, Y. Hu, ed., Marcel Dekker Inc., 2001.

[3] R. Hartenstein, "Coarse grain reconfigurable architectures," in *Proc. 6th Asia South Pacific Design Automation Conf.*, Yokohama, Japan, 2001, pp. 564–570.

[4] M. Myjak and J. Delgado-Frias, "A two-level reconfigurable architecture for digital signal processing," in *Proc. 2003 Int. Conf. on VLSI*, Las Vegas, NV, Jun. 2003, pp. 21–27.

[5] T. Isshiki et al., "High density bit-serial FPGA with LUT embedding shift register function," in *Proc. 2002 Asia-Pacific Conf. on Circuits and Systems*, vol. 1, Oct. 2002, pp. 475–480.

[6] S.A. Rahim and L.E. Turner, "A field programmable bit-serial digital signal processor," in *Proc. 4th IEEE Int. Workshop on System-on-Chip for Real-Time Applications*, Jul. 2004, pp. 295–298.

[7] A. Tisserand, P. Marchal, and C. Piguet., "An on-line arithmetic based FPGA for low-power custom computing," in *Proc. 9th Int. Workshop on Field Programmable Logic and Applications*, London, England, vol. 1673 of LNCS, Sep. 1999, pp. 264–273.

[8] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd ed., Upper Saddle River, NJ: Pearson Education, Inc., 2003, pp. 591–592.

[9] M. Myjak and J. Delgado-Frias, "Pipelined multipliers for reconfigurable hardware," in *Proc. 11th Reconfigurable Architectures Workshop*, Santa Fé, NM, Apr. 2004, pp. 150–154.

[10] M. Myjak and J. Delgado-Frias, "A symmetric differential clock generator for bit-serial hardware," in *Proc. 2005 Conf. on Computer Design*, Las Vegas, NV, Jun. 2005, in press.