Dynamic Reconfigurable Distributed Processing Network with Dual Levels of Operand Granularity

Fernando Martinez Vallina, Erdal Oruklu, Jafar Saniie
Electrical and Computer Engineering Department
Illinois Institute of Technology
Chicago, USA

***Abstract*—** This paper introduces a platform capable of handling varying operand sizes across a set of algorithms to be accelerated by a low power hardware core. The key to dealing with different levels of operand granularity is to implement them on the Reconfigurable Distributed Processing Network (RDPN), which can handle dynamic changes to operand size. This approach allows algorithms which may be incompatible for the same hardware platform to be accelerated by a common hardware core. This acceleration core based on the RDPN allows algorithms of different levels in operand granularity to be accelerated by a low power system without the need for extra logic. The RDPN architecture provides a flexible platform, which bridges the gap between software and hardware by extending the applicability of an embedded hardware system. This extended applicability is a result of the native support for word size extension and contraction within the RDPN computational fabric. A case study from the signal processing algorithmic domain will be presented to show how the RDPN fabric dynamically adapts to the operand granularity required by different stages of an algorithm.

I. INTRODUCTION

Questions of operand granularity and hardware resources are key concerns when deciding if an algorithm can be adequately mapped into a hardware platform. In most cases these issues are handled during the initial design and the set of algorithms to be implemented is considered to be fixed from the start. While this is an efficient approach for a limited set of algorithms, it lacks the flexibility needed to incorporate additional algorithms during the life of the system. Algorithms such as those in digital signal processing often share the same set of hardware resource requirements, but may vary in operand granularity depending on the target application. This variation in operand granularity makes it difficult to design a hardware platform that can accommodate a broad range of algorithms without any redundant logic and increased power consumption.

Traditionally, issues concerning different levels of operand granularity have been dealt with by using an FPGA based solution instead of an ASIC. Although the FPGA can accommodate many levels of operand granularity, switching between levels can not be carried out without a reprogramming of the chip. The reprogramming of an FPGA causes all useful computations to be halted and the executing algorithm is delayed until the reprogramming is complete. In contrast, the Reconfigurable Distributed Processing Network (RDPN) [1] allows for the real-time switching of operand granularity through the use of a distributed processing network without halting algorithm execution.

The RDPN architecture is based on a hierarchical distributed processing network architecture, which tackles the problem of reconfigurable computing similar to the GARP [2] and CHIMAERA [3] reconfigurable processors. The main difference between GARP and CHIMAERA is the level of operand granularity. GARP employs coarse grain optimizations [2]. In contrast, the CHIMAERA is targeted at fine grain optimizations [3]. The RDPN architecture provides both fine and coarse grain optimization throughout the entire development cycle with dynamic granularity switching at the hardware level.

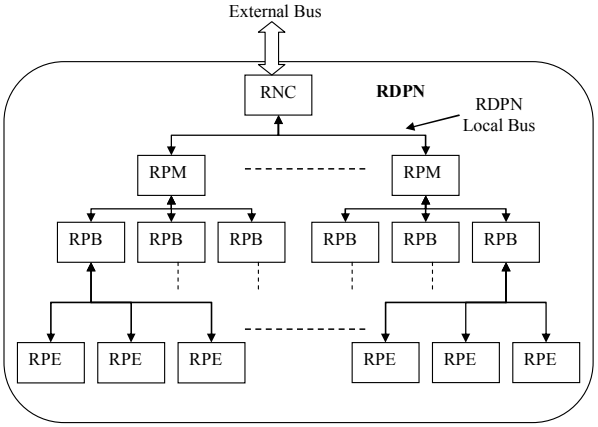

Figure 1. RDPN Hierarchical Overview

As shown in Fig. 1, the RDPN processing fabric is modeled as a four level hierarchical network in which the computational elements are placed at the network edge in the Reconfigurable Processing Element (RPE) [1]. This setup of computational elements allows the RNC to dynamically group RPEs into clusters that physically map and meet the resource and operand size requirements of a task. The four levels of the architecture consist of the Reconfigurable Network Controller (RNC) [1], the Reconfigurable Processing Module (RPM), the Reconfigurable Processing Block (RPB) [1], and the Reconfigurable Processing

0-7803-9197-7/05/$20.00 © 2005 IEEE.

Element (RPE) [1]. Out of these network levels, the RNC is responsible for the dynamic switching of operand size.

The development cycle of the RDPN consists of two stages. The first stage is the hardware synthesis of the RDPN with optimizations targeted at the predominant tasks in the executing task mix. Optimizations at the hardware level are determined by the number and type of processing elements within the RDPN fabric. These optimizations determine the type of algorithms which will have the best performance on a unique implementation of the RDPN. The second stage of the development cycle comprises the software development for the RDPN and provides both high and low level optimizations. High level optimization is done during program compilation in which the RDPN compiler will carry out global optimization on the user code. Low level optimization is done during runtime by the RNC. The low level optimization of the user application during runtime implies that the RNC will keep a tight match between executing tasks and available resources to minimize the idle time of any computational unit. These runtime optimizations also include the dynamic switching of operand granularity across the RDPN processing fabric to match task requirements and maximize resource utilization. As part of these runtime optimizations, the RNC provides translation of RDPN compile code into the RDPN mapping language. This online code translation simplifies the design of the software compiler while at the same time allows the RNC to have a tighter control over a dynamic system. Code translation and task assignment by the RNC to the computational elements is carried out based on current resource availability at any given point in time.

II. RDPN CONTROL AND OPERAND GRANULARITY

Real-time switching of operand granularity within the RDPN computational fabric is a direct result of the hierarchical nature of the RDPN. This hierarchy allows the RNC to control the processing fabric in a way that permits operand word size extension and contraction without halting the system. There are no strict limitations placed on the operands of the executing task mix by the RDPN. The only limitations experienced by tasks executing on the RDPN are during the queuing stage of compile code translation. At this stage a task has to wait for resources to become available before it can be placed in the executing task mix.

There is only one RNC in the RDPN. Besides functioning as a central communication node between the RDPN and the rest of the host system, the RNC provides real-time task mapping and scheduling as well as command and control of the RDPN processing fabric. The functions of the RNC are

- To provide a central controller for the RDPN
- To direct the flow of configuration and instruction packets throughout the processing fabric
- To dynamically alter the level of operand granularity across the processing fabric
- To coordinate the concurrent execution of both related and unrelated tasks
- To carry out dynamic runtime reconfiguration
- To translate RDPN compile code into the RDPN mapping language

The architecture of the RNC is shown in Fig. 2.

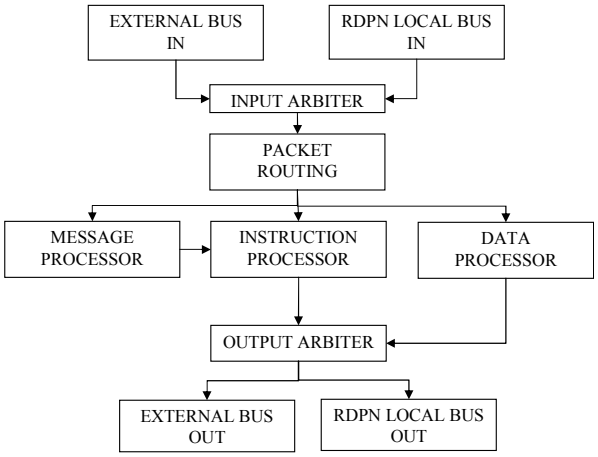

Figure 2. RNC Architecture

The *external bus* is the RNC interface to the components external to the RDPN such as a flash card or some other form of memory. The RDPN *local bus* is the interface between the RNC and the RPMs. The main components of the RNC are the three parallel packet processors, namely *message processor*, *data processor*, and *instruction processor*. The function of the message processor is to initiate and handle all queries and replies concerning resource allocation. The data processor is responsible of routing data streams between the RPMs and the external bus. The instruction processor is responsible for the queuing and translation of RDPN compile code into the RDPN mapping language. The use of on the fly code translation and mapping based on available resources allows the RNC to dynamically change the level of operand granularity across different sectors of the RDPN processing network.

The finest level of granularity available to the RNC is defined as the operand size of a single Reconfigurable Processing Element (RPE). The RPE is the lowest level element in the RDPN network hierarchy, and it is where all the computational work takes place. The operand size of these functional units is set during the realization of the RDPN on either an FPGA or an ASIC. Coarser levels of operand granularity are achieved in real-time based on instructions received from the Reconfigurable Network Controller (RNC). These instructions specify the required operand size and group RPEs into virtual clusters to achieve it. The processing fabric of the RDPN has support for word size extension in the form of embedded glue logic.

The glue logic is needed to extend the word size of the RPE from its base size set at the time of implementation to

the required size during algorithm execution. Fig. 3 shows how this logic is embedded into each RPB block and allows the block to generate results of the correct operand size. During algorithm execution the RNC will command the processing fabric to carry out word size extension if the required operand size exceeds the size of a single RPE. For example if the algorithm in execution uses 16-bit operands and the RPE is implemented as an 8-bit unit, word size extension will occur. During this process the initial 16-bit operands are split into 2 8-bit blocks. These resized operands are then sent to the RPEs for parallel computation of all portions of the final result. The intermediate results generated by the RPEs are passed on to the glue logic. This logic block is responsible for recombining the intermediate results into a final result based on an instruction from the RNC. In cases where a larger operand size is required such as 64-bits, two levels of the glue logic are used to generate the final result. Glue logic with RPB will recombine the first level 8 bit results from the RPEs to generate a valid 32 bit intermediate result. Once the intermediate results are available, glue logic at the RPM level will combine the 32 bit intermediate results to generate the final 64 bit result.

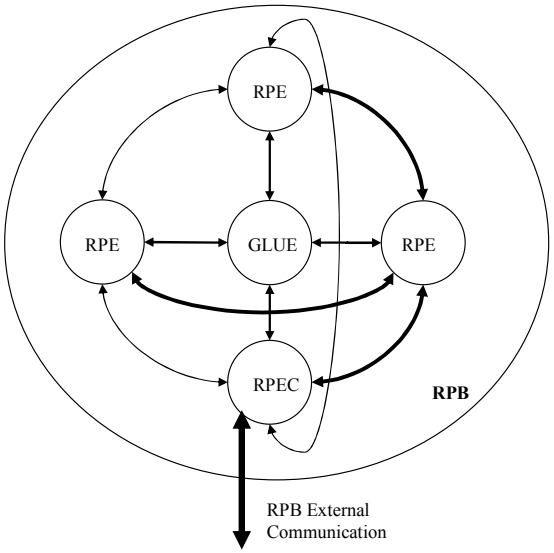

Figure 3. RPB with Glue Logic

By distributing the glue logic in the RDPN fabric at both the RPB and RPM levels, RNC is capable of clustering the RPEs into a virtual Macro-RPE. As shown in the previous example, the Macro-RPE behaves as a single RPE and has a large dynamic range of possible operand sizes depending on the instructions received from the RNC. The only limitation on operand size is given by the maximum size of the RDPN processing fabric. This size depends completely on the realization and it does not have an upper bound determined by the RDPN architecture. The use of glue logic in combination with control from the RNC allows for dynamic clustering and declustering of RPE elements. This makes the RPDN an optimal platform for multiple algorithm acceleration with tight coupling between available hardware and application requirements. The only hardware overhead present in this computational fabric is a result of having the RNC and a few distributed control elements such as the glue logic.

The embedded support for word size extension and contraction gives the RDPN architecture the necessary flexibility to tackle both the problems of operand granularity and power consumption. These characteristics extend the applicability of solutions built upon the RDPN and make it suitable for mobile environments where both power and platform versatility are major design concerns.

III. JPEG2000 CASE STUDY

As a case study, we chose the implementation of the JPEG2000 image compression standard in order to demonstrate the multitasking functionality of the RDPN. JPEG2000 is a popular compression standard for still images [4]. It offers error resilience, good low bit rate performance, scalability, and lossless or lossy encoding. However, JPEG2000 hardware realization presents a formidable challenge for DSP chips or μPs due to many control intensive operations and data interdependencies in the algorithm core blocks [5]. Fig. 3 shows the core block diagram for the JPEG2000 standard.

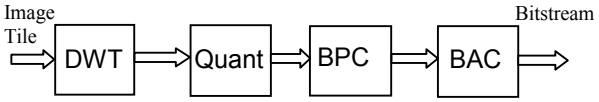

Figure 3. JPEG2000 Core Block Diagram

These core blocks include discrete wavelet transform (DWT), quantization, bit-plane coder (BPC) and binary arithmetic coder (BAC). BAC block compresses the incoming symbols from BPC block and through packetization generates JPEG code stream.

The parallel processing fabric in the RDPN architecture provides a platform on which an efficient implementation of JPEG2000 can be built. The inherent parallelism and the necessity for multitasking in JPEG2000 can be met by employing sufficient RPM units in RDPN for each individual core operation such as DWT.

In the case of DWT core, the 2D wavelet decomposition for images can be realized concurrently using dedicated wavelet processors for each tile. A tile is a non-overlapping rectangular block inside the image and during the pre-processing stage, an *NxN* image is decomposed into several tiles. 2D wavelet decomposition is a separable transform. This property allows for the 2D-DWT to be applied to each tile using two successive 1D wavelet transforms along the rows and columns of the specific tile. The 1D-DWT implementation in JPEG2000 uses lifting scheme for 9/7 or 5/3 wavelet filters [4]. These filters can be implemented using 4 and 2 RPE blocks for 9/7 and 5/3 filters respectively [6]. Each lifting step can be implemented by a single RPE

element [7], assuming 8-bit operands. For larger data sizes, a Macro-RPE will be used.

In this retrospective, the hierarchical structure of RDPN enables the following task allocation:

- An RPM unit corresponds to a core block in JPEG2000 algorithm such as DWT.
- An RPB corresponds to a 1D DWT processor for a specific tile
- An RPE corresponds to a single lifting step.

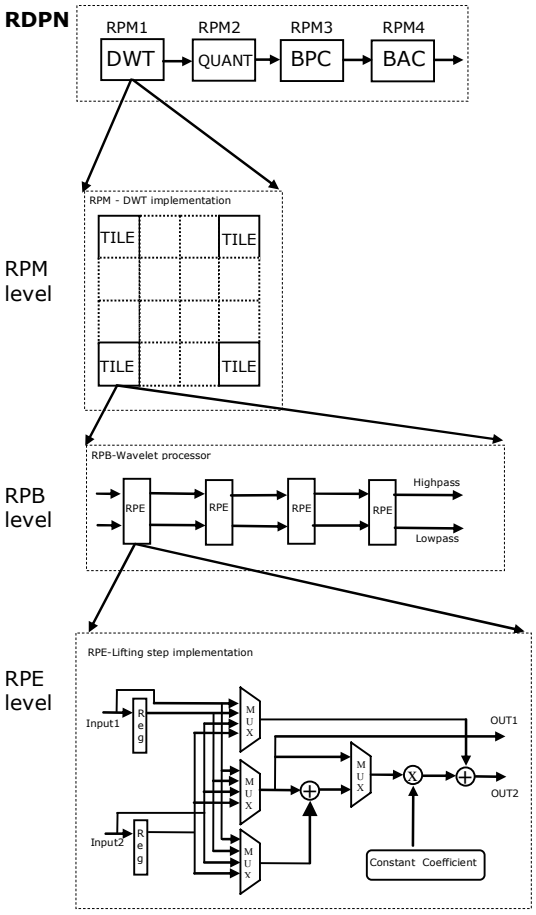

Figure 4. RDPN Implementation of JPEG2000

Figure 4 shows the mapping of the JPEG2000 algorithm as a set of concurrent tasks on the RDPN. The algorithm mapping across all architectural levels of the RDPN is shown by expanding the RPM executing the DWT core block. The breakdown of the DWT across the different architectural levels simplifies the process of spatially mapping the algorithm and increases the level of parallelism in the task. As a result, the algorithm is tightly matched to the available resources and system idle time is minimized.

The architecture of the RDPN makes it possible for the reconfigurable processor approach to overlap execution and reconfiguration times. This leads to an increase in algorithm performance and an increase in system execution time versus idle time ratio.

IV. CONCLUSION

In this paper, we introduce the use of the RDPN as a suitable platform for the implementation of algorithms with varying operand sizes. The RDPN provides designers with the ability to alter operand size in any portion of the RDPN processing fabric in real-time without having to halt the system. This allows for algorithms with different levels of operand granularity to share the same hardware platform and be implemented on a low power and high performance processing core.

REFERENCES

[1] F. Martinez Vallina, E. Oruklu, and J. Saniie, "A Distributed Processing Network Architecture for Reconfigurable Computing," Proceedings of IEEE EIT 2005 Conference, May 2005.

[2] J. Hauser and J. Wawryznek, "GARP: A MIPS processor with a reconfigurable processor," Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, April 1997.

[3] Z.A. Ye, A. Moshovos, S. Hauck, and P. Banerjee,"CHIMAERA: A High-Performance Architecture with a Tightly-Coupled Reconfigurable Functional Unit," Proceedings of the 27th International Symposium on Computer Architecture, June 2000.

[4] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview", IEEE Transactions on Consumer Electronics, vol. 46, no. 4, 2000.

[5] K. Andra, C. Chakrabarti, and T. Acharaya, "A High-Performance JPEG2000 Architecture," Circuits and Systems for Video Technology, IEEE Transactions on Volume 13, Issue 3, Page(s):209 – 218, March 2003.

[6] H. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient architectures for 1-D and 2-D lifting based wavelet transforms", IEEE Trans. on Signal Processing, vol.52, no.5, pp. 1315-1326, May 2004.

[7] E. Oruklu, F. Martinez Vallina, and J. Saniie, "A Reconfigurable Architecture for Target Detection in High Density Clutter Environments using Subband Decoding Algorithms," GSPx 04 Technical Conference, Sept. 2004.