RECONFIGURABLE ANTENNA PROCESSING WITH MATRIX DECOMPOSITION USING FPGA BASED APPLICATION SPECIFIC INTEGRATED PROCESSORS

M.P.Fitton, S.Perry and R.Jackson
Altera European Technology Centre,
Holmers Farm Way, HighWycombe, Bucks HP12 4XF, UK
mfitton@altera.com, sperry@altera.com, rjackson@altera.com

ABSTRACT

The flexibility of FPGAs makes them ideal for application within Software Defined Radio (SDR). In this contribution, the implementation of Application Specific Integrated Processors (ASIP) is demonstrated on the FPGA, specifically targeting QR matrix decomposition. The ASIP comprises a number of functional units, controlled by a simple processor and associated program, and dedicated for one or more specific algorithms or operations. SDR Reconfiguration simply requires modification of the program of the ASIP, without full or partial reconfiguration of the device to meet different requirements. QR Decomposition based RLS (QRD-RLS) is suitable for a wide variety of wireless applications, including antenna processing and amplifier digital predistortion. Using the ASIP approach, it is possible to trade off size and performance to reach the optimum architecture for a particular set of requirements.

1. INTRODUCTION

Programmable logic is widely seen as a suitable solution Software Defined Radio and reconfiguration. Here, a novel Application Specific Integrated Processor approach is employed in FPGA, facilitating the processing of computationally intensive tasks without requiring complete or partial reconfiguration of the device to meet different requirements.

In contrast to other work [1], this contribution considers the ASIP technique specifically for matrix manipulation. Although the resulting hardware can be generally applied in adaptive filtering (e.g. Polynomial based Digital predistortion), the application area considered here is primarily antenna processing. QR decomposition is a numerically stable technique that is employed in spatio-temporal or spatial beamforming and Multiple-Input Multiple-Output (MIMO) techniques [2]. In particular, in this contribution we consider the QR decomposition-based Recursive Least Squares (QR-RLS) algorithm [3].

From a practical point-of-view, this approach would allow a system to be readily configured "on-the-fly" for a number of different requirements. Multiple modes of SDR operation and radio standards could be supported in a scalable way. Moreover, within a single mode the system can be intelligently configured to varying wireless channel conditions and external requirements, trading off hardware, processing time (e.g. update rate) and quality (e.g. number of antennas). For example, the same hardware could support spatial beamforming, spatio-temporal beamforming (to combat time dispersion in the wireless channel) and Layered Space Time reception, depending on the prevailing channel conditions, interference, regulatory requirements, etc.

Using the Application Specific Integrated Processors (ASIPs) technique the FPGA is configured to provide a number of functional units, and controlled by a simple processor and associated program. Reconfiguration then merely requires modification of the program of the ASIP and can therefore be performed quickly and simply. This approach facilitates high-level synthesis and a more software-oriented approach than conventional hardware design. Moreover, this technique permits a straightforward migration from programmable logic to ASIC without compromising the reconfiguration.

2. APPLICATION SPECIFIC INTEGRATED PROCESSORS IN FPGA

Conventionally, architectures address different ends of the performance spectrum, from a general purpose processor implementing software algorithms to fixed dedicated custom pipelines. Often, what is required is a balance of the high level performance associated with dedicated hardware, coupled with the flexibility of a processor running standard software.

Figure 1 shows the general form of an FPGA based ASIP. A pipelined program memory and program counter supply the machine with an encoded instruction word. The program memory is typically included within the processor and exploits the dual-port facilities of the memories to allow external sources to load program code.

The encoded instruction word feeds a decode block that decodes the data to provide a set of control signals for the processor. Control signals include: immediate values such

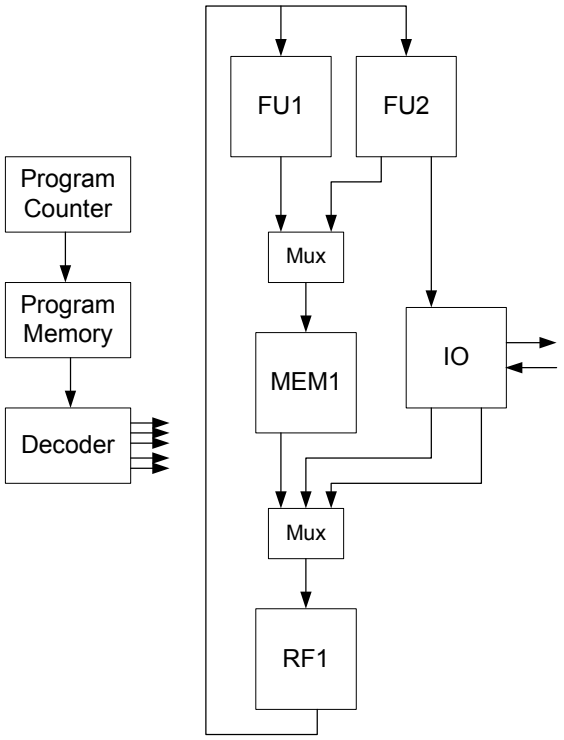


Figure 1 Application Specific Integrated Processor.

as literals, register file read and write addresses; function unit enable and operation select signals; multiplexor operand-select codes.

The processing core includes a set of function units and the multiplexors that route data between them. The function units include memories, registers, basic arithmetic and logic units, and multiply-add blocks. These blocks may exploit specific features of the FPGA device or may rely on standard libraries such as LPM [4]. In addition, custom application specific units may be included.

Function units implementing bus-masters, slaves, general purpose I/O, and streaming point-to-point protocols provide I/O functionality.

3. QR DECOMPOSITION BASED RECURSIVE LEAST SQUARES

3.1 Background

QR-Decomposition is a well-accepted technique employed in matrix calculations. The matrix A is decomposed into Q and R:

$$\mathbf{A=Q.R}$$

Where R is upper triangular and Q is orthogonal, that is:

$$\mathbf{Q^T.Q=1}$$

In this example, Q is formed of a sequence of Givens rotations [3], each designed to annihilate a particular element of the matrix. QR-Decomposition can be used to solve systems of linear equations [6]; i.e. to solve for optimum solution **w**, given output vector **z**, first form $\mathbf{Q}^T.\mathbf{b}$:

$$\mathbf{A.w = z}$$
$$\mathbf{R.w = Q^T.z}$$

Then solve for **x** using backsubstitution, as shown below for *N* coefficients.

$$w_N = \frac{z_N^{'}}{R_{NN}}$$

$$w_i = \frac{1}{R_{ii}}\left(z_i^{'} - \sum_{j=i+1}^{N} R_{ij} w_j \right)$$

$$for\ i = N-1,...,1$$

It is often appropriate to solve a succession of linear systems, each slightly different from the previous one. Calculating the optimum solution afresh for each iteration is prohibitively expensive in terms of complexity, as each calculation is $O(N^3)$. However, it is possible to update the matrix decomposition in $O(N^2)$ operations.

In particular, the Recursive Least Squares form of QR decomposition (QRD-RLS) is used to compute and update the least-squares weight vector of a finite impulse response filter. Standard Recursive Least Squares uses the time-averaged correlation matrix of the data; in comparison QRD-RLS operates directly on the input data matrix. This approach is more computationally complex, but has the advantage that it is more numerically stable than standard RLS [3]. With QRD-RLS, the decomposed matrices that are formed are iteratively updated with a forgetting factor λ, as shown in subsequent details on implementation.

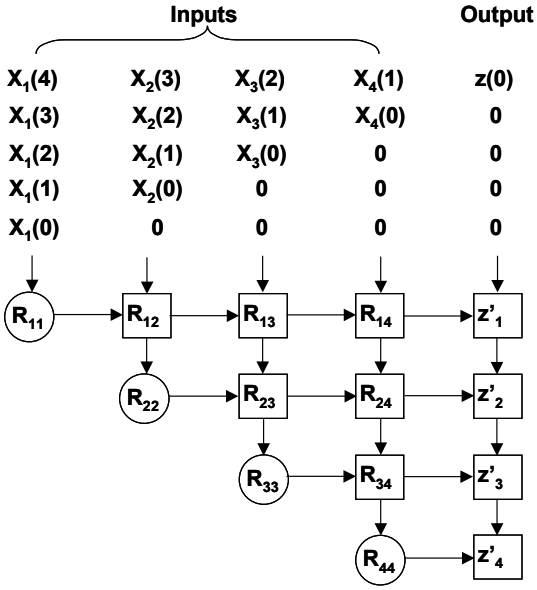


Figure 2 Systolic Array Implementation

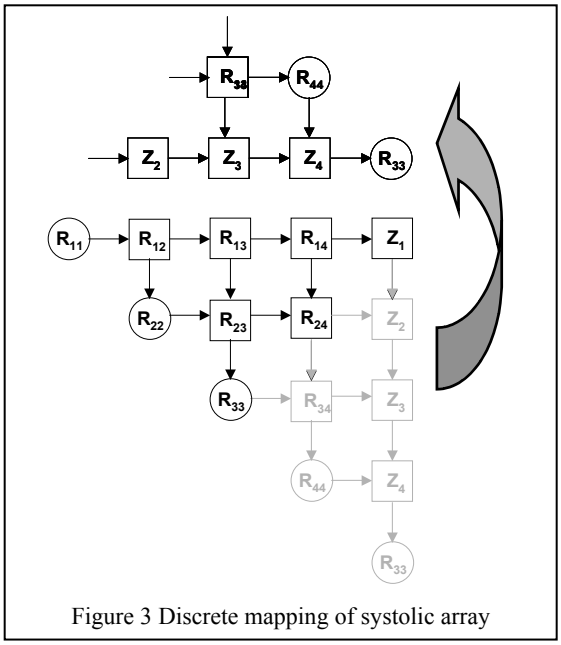
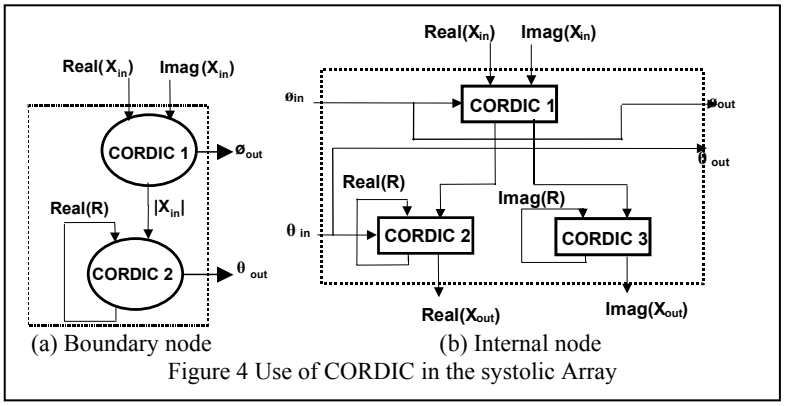


Figure 3 Discrete mapping of systolic array



(a) Boundary node (b) Internal node

Figure 4 Use of CORDIC in the systolic Array

3.2 Application areas

There are a number of areas where it is appropriate to apply QR-decomposition, and particularly QR-decomposition based RLS that provides a method for iteratively updating the optimum solution, based on new inputs.

The technique can be applied in general signal processing problems (i.e. time domain equalization), although complexity may prevent this. However, it may be appropriate to apply the technique in antenna beamforming.

The algorithm can also be exploited in Multi-Input Multiple Output techniques [2], in particular to solve the channel covariance matrix, allowing the parallel data streams to be extracted.

Another example of where this technique can be used is polynomial-based amplifier digital predistortion. Here an adaptive filter is applied on a number of higher-order polynomials of the input data, for example to apply an inverse of the transfer characteristic of a subsequent power amplifier. In this case, QRD-RLS can be used to calculate and iteratively update the optimum filter coefficients that are applied.

4. QRD-RLS ARCHITECTURE IMPLEMENTATION

QR-RLS is suitable for a parallel implementation in the form of a systolic array [3], which is ideal for a hardware solution. However, the resulting architecture can be difficult to reconfigure or scale, and may become too large, especially for a large number of inputs or limited hardware requirement. In this case, the mapping of the systolic array processing cells to available hardware resources necessitates complex control from a conventional RTL perspective. In contrast, using the application specific processor facilitates control and re-use of hardware, permitting a readily scalable and reconfigurable solution. This is complementary to a conventional general-purpose processor approach, as an ASIP solution permits efficient use of the available hardware, targeted for a specific set of requirements [1].

4.1 Systolic Array

An illustration of a systolic array for QRD-RLS is included in figure 2; the example uses four input streams (i.e. N=4) as well as the (desired) output data (**z**). Data flows from top to bottom in the structure, and data is inputted in a time-skewed; the calculations for a particular decomposed matrix (**R**) propagate through the array on a diagonal wavefront.

Two types of systolic node processing elements are employed here: internal cells and boundary cells. *Boundary cells* are used to calculate the Givens rotation that is applied across a particular row in the matrix. As such, the new (complex valued) input is compared to the stored data value (denoted R_{ij}), and a unitary transform is calculated which annihilates the previous value (which is the conceptual output) and calculates the new value of this element. This value corresponds to the magnitude of a vector made up of the input value and the previous value (scaled by the forgetting factor λ).

The unitary transform (Givens rotation), which is calculated in the boundary cell, is outputted and applied to the remainder of the row by *internal cells* (with an index R_{ij}, where $i \leq j$). These processing elements apply the transform to input values, and previous (stored) values, to calculate a new (stored) value, and an output. The transform is also outputted, to be used by the next boundary cell in the row.

4.2 Discrete mapping

In section 4.1, we summarize the operation of the nodes within the systolic array. In the fully parallel configuration, each node would mapped onto discrete hardware, giving a fast but large solution. In reality, there maybe insufficient hardware resources, or a different balance may be required. In general, this can be achieved by mapping multiple nodes

onto a single instantiation of hardware. One method is time-multiplexing nodes onto a number of node processors is using *discrete mapping* [7].

The nodes in the array are re-arranged, as shown in figure 3. This technique maps the same number of nodes onto each diagonal and regularizes the connections between the nodes. One diagonal is then processed in each time-slots, requiring five time slots and three processors (2 internal and 1 boundary) in this example.

There are obviously a wide variety of configurations that can be explored ranging from a fully parallel implementation to a single processor which performs all the required operations for all nodes. It is the strength of the Application Specific Processor approach that these trade-offs can be easily explored.

4.3 The use of CORDIC in QR Decomposition

The operation of the processing nodes has been described, but no specific information on implementation has been supplied thus far. One method to calculate and apply the unitary transformation is to use Coordinate Rotation by Digital Computer (CORDIC). In general, this technique is employed to transform between Cartesian and Polar coordinates (vector translation) or to apply a vector rotation.

This maps well to the requirements of a systolic array performing QRD-RLS. The boundary cells calculate the rotation required to annihilate the lower triangular elements of the decomposed matrix; the internal cells can then apply these rotations.

The operation is somewhat complicated if complex data is used, as shown in figure 4. In the boundary cell, it is firstly necessary to rotate the (complex) input vector so that the quadrature component is zero; another CORDIC block is then used to annihilate the previously stored component (with a phase rotation of θ). Both of these phase rotations are then applied in the internal node. In this case, as the output and stored values are both complex, three CORDIC blocks are required.

4.4 ASIP implementation of QRD Systolic Array

Previous sections describe the operation if a QRD-RLS systolic array calculation; here we consider the peculiarities of an ASIP implementation of the functionality.

An ASIP architecture lends itself particularly well to this algorithm, as it is possible to share resources within a node and between nodes. Furthermore, it is straightforward to support encapsulation and abstraction, easing the design process.

To illustrate these concepts, and to balance the use of Logic Elements and hard multipliers in an Altera Stratix device, a mixed Cartesian-Polar calculation is performed. Here, the unitary transform is calculated in the boundary cells using CORDIC, as described in section 4.3. However, the transform is then applied using complex multipliers, which can be configured in the Stratix device to accept a new complex calculation every cycle.

The scheduling and operations in the systolic array are then controlled by the processor program, which directs input data to the appropriate functional units, initiates processing and deals with the outputs of a particular calculation. In the case of a single processing element performing all calculations for all nodes, the program merely inputs appropriates data to the data path hardware for each calculation. With more hardware, more nodes can be processed in parallel; in this case the ASIP program controls the ordering of the calculations and the routing of input and output data.

From a memory architecture perspective, the architecture encapsulates the memory associated with a particular processing element *within that element*. This reduces the memory bandwidth requirements, as compared to a single large memory for all processing elements. Furthermore, using a discrete mapping regularizes the connection between nodes (and therefore processing elements) such that a maximum of one value is written into each memory, each iterations regardless of the size of the array.

4.5 Back substitution

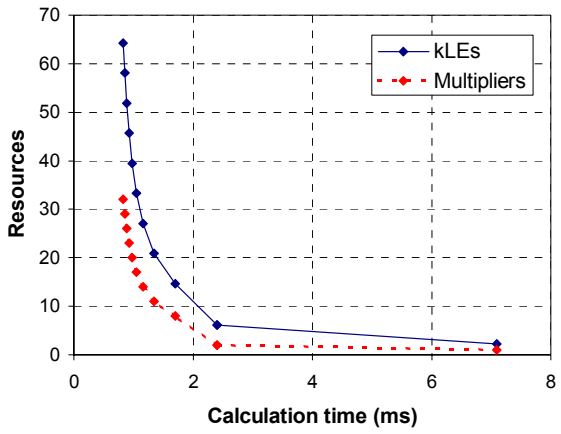
Once the matrix has been decomposed, it is necessary to perform back-substitution to solve for the optimum adaptive filter weights, as described in section 3.1. A custom logic implementation can be used;. the resource utilization required will depend on the number of variables. The advantage of this approach is that it is possible to provide an updated weight vector for each new calculation of the decomposed matrix.

Alternatively, a Nios II soft processor core is applied to perform backsubstitution. The Nios II soft processor family from Altera is a parameterisable set of general-purpose RISC processor cores providing as standard:

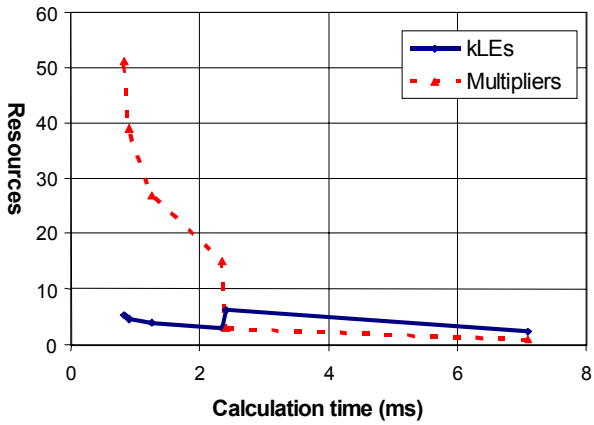
- 32 bit instruction set architecture
- 32 bit data path and address space
- 32 general-purpose registers
- 32 × 32 multiply and divide instructions
- Generic data and instructions caches

As such, it is likely that a Nios II core may be present to perform other control functionality and could be reused to perform backsubstitution.

If we consider an array with 20 coefficients, , the Nios processor takes approximately 5623 clock cycles or 37μs at 150MHz (the clock speed for fast Nios II implementations). Although slower than a pure hardware approach, the presence of the Nios processor lends flexibility to the system. Moreover, the Nios processor can be used to implement other data and control functions on the FPGA. This facilitates a complete System on a Programmable Chip (SOPC) solution without the need for an external processor.



(a) Configuration A (Polar)



(b): Configuration B (Polar/Cartesian)

Figure 5: Size-speed trade-off for Configuration B (Polar/Cartesian)

The final approach considered here is to use an additional ASIP for the backsubstitution calculation. Here it is assumed that this calculation will occur periodically rather than after every update of the decomposed matrix. In this case, an additional 460 cycles are required to perform the back substitution (only 2.3µs at 200MHz).

5. PERFORMANCE AND RESOURCE UTILISATION

Here we consider a number of architectures, and the corresponding resource utilization. In general, we consider targeting Altera Stratix technology. The Stratix device family is based on a 1.5-V, 0.13-µm, all-layer-copper process technology and offers up to 79,040 logic elements (LEs), 7 Mbits of embedded memory, optimized digital signal processing (DSP) blocks, and high-performance I/O capabilities. However, it is also possible to utilize different technology, including Altera Cyclone II and Stratix II families.

Figure 5 shows the resource utilization against calculation time for an arbitrary requirement of 20 coefficients and 2048 updates. It is obviously possible to support different numbers of coefficients and updates; these values are chosen to illustrate performance and the size-performance tradeoff. The diagrams describe resource utilization in terms of thousands of logic elements (kLEs) and number of 18bit multipliers.

Configuration A uses CORDIC blocks to perform a the QR Decomposition using polar calculations. Consequently, a large number of Logic Elements are used in the realization of the CORDIC block. Multipliers are used to remove the scaling which is reduced by CORDIC; alternatively, these can be replaced with shift-and-add circuitry to remove the need for hard multipliers. The alternative architecture uses CORDIC in the boundary cell calculations and applies these with complex multiplications in the internal cells. This provides a different size-performance tradeoff, with more multipliers used (as shown in figure 5b).

The lowest performance corresponds to a single processing element, containing a single, time-shared CORDIC block. This configuration requires 2.3kLEs, one 18-bit multiplier and 7 M4ks (4kbit memories). The time required for 2048 iterations of 20 elements is 7ms. Another point on the size-speed tradeoff, a calculation time of approximately 1.3 ms is possible with a mixed Cartesian-Polar architecture using 3.8kLEs and 27 18bit multipliers. Alternatively, similar performance is exhibited with an all-Polar architecture requiring approximately 21kLEs. Scaling of CORDIC gain is required, using either shift-and-add hardware or hard multiplies (a total of 11).

To calculate 2048 updates of the QRD-RLS calculation, a total of 5.16 million Multiply-and-Accumulates (MMACs) are required. With a calculation time of 1.3ms, this corresponds to 4000MMACs per second. As a reference, the CORDIC-only architecture for this implementation corresponds to 26% of the available Logic Elements in the largest Stratix device (EP1S80), 13% of the available hard multipliers, and less than 0.5% of the embedded memory bits. Consequently, in this example the remaining resources can be utilized to improve the QRD-RLS calculation time, or used for other applications.

6. CONCLUSIONS

In the contribution, we have demonstrated how a QR-Decomposition based RLS technique can be implemented with Application Specific Integrated Processors. The FPGA is configured as a number of custom processors, that can efficiently re-use available resources. This approach facilitates high-level synthesis and a more software-oriented approach than conventional hardware design. Control of the functionality is afforded by a simple program kept in random access memory, allowing significant on without modifying hardware resource. In addition, this approach permits a straightforward migration from programmable logic to structured ASIC **Error! Reference source not found.** without compromising the reconfigurability.

Results indicate that a high performance can be exhibited, in terms of overall calculation time. However it is also possible to trade-off size and performance to reach the optimum implementation for a particular set of requirements.

7. REFERENCES

[1] "Reconfigurable Radio with Application Specific Integrated Processors and Programmable Logic," submitted to SDR Technical Conference 2004

[2] B.Vucetic, J. Yuan, *Space-Time Coding*, Wiley

[3] S.Haykin, Adaptive Filter Theory, 4th edition, Prentice Hall

[4] Electronic Industry Alliance, "EIA/IS-103-A Library of Parameterized Modules" http://www.ediforg/lpmweb, 1999.

[5] Tim Zhong Mingqian, A.S.Madhukumar, and Francois Chin, "QRD-RLS Adaptive Equalizer and its CORDIC-Based Implementation for CDMA Systems," *International Journal on Wireless & Optical Communications,* Volume 1, No.1 (June 2003), pages 25-39.

[6] William H. Press, et al *Numerical Recipes in C*

[7] R.L. Walke, R.W.M. Smith, "Architectures for Adaptive Weight Calculation on ASIC and FPGA," *33rd Asilomar Conference on Signals, Systems and Computer*s, 1999

[8] Altera Nios II Processor, http://www.altera.com/literature/lit-nio2.jsp

[9] Altera HardCopy Devices www.altera.com/products/devices/hardcopy