U-Tee Cheah

The MorphoSys Project: Dynamically Reconfigurable Computer Architecture

Key Terms:

- Frame Buffer (FB)
- Tiny Reduced Instruction Set Computer Processor
- Reconfigurable Cells (RCs)
- Direct Memory Access (DMA) Controller
- Reconfigurable Processors: Static and Dynamic
- RISC

Author

U-Tee "Benjamin" Cheah began his research experience by casually approaching his faculty mentor. His project focused on enhancing graphics performance via a novel processor. Ben's most memorable experience occurred when, after much labor, his portion of the project worked. In the future, Ben hopes to attend business or engineering graduate school. He advises students to establish a productive dialogue with their professors in order to facilitate their research.

NEXT

U-Tee Cheah

Abstract

Dynamically reconfigurable microprocessor design is an intriguing prospect for the microprocessors of the future. Dynamic reconfiguration means that the microprocessor can reconfigure itself (the datapath) during execution of a program to optimize its performance, rather than having a pre-determined datapath. The MorphoSys project involves designing a dynamically reconfigurable microprocessor geared specifically toward image processing applications and determining if the trade-off in time, cost, and silicon area result in an overall better microprocessor.

The microprocessor will include a Tiny Reduced Instruction Set Computer (RISC) processor to implement normal processor functions, and an array of Reconfigurable Cells (RCs), which will implement most image processing applications more efficiently than the general purpose processor. The design also includes a Frame Buffer (FB) to store whole frames of images for use by the RCs, and a Direct Memory Access (DMA) controller to provide the FB faster access to memory.

BACK NEXT

Faculty Mentor

The MorphoSys project is targeted for the design of a new generation of microprocessors that will enable the emerging technologies such as: smart missiles, wireless audio/video communicators, autonomous vehicles, and many other technologies planned for the beginning of the next millennium to come to fruition. The Reconfigurable Computing Laboratory of UCI is one of the selected groups in the nation to work on a new paradigm for developing “soft” hardware computer technologies. The notion of “soft” hardware refers to the reconfigurability of the hardware, which is different from current microprocessors. Reconfigurable hardware

Nader Bagherzadeh
School of Engineering

systems are more flexible allowing the system to adjust its hardware resources to accommodate the needs of the user. The UCI group is working on a new microchip called M1 that implements the ideas developed for the MorphoSys project.

BACK NEXT

U-Tee Cheah - The MorphoSys Project: Dynamically Configurable... [1] [2] [3] [4] [5] [6] [7]

Back to Journal 1998 Index

Introduction

With the advent of the Multimedia Age, the system requirements of programs have increased dramatically. The complexities of these applications require that processors be powerful, yet flexible enough to accommodate various classes of applications efficiently. Recently, various reconfigurable processors have been considered for satisfying these requirements. Reconfigurable computing represents a median between general purpose processors, which are capable of running different applications at the cost of increased execution time, as well as for Application-Specific Integrated Circuits (ASICs), which are processors only suitable for running certain types of applications but with much faster implementations. The software programmer or the compiler of a reconfigurable processor should be capable of recognizing the configuration that is best suited for the program (or portion of) in question and ensuring that the processor is set to that configuration.

Traditional reconfigurable processors are statically reconfigurable, which means that the processor is configured at the start of program execution and remains unchanged for the duration of the program. In order to reconfigure a statically reconfigurable processor, program execution would have to be halted while the reconfiguration is in progress, slowing down the execution of the program. Dynamically reconfigurable processors, on the other hand, allow reconfiguration and execution to proceed at the same time.

The MorphoSys project proposes to design a unique dynamically reconfigurable computer architecture geared toward increasing performance of image processing applications. The main components of the architecture (Figures 1 and 2) are the Tiny Reduced Instruction Set Computer (RISC) processor, Reconfigurable Cell (RC) Array, Context Memory, Frame Buffer (FB), Direct Memory Access (DMA) Controller, and main memory (i.e. DRAM, SRAM or RAMBUS).

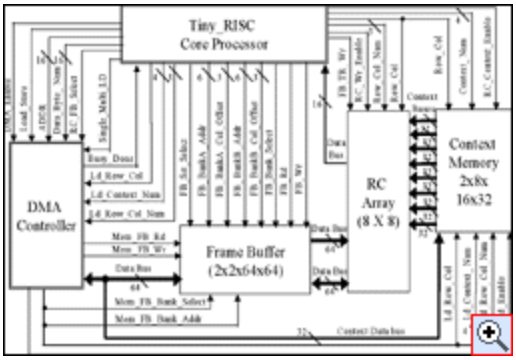

Figure 2
MorphoSys system interface

Several other components (not shown) complete the architecture: secondary storage (i.e. hard drive) and the external interface of the architecture (i.e. to the input and output ports). Tiny RISC is a general purpose processor that handles all instructions not related specifically to image processing. The RC Array is the dynamically reconfigurable co-processor that handles most of the image processing for MorphoSys. The Context Memory stores configurations (called contexts) for the RC Array; the FB and DMA Controller act as the interface between the RC Array and the other components of MorphoSys.

At the time of this writing, the Tiny RISC processor, DMA Controller, and FB have been modeled in behavioral VHDL code; the VLSI design stage of these components has begun. The VLSI design of the RC Array is complete.

Instruction set of the MorphoSys Architecture

The current MorphoSys instruction set consists of 44 Tiny RISC instructions (including newly-added Branch If Greater Than, Branch If Less Than, and Branch If Equal To) and 12 instructions for the operation of the RC Array, FB and DMA Controller.

Components of the MorphoSys Architecture

Frame Buffer:
The FB is a fast memory buffer used to store portions of frames of data (in image processing applications, each image is called a frame). It is the buffer between the DMA Controller and RC Array. The DMA Controller has a 64-bit data bus connection to the FB. The same bus is used for

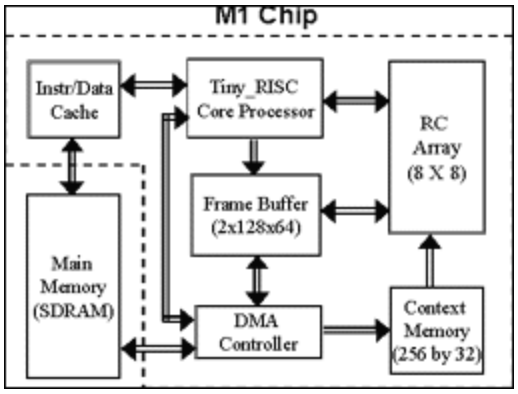

Figure 1
The MorphoSys top-level block diagram

reading and writing data to and from the FB, and therefore the DMA Controller may not read and write at same time. The RC Array has two 64-bit

BACK NEXT

---

buses. One is a read-only bus; the other is a read-write bus. The FB has to send enough data per cycle to a whole row or column of the RC Array. Since there are eight RCs, each needing two 8-bit operands, a total of 128 bits (8 RCs * 2 operands/RC * 8 bits/operand = 128 bits) is necessary, hence the two 64-bit read buses. One 64-bit bus is needed to write data back to the FB from the RC Array because each RC produces an 8-bit output (8 RCs * 1 output/RC * 8 bits/output = 64 bits). Since one of the data buses between the RC Array and the FB is used for both reading and writing, a read and write between these two modules may not occur at the same time. However, the DMA Controller and RC Array buses are independent of each other and may each read or write at will, with the constraint presented below.

The FB is divided into two separate sets (memory buffers). This configuration allows the DMA Controller to access one set while the RC Array is accessing the other. Each set may be accessed by either the DMA Controller or by the RC Array, but not by both simultaneously. Each set is further divided into two banks (Figure 3), each 64 bits wide. The DMA Controller accesses one bank at a time, while the RC Array accesses both banks within the same set at the same time. Thus, the DMA Controller must deliver data to the FB at a rate at least twice as fast as the rate the RC Array reads it. Fortunately, this stipulation does not degrade the performance of the RC Array because many typical applications require the RC Array to perform several operations on the same set of data before the desired result is obtained (the DMA can fill a set of the FB with data before the RC Array needs another set of data).

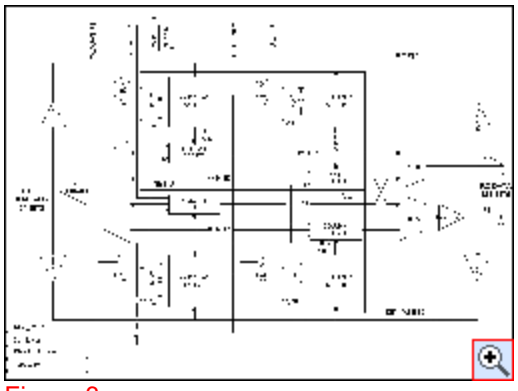

Figure 3
FB Internal Block Diagram

Direct Memory Access Controller:
The DMA Controller acts as the interface between the main memory of the processor and the FB and RC Array modules. The data bus between the DMA Controller and the FB is a 64-bit read-write bus, while the data bus between the DMA Controller and the RC Array is 32 bits wide (Figure 2). Since the data bus to and from memory is 32 bits wide, the DMA Controller needs two cycles to assemble 64 bits of data from the memory for the FB, and one cycle to assemble the 32-bit data for the RC Array.

The DMA Controller has three main components: the Data Packing Register, the Address Generator Unit, and the State Controller. The Data Packing Register assembles the 64 bits of data for the FB. The Address Generator Unit generates and tracks addresses for the memory, FB, and RC Array. The State Controller receives information from the Tiny RISC processor and determines the following sequence of data transfers to and from the FB and RC Array. The amount of data transferred is specified by the information from Tiny RISC.

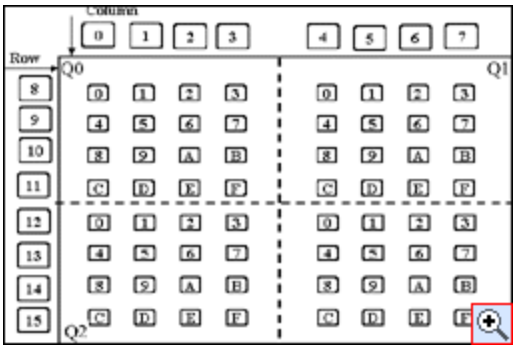

Figure 4
RC Array - 8x8 array of RCs

Reconfigurable Cell Array:
The RC Array is a Single-Instruction Multiple-Data (SIMD) multiprocessor. It consists of an 8x8 array (Figure 4) of processing units (called RCs). The array is row or column reconfigurable, meaning that a whole row or column can be reconfigured at the same time, with the same context across all eight cells. With the same context, each row or column executes the same instruction on different data, hence making each row or column a SIMD multiprocessor. Each RC stores a copy of its current context in its Context Register, which is internal to each RC and separate from the Context Memory. Capability to reconfigure single RCs is present. The power of the RC Array lies in the fact that, depending on a specific application's needs,

it can be configured to be a row or a column of eight multi

BACK NEXT

Page 3

U-Tee Cheah - The MorphoSys Project: Dynamically Configurable... [1] [2] [3] [4] [5] [6] [7]

Back to Journal 1998 Index

processors, with each multiprocessor consisting of eight RCs (it could be a 64 processor); the connections between processors have been designed to allow fast, efficient transmission of data.

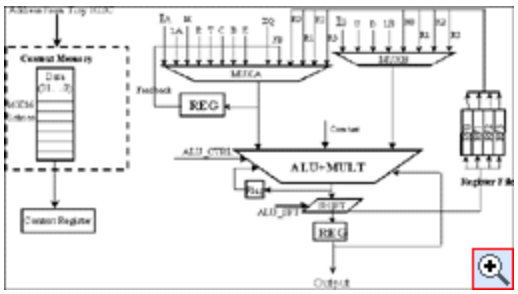

Figure 5
RC internal structure

The internal structure of an RC is shown in Figure 5. Each RC is capable of normal arithmetic (i.e. ADD, SUBTRACT), logic (i.e. AND, XOR, SHIFT, SET_ IF_EQUAL, etc.), multiply, and multiply-accumulate operations. The input and output data bus is 8 bits wide, but internally, the bus is widened to 16 bits to obtain higher precision in the final result (the RC Array takes several instructions to produce the final result, and the internal datapaths are 16-bit to save intermediate results with 16-bit precision; the output register is 32 bits to save the output of multiply and multiply-accumulate operations—two 16-bit operands produce a maximum of 32 bits from a multiply). Only the lower 8 bits of the output register are taken to be the final results which are written out on the data bus (the upper 24 bits are discarded because only the lower 8 bits provide the precision necessary for the applications studied—the internal datapaths and the output register are larger to maintain the precision of the intermediate results, as previously mentioned)

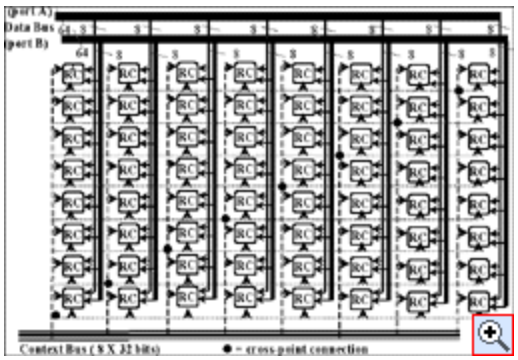

Figure 6
RC Array data and context buses

Figure 6 shows the data and context buses. Data is broadcast to either an entire row or an entire column, and supply both the IA and IB inputs for each RC. Contexts are saved in the Context Memory, which is a 16x16 array of 32-bit registers. With that size, the Context Memory can store up to 16 different configurations for each of the eight rows and each of the eight columns. The context formats are shown in Figure 7.

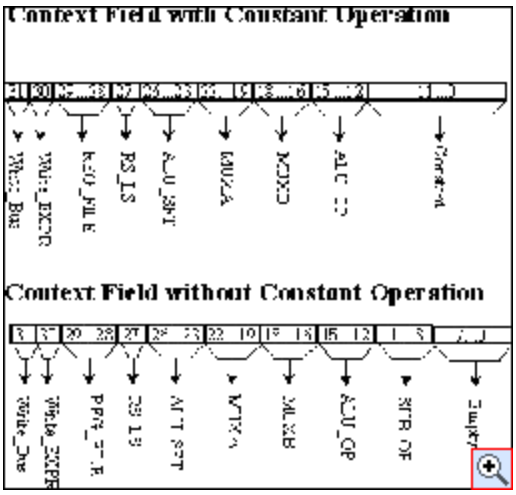

Figure 7
RC Array Context Formats

For purposes of visualizing the connectivity among RCs, the RC Array has been divided into four quadrants (each quadrant is a 4x4 array of RCs). The various internal connections between the RCs are shown in Figures 8 through 13. The first type of connectivity is North-South-East- West, where each RC is connected to its nearest four neighbors (Figure 8). The second type is connectivity within a quadrant. Each RC is fully connected to every other RC in the same quadrant row

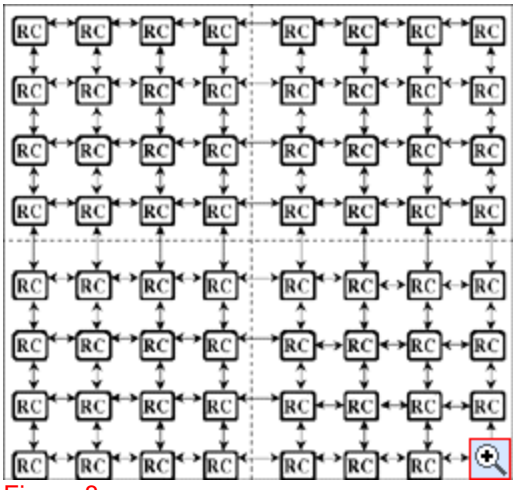

Figure 8
RC Array North-South-East-West connectivity

BACK NEXT

Page 4

U-Tee Cheah - The MorphoSys Project: Dynamically Configurable... [1] [2] [3] [4] [5] [6] [7]

Back to Journal 1998 Index

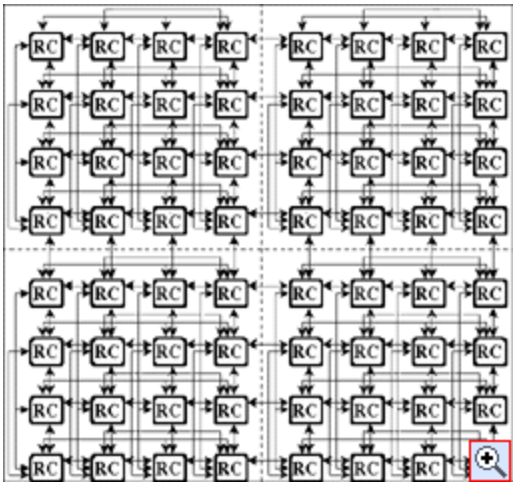

Figure 9
RC Array NSEW and full quadrant row/column connectivity

or quadrant column (Figures 9 and 10). The data from the leftmost RC in the quadrant row is called Left A (LA), the data from the rightmost RC in the quadrant row is called Right (R), and the data from the remaining-

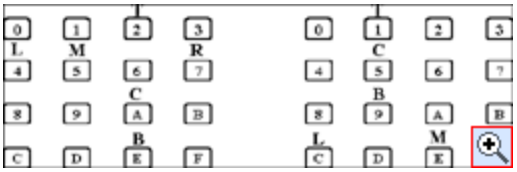

Figure 10
RC Array Port A connectivity

RC in the quadrant row is called Middle (M). The data from the topmost RC in the quadrant column is called Top (T), the data from the bottommost RC in the

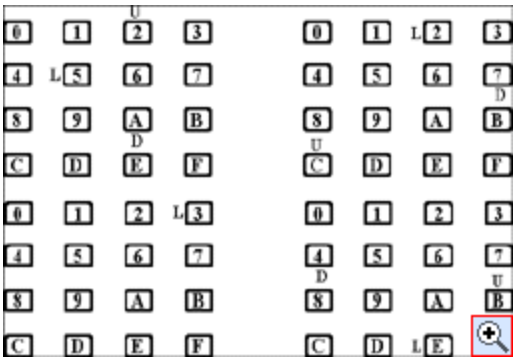

Figure 11
RC Array Port B connectivity

quadrant column is called Bottom (B), and the data from the remaining RC in the quadrant column is called Center (C). Note that the names do not reflect the direction from which the data came (i.e. Left could come from the RC to the right of the RC in question if the RC in question is the leftmost RC in its quadrant row; similarly, Top could come from an RC below). These data are only available for Port A (operand A). Port B (operand B) utilizes only the North, South, and West connections, which in Figure 10 are called Up (U), Down (D), and Left B (LB). This was determined after studying the applications of interest and discovering that the North and South inputs are necessary; the East and West are not. Due to the lack of bits in the context, the number of inputs to the Port B input MUX is limited by necessity—only East or West could be included, and West was chosen arbitrarily.

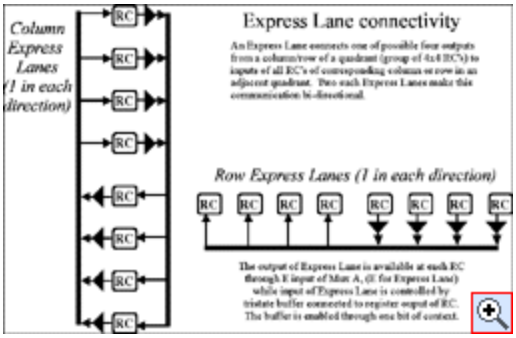

Figure 12
RC Array Express Lane

The third type of connectivity are the "Express Lanes," where every RC is a row or column and is connected to every RC in the same row or column in an adjacent quadrant (Figures 12 and 13). Using the Express

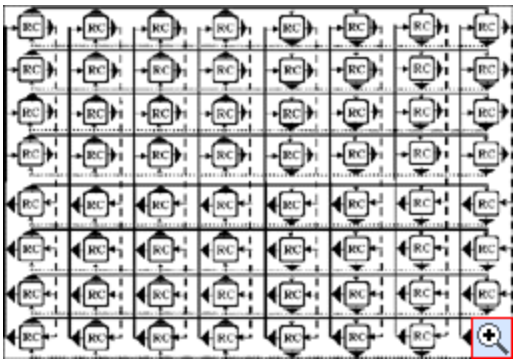

Figure 13
RC Array inter-quadrant Express Lanes connectivity

BACK NEXT

U-Tee Cheah - The MorphoSys Project: Dynamically Configurable... [1] [2] [3] [4] [5] [6] [7]

Back to Journal 1998 Index

Lanes, RCs in adjacent quadrants can send data (denoted by E for Express) to each other without having to go through other RCs, thus saving time. While quadrants which are diagonally across from each other need to use an intermediate quadrant to transfer data between them, it has been determined that this does not occur often in the applications of interest. Hence the resulting performance degradation should not be significant. Figure 5 shows the two ports of an RC: Port A has the inputs IA (Immediate value for Port A, from Frame Buffer), LA (Left, Port A), M (Middle), R (Right), T (Top), C (Center), B (Bottom), XQ (Cross-quadrant), FB (Feedback), and R0 to R3 (data from the 4-deep Register File of the RC); Port B has the inputs IB (Immediate value for Port B, from FB), U (Up), D (Down), LB (Left, Port B), and R0 to R3 (data from the 4-deep Register File of the RC). XQ is data that comes from the cell in the next quadrant directly across from the RC in question. The Feedback input to Port A is the input selected for Port A in the previous cycle. The Register File holds up to four previous outputs of the RC (R0 to R3) and can supply them as inputs to the RC. Note that IA and IB are different data (coming on two separate buses from the FB).

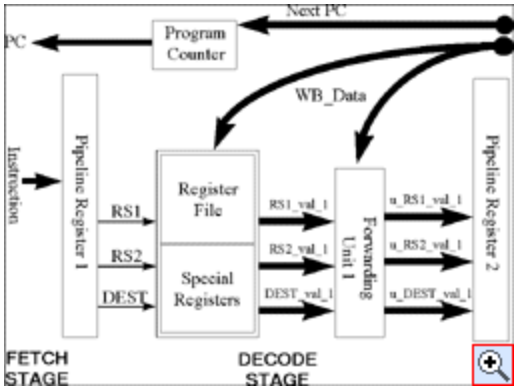

Figure 14
Tiny RISC pipeline stages 1 and 2

Tiny RISC processor:
Tiny RISC is a 4-stage pipelined processor (Figures 14 and 15), designed at the University of California, Irvine by Christopher Christensen as a powerful 16-bit RISC microprocessor. For the MorphoSys project, the design was modeled in behavioral VHDL and the data and address bus sizes were increased to 32 bits. Tiny RISC has four important registers in addition to the Register File and the Special Register File. The first is the Program Counter register, which contains the address of the program execution point. The other three are pipeline registers, which provide the latched interface between each pipeline stage. All processor rising edge of the global clock. The first pipeline register is latched when the instruction acknowledge signal is lowered by the instruction cache controller (which indicates that the instruction is ready).

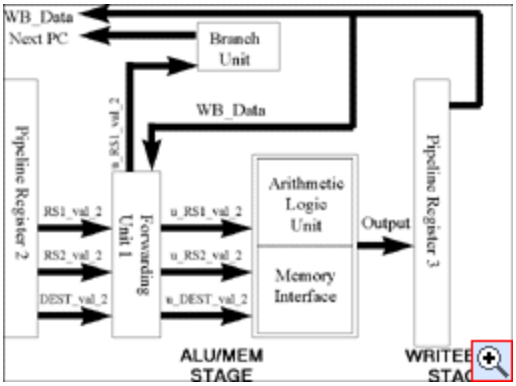

Figure 15
Tiny RISC pipeline stages 3 and 4

The first stage is the Instruction-Fetch stage, where the 32-bit instruction is fetched from the instruction cache. The next stage, the Instruction-Decode stage, includes the Register File and the Special Register File. The Register File consists of sixteen 32-bit registers designed to hold regular data, while the Special Register File has five 32-bit registers designed to hold certain important information relating to interrupts and interrupt returns. Table 1 describes the purpose of each Special Register. The Execution stage is the third pipeline stage, and consists of the Arithmetic Logic Unit (ALU) and the interface to and from the data memory. The Execution stage also includes the Branch Unit, which contains all the logic necessary for determining the next PC value (which is normally incremented, unless the instruction requires a Jump). The final stage is the Writeback stage, where the output from the ALU or the data requested from memory is written back to the Register File. If the instruction is MTS (Move To Special Register), the data stored in the Register File location specified is written back to the Special Register specified. If the instruction is MFS (Move From Special Register), the opposite is done.

Due to its pipelined architecture, Tiny RISC requires two forwarding units to avoid costly delays due to Read After Write (RAW) dependencies. This occurs because it takes the processor two cycles after an instruction has been decoded to write data back to the Decode stage. The two instructions immediately following the instruction in question will need to obtain the updated data (if they are reading from

registers, except the first pipeline register, are synchronous to the

the register that is being written back to). One forwarding unit appears in the

BACK NEXT

Table 1
Tiny RISC Special Register File

Special Register (SREG) #	Description
0	Contains information regarding which interrupts can be serviced (in its IMASK field) and which interrupt is currently requesting service (in its INUM field) – see Figure 16
1	Stores the value of previous SREG(0) when an interrupt occurs, so that it can be restored when the interrupt is returned (the servicing of the interrupt is done)
2	Stores the value of the current PC when an interrupt occurs, so that control can be returned to the correct point after the interrupt has been serviced
3	Provides the address to which the PC jumps when an interrupt occurs
4	Stores the value of the next PC when an interrupt occurs, so that control can be returned to the correct point after the interrupt has been serviced [i.e. if there was no Branch/Jump in the pipeline immediately prior to the interrupt request, the next PC would be the current PC + 1; if there was a Branch/Jump, the next PC would be the address specified by the Branch/Jump (if it is supposed to be taken)]

Decode stage, where it forwards the updated data to the second instruction after the instruction in question. The other forwarding unit is in the Execution stage, where it forwards the updated data to the instruction immediately following the instruction in question.

Application Programs

The purpose of the MorphoSys project is to increase the performance of image processing applications with its unique RC Array-Tiny RISC design. The RC Array dramatically speeds up compute-intensive array-type operations, while the Tiny RISC handles routine operations efficiently. Among the applications studied are Auto-matic Target Recognition (ATR), Motion Estimation (ME), and Discrete Cosine Transform (DCT).

It was found that while MorphoSys is not as efficient as ASICs (i.e. for ME, it takes almost twice as many cycles as an ASIC designed especially for ME), MorphoSys performs much better than a non-reconfigurable processor, like the Intel MMX, which was designed to speed up multimedia (image, audio, etc.) processing applications. The Intel MMX processor uses approximately 28-times the number of cycles needed by MorphoSys to perform full search block matching (for ME), and 10-times the number of cycles to do DCT.

Conclusion

The MorphoSys project has proven that a dynamically reconfigurable processor is capable of vast improvements over a non-reconfigurable processor in terms of speed. The future goals of the project include integrating a reconfigurable memory array into the processor, which will allow better utilization of memory resources and greater performance gains, and producing the complete MorphoSys prototype processor within two years. The first prototype, without the reconfigurable memory array, is due at the end of 1998.

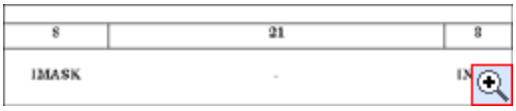

Figure 16
Format of SREG(0)

Acknowledgements

My undergraduate research is supported by the UC Irvine Presidential Undergraduate Fellowship (PUF) and the Undergraduate Research Opportunities Program. Thanks to Professor Nader Bagherzadeh for taking me on as an undergraduate researcher, to Professor Eliseu Filho and Guangming Lu for their invaluable help in my portion of the project, and to Hartej Singh for his willingness in explaining portions of the project in which I am not involved. The following other people have contributed to the MorphoSys project and directly or indirectly aided in the composition of this paper: Professor Fadi Kurdahi, Professor Tomas Lang, Robert Heaton, Ming-Hau Lee, Maneesha Bhate, Matt Campbell, Alexander Gascoigne, Nambao Van Le, Robert Powell, Rei Shu, Lingling Sun, Cesar Talledo, Eric Tan, Tom Truong, and Tim Truong. My sincerest thanks to all of you. Most importantly, I want to thank my parents for supporting me through college, and my grandmother for taking care of me.

Works Cited

Singh, Hartej, Ming-Hau Lee, Guangming Lu, Fadi J. Kurdahi, Nader Bagherzadeh, Tomas Lang, Robert Heaton, and Eliseu M. C. Filho. "MorphoSys: An Integrated Re-configurable Architecture." *NATO Symposium on System Concepts and Integration.* Monterey, CA. April 1998.

BACK

U-Tee Cheah - The MorphoSys Project: Dynamically Configurable... [1] [2] [3] [4] [5] [6] [7]

Back to Journal 1998 Index