

# A Reconfigurable Data Path Processor for Space Applications

Gregory W. Donohoe, K. Joseph Hass, Stephen Bruder,\* Pen-Shu Yeh\*\*

University of New Mexico, Institute of Advanced Microelectronics

\* New Mexico Institute of Mining and Technology

\*\* NASA Goddard Space Flight Center

## Abstract

This paper introduces a reconfigurable processor under development, and targeted to an ultra-low-power, radiation tolerant CMOS process. The architecture has been selected to implement a synchronous pipeline computational model. The highly-parallel, reconfigurable data path emphasizes agile data flow instead of agile control flow for data-intensive, streaming processing for spacecraft.

## I. INTRODUCTION

The Reconfigurable Data Path Processor (RDPP) is conceived as an ultra-low-power, radiation-tolerant processor for data-intensive, streaming applications aboard spacecraft. The RDPP will be an integrated circuit, implemented in the ultra-low-power, radiation-tolerant CMOS technology being developed at the Institute of Advanced Microelectronics. The requirements of the RDPP are that it yield high performance on selected applications with low power consumption, and that it be versatile, radiation tolerant, and designer friendly. Radiation tolerance and low power are addressed through the choice fabrication process, performance is achieved through parallelism, and versatility comes from the ability to reconfigure the hardware. Support software is being developed to simplify designing with the RDPP. The processor will be extensible to larger problems both spatially and temporally. The design will permit multiple instances of the RDPP to be tiled onto a circuit board, thus achieving extensibility in space. Alternatively, the design will enable partial results to be stored in memory and later retrieved, giving extensibility in time. Aboard spacecraft, fine-grained reconfigurability allows a component to be reconfigured to perform different functions at different stages of a mission, saving on size, weight and power [2, 7].

The RDPP architecture is based on a configurable data path, using a synchronous data flow model. We can classify reconfigurable architectures according to granularity, as shown in Figure 1.. The conventional von Neumann processor has minimal granularity, with a single arithmetic-logic unit (ALU) through which all data must pass. Programmable logic

devices, such as Field Programmable Gate Arrays (FPGAs) have maximum granularity, as they are programmable at the gate level. Complex Programmable Logic Devices (CPLDs) are slightly less granular than FPGAs, while digital signal processors (DSPs) and super-scalar CPUs are more granular than simple microprocessors.

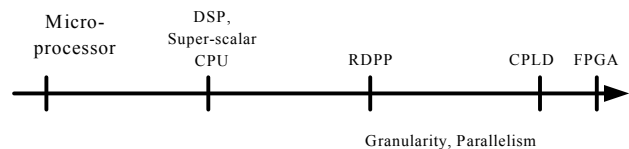


Figure 1: Granularity spectrum of reconfigurable devices.

Fine-grained reconfigurable granularity offers great flexibility. It enables the architecture of the processor to be modified to closely match the architecture of the computation problem, offering the possibility of very high performance. However, fine-grained reconfigurability exacts a high price in area and performance. It is estimated that only 1% of the area of a typical FPGA is available for useable logic [1]; the rest is consumed in interconnect and configuration memory. Complex modules synthesized from existing gates in FPGAs cannot take advantage of the circuit- and layout-level optimizations possible when these modules are designed by hand. The RDPP falls between the DSPs and the CPLDs, with multiple, configurable processing elements connected by a configurable data path. A coarser granularity enables run-time reconfiguration, and permits processing elements to be optimized at the circuit and gate levels, yielding higher performance.

## II. IMPLEMENTATION TECHNOLOGY

The RDPP is being developed for implementation in Ultra-Low-Power, Radiation Tolerant CMOS using an AMI 0.35 $\mu$  process. Two of the research thrusts of the Institute of Advanced Microelectronics are Ultra Low Power (ULP) CMOS VLSI chips and Radiation Tolerant (RT) devices

produced with commercial fabrication processes [4]. In conventional CMOS circuits, the dominant source of energy consumption is dynamic power, the power dissipated when the device is switching; it varies with the square of the supply voltage. The ULP program achieves two-order-of-magnitude power reduction over conventional (3.3V) CMOS by operating at very low supply voltages, i.e., ½ volt. Control of the switching threshold is managed through dynamic back-bias techniques.

A fortuitous consequence of the ULP technology is that, according to tests performed so far, these devices are naturally tolerant to Total Ionizing Dose (TID) radiation effects, achieving a total dose tolerance of 100 to 300 KRads. To achieve Single Event Upset (SEU) immunity, the IA $\mu$ E employs 12-transistor SEU-resistant memory cells called Whittaker Cells [6]. Single Event Latchup (SEL) immunity is achieved through layout techniques. Currently, the ultra-low-power and radiation-tolerant technologies are being combined into a ULP/RT technology.

The choice of ULP/RT technology for the RDPP affects the design parameters. Low-power design in conventional CMOS seeks to limit dynamic power dissipation by minimizing the amount of switching activity. This encourages memory-intensive, “vertical” designs, with little parallelism and no wasted switching events. A standard approach to low-power CMOS design is to turn off unused modules by blocking the signals to them; clock gating, or blocking the clock signals to unused parts of the chip, is a common example.

The ULP/RT technology seeks to achieve a balance between switching power and static power. While dynamic power is significantly reduced, static power dissipation, due to leakage in parasitic source and drain diodes, is proportionally more significant. Static power dissipation is independent of the amount of switching activity, but is proportional to the source and drain area, and so increases with the number of transistors in the circuit. To maximize the power savings in the ULP technology, we avoid designs incorporating large numbers of idle transistors, such as memories. Instead, we seek performance through parallelism in the data path, and strive to keep every transistor busy.

### III. ARCHITECTURE

The development of the architecture is proceeding in three phases. First, we must define a computational model. Next, we define the architectural components: processing elements, memory, interconnects, and control strategy. Finally, we model the architecture on sample problems through HDL simulations.

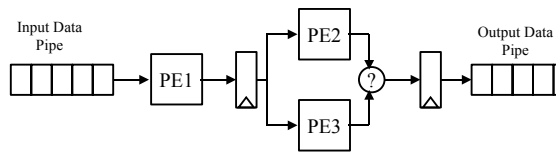
#### A. Computational Model

In the words of Edward Lee of the University of California at Berkeley [5], “Applications are built on a model of computation, whether the designer is aware of this or not.” The most familiar computational model is the sequential processor, which has an infinitely large, randomly-addressable memory, a single arithmetic-logic unit (ALU), and a control unit. The memory stores information and instructions, and the ALU transforms bits patterns; a control unit reads data and instructions from memory, and routes data through the ALU, and back into memory. This computational model is deeply embedded in programming languages such as C and Matlab. Time is not explicit in this model, only sequence. When the computer executes a function, such as  $\sin(x)$ , the main flow of execution stops; the  $\sin()$  function is executed until it terminates, and the main program flow resumes where it left off. Thus, program agility is achieved by changing the flow of execution.

Embedded systems, such as those used in control and signal processing applications aboard spacecraft, must deal explicitly with time. For historical and economic reasons, embedded systems designers are forced to adapt sequential processors into this time-sensitive model. A multi-tasking system then creates the illusion of concurrency by switching rapidly between tasks. Architectural options for increasing the performance of these systems are to increase the clock frequency, to introduce concurrency through pipelined instruction and data paths, and to speed up memory access with multilevel cache. For a given implementation technology, the speed improvements are necessarily limited.

#### B. RDPP Architecture

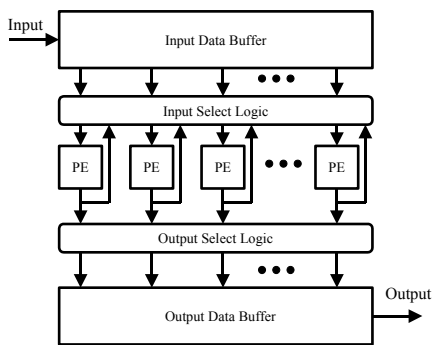
The RDPP takes a different approach, based on a synchronous pipeline model. In this model, multiple processing elements are connected in a network, and data and control information flow between them. Execution agility is achieved, not through conditional branching of the execution path, but through conditional switching of data paths.



**Figure 2: example of conditional data transfer in a pipelined processor.**

The example in Figure 2 has three processing elements, separated by clocked registers. Data flows from left to right. PE2 and PE3 receive the same input data, but perform different computations on it. A conditional switch, indicated by the question mark, decides which one of the two results is passed on downstream. Thus, this network makes decisions by means of conditional switching of data paths instead of through conditional branching.

The RDPP architecture, shown in Figure 3, consists of an array of configurable processing elements linked to each other, and to input and output buffers, through a flexible switching network. The evolution of a computation in the RDPP consists of setting up a pipeline, executing it until completion, then setting up a different pipeline with the same components, executing it, etc.



**Figure 3: RDPP Architecture.**

#### IV. APPLICATION EXAMPLES

Some typical data-intensive spacecraft applications are digital filters, pixel readout correction, hyperspectral image data conversion, and object detection and tracking. The processor may be required to operate on four kinds of data: (1) sensor signal data; (2) address data; (3) state information, such as pixel labels; and (4) status information, such as “done” signals.

This first example is a finite impulse response (FIR) filter, which is a natural application for pipelined processors. The

$$\text{output } y_k \text{ at sample time } k \text{ is given } y_k = \sum_{i=0}^3 a_i x_{k-1},$$

where  $x_k$  is the input at sample time  $k$ . The filter coefficients

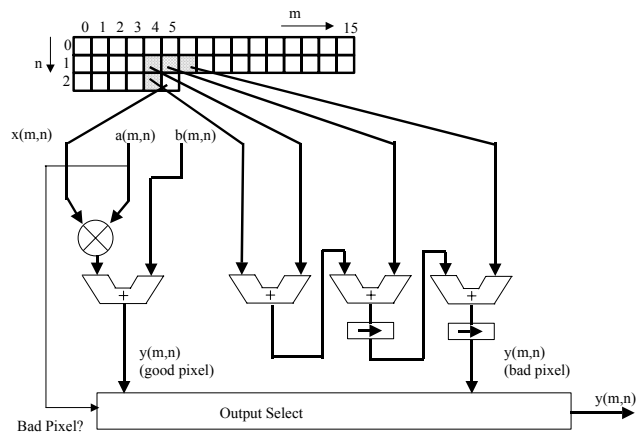


$a_i$  are stored in registers in the processing elements. The input samples are delayed in the input data buffer.

**Figure 4: Finite Impulse Response Filter Example**

For this four-tap example, the delay elements are implemented in the input buffer memory. The filter weights are contained in registers local to each processing element.

Sensor nonuniformity correction illustrates the use of conditional switching. Imaging focal plane arrays typically exhibit pixel-by-pixel variation due to manufacturing tolerances; in particular, each pixel has a brightness offset due to leakage or dark current, and a gain variation. To obtain accurate data, the sensor must be calibrated. In a calibration phase, we estimate the offset and gain factor for each pixel, storing these in a memory. In operation, we correct each pixel by multiplying by its corrective gain factor, and adding its corrective offset.



### Figure 5: Pixel readout correction example

If  $m$  and  $n$  represent the horizontal and vertical coordinates, and  $x(m, n)$ ,  $a(m, n)$  and  $b(m, n)$  are the input pixel and the corresponding gain and offset correction parameters, then the corrected output pixel is computed from  $y(m, n) = a(m, n)x(m, n) + b(m, n)$ . Sometimes, however, a pixel is simply "dead", rendered non-responsive due to radiation or some other failure mechanism. In this case, the pixel value is commonly replaced by a spatial average of its neighbors. In the RDPP, we compute both the "good pixel" and "bad pixel" cases simultaneously. Two processing elements read the gain and offset values and correct each incoming pixel. At the same time, three more processing elements compute the spatial average of three neighbors. In the example above, a bad pixel is replaced by:

$$y(m, n) = \frac{1}{4}(x(m-1, n-1) + x(m, n-1) + x(m-1, n))$$

. In the calibration data, a particular code indicates which case applies to this pixel. If it is a good pixel, a signal to the output select logic selects the scaled pixel value. If the pixel has been marked as bad, the spatial average is chosen.

## V. SUPPORT SOFTWARE

Software support for the RDPP will consist of a design entry tool and simulator, a compiler, and run time support tools. The highest priority at this time is a design tool to assist in verifying the RDPP architecture. The RDPP processing elements will have only fixed-point arithmetic capability. In order to simplify application design, a representation system and software tool are being developed. Based on a Sign-Integer-Fraction (SIF) number scheme, the tool will provide the application designer some of the convenience of floating point with the hardware efficiency of fixed point [3].

## VI. CONCLUSION

An Ultra-Low-Power, Radiation-Tolerant (ULP/RT) design library based on the AMI 0.35 $\mu$  CMOS process has been selected to implement a space-qualified RDPP integrated circuit. The basic RDPP architecture had been outlined and mapped to a set of candidate challenge problems. VHDL models are being developed, which will be used for design verification and evaluate design alternatives. While the basic

architecture seems feasible, the details of the processing elements must be defined, along with the interconnect scheme, on-chip memory, and the external interface.

The ULP/RT technology changes the low-power design space. Dynamic power consumption is significantly reduced. Low-power design techniques as applied to standard CMOS, which seek to minimize switching activity, will provide little power reduction with ULP/RT. However, since switching carries a negligible power cost in this technology, highly parallel architectures based on conditional data transfer become more attractive. The RDPP is being developed to exploit the particular properties of the ULP/RT technology.

## REFERENCES

- [1] DeHon, André. "Reconfigurable Architectures for General-Purpose Computing", *Massachusetts Institute of Technology Artificial Intelligence Laboratory Report No. 1586*, sponsored under contract F30602-94-C-0252, October 1996.
- [2] Donohoe, Gregory W., and James Lyke, "Adaptive Computing for Space", *Proc. 42<sup>nd</sup> Midwest Symposium on Circuits and Systems*, Las Cruces, New Mexico, August 8-88, 1999.
- [3] Donohoe, Gregory W., David Buehler, and Stephen Bruder, "A Design Strategy for Fixed Word Length Data Paths, to appear, *Proc. 9<sup>th</sup> NASA Symposium on VLSI Design*, Albuquerque, New Mexico, Nov. 8-9, 2000.
- [4] Gambles, Jody W., and Gary K. Maki, "Rad-tolerant flight VLSI from commercial foundaries", *Proc. 39<sup>th</sup> Midwest Symposium on Circuits and Systems*, pp. 1227-1230, August, 1996.
- [5] Lee, Edward A., and Thomas M. Parks, Dataflow Process Networks, *Proceedings of the IEEE*, May 1995.
- [6] Liu, M. N., and S. Whitaker, "Low Power SEU Immune CMOS Memory Circuits", *IEEE Trans. On Nuclear Science*, vol. 39, pp. 1679-1684, December 1992.
- [7] Villaseñor, J., and W.H. Mangione-Smith, "Configurable Computing", *Scientific American*, June, 1997