

# MorphoSys: A Reconfigurable Architecture for Multimedia Applications

Hartej Singh, Ming-Hau Lee, Guangming Lu, Fadi J. Kurdahi, Nader Bagherzadeh  
*University of California, Irvine, CA (United States),*  
and Eliseu M.C. Filho, *Federal University of Rio de Janeiro (Brazil)*

## Abstract

We describe the MorphoSys reconfigurable system, which combines a reconfigurable array of processor cells with a RISC processor core and a high bandwidth memory interface unit. We introduce the array architecture, its configuration memory, inter-connection network, role of the control processor and related components. Architecture implementation is described in brief and the efficacy of MorphoSys is demonstrated through simulation of video compression (MPEG-2) and target-recognition applications. Comparison with other implementations illustrates that MorphoSys achieves higher performance by up to 10X.

## 1. Introduction

Reconfigurable computing systems are systems that combine reconfigurable hardware with software programmable processors. These systems allow the user to configure or customize some part of the hardware for different applications. Reconfigurable computing adheres to a hybrid approach between the extremes of ASICs (Application-specific ICs) and general-purpose processors. A reconfigurable system typically has wider applicability than an ASIC and better performance than a general-purpose processor for target applications.

Conventionally, field-programmable gate arrays (FPGAs) have been used for reconfigurable computing. These allow designers to manipulate gate-level devices. However, FPGAs are slower than ASICs, have lower logic density and are inefficient for word operations. Several researchers have proposed other reconfigurable systems such as PADDI [1], rDPA [2], DPGA [3], and REMARC [4]. These are described briefly in a following section.

### 1.1. MorphoSys: Reconfigurable architecture

This paper describes **MorphoSys**, a novel architecture for reconfigurable computing systems. This design model,

shown in Fig. 1, involves a reconfigurable SIMD component on the same die with a general-purpose RISC processor, and a high bandwidth memory interface. This integrated architecture model will hopefully provide the potential to satisfy the increasing demand for low cost stream/frame data processing in multimedia applications.

MorphoSys targets applications with inherent parallelism, high regularity, computation-intensive nature and word-level granularity. Some examples of these applications are video compression (DCT/IDCT, motion estimation), template matching, etc. MorphoSys also supports complex bit-level applications such as ATR (Automatic Target Recognition).

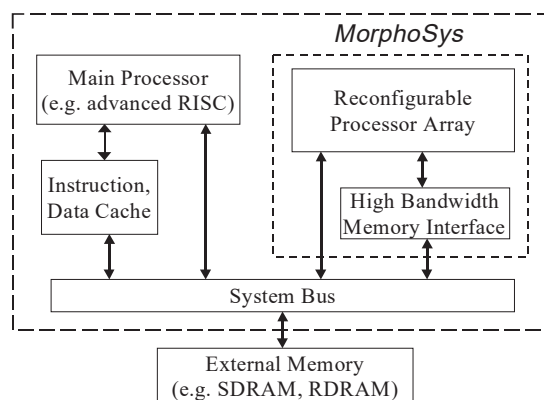


Fig. 1: Integrated architectural model for reconfigurable systems

### 1.2. Organization of paper

Section 2 explains some important terms relevant to reconfigurable computing. A brief review of related research contributions follows. Section 4 introduces the architecture of M1, the first version of MorphoSys reconfigurable system, and describes the reconfigurable cell array and other components. Section 5 describes the physical design process with area and delay estimates. We provide performance estimates in Section 6 for a set of applications (video compression and ATR).

## 2. Important concepts

Different parameters are used to characterize the design of a reconfigurable computing system. Some of these are:

- Granularity (fine or coarse)*: refers to the data size for operations. Bit-level operations (gates or look-up tables) are fine-grain but coarse-grain granularity implies operations on word-size data, using ALUs.
- Depth of Programmability (single versus multiple)*: This pertains to the number of configuration planes resident in a system. Systems with multiple contexts (each of which specifies a configuration plane) may perform different functions without reload of configuration data.
- Reconfigurability (static versus dynamic)*: System reconfiguration is either static (execution is stopped) or dynamic (parallel with execution). Single context systems have static re-configuration. Multi-context systems may be dynamically reconfigurable.
- Interface (remote versus local)*: A local interface implies that the host processor and programmable component reside on the same chip.
- Computation model*: The computation model may be described as either SIMD or MIMD. Some systems may also follow the VLIW model.

## 3. Related research

There has been considerable research effort to develop reconfigurable computing systems. Some of these research prototypes are described below.

- The *Splash* [5], a linear array of processing elements, is useful for linear systolic applications. *DECPeRLe-1* [6] is a two-dimensional array of 16 FPGAs. Both systems are fine-grained, with single configuration and static reconfigurability.
- PADDI* [1] has a set of concurrently executing 16-bit (coarse-grain) functional units (EXUs). *PADDI* targets real-time DSP applications (filters, convolvers, etc.)
- rDPA* [2]: The reconfigurable data-path architecture consists of an array of data-path units (DPUs). Each DPU consists of ALU, micro-programmable control and four registers. It is dynamically reconfigurable.
- REMARC*: This [4] consists of a reconfigurable coprocessor, with 64 programmable units. Each 16-bit unit has a 32 entry instruction RAM, a 16-bit ALU, 16 entry data RAM, instruction register, and several other registers. This system targets multimedia applications.
- DPGA*: A fine-grain prototype system, the Dynamically Programmable Gate Arrays (DPGA) [3] use traditional 4-input lookup tables as the basic array element. Array elements are grouped as sub-arrays that have complete row and column connectivity.

## 4. MorphoSys: System architecture

Fig. 2 shows the organization of the MorphoSys system. It is composed of an array of reconfigurable cells (RC Array) with configuration data memory (Context Memory), a control processor (Tiny RISC), a data buffer (Frame Buffer) and a DMA controller.

### 4.1. System components

*Reconfigurable Cell Array*: The main component of MorphoSys is the 8 x 8 RC (Reconfigurable Cell) Array. Each RC has an ALU-multiplier, a register file and is configured through a 32-bit context word.

*Host/Control processor*: The controlling component of MorphoSys is a 32-bit processor, called Tiny RISC, based on [7]. Tiny RISC handles serial operations, initiates data transfers and controls operation of the RC array.

*Frame Buffer*: The two-set Frame Buffer enables stream-lined data transfers between RC Array and main memory, by overlapping of computation with data load and store, alternately using the two sets.

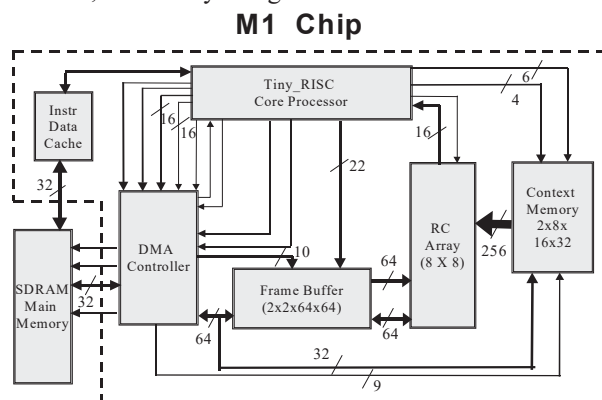


Fig. 2: MorphoSys (M1 chip) components

### 4.2. Salient features of MorphoSys

The RC Array follows *SIMD* model of computation. All RCs in the same row/column share same configuration data. In brief, important features of MorphoSys are:

- Coarse grain*: MorphoSys operates on 8 / 16-bit data.
- Configuration*: RC array is configured by context words, which specify an instruction opcode for RC.
- Depth of programmability*: The Context Memory can store up to 32 planes of configuration.
- Dynamic reconfiguration*: Contexts are loaded into Context Memory without interrupting RC operation.
- Local/Host Processor*: The control processor (Tiny RISC) and RC Array are resident on the same chip.
- Fast Memory Interface*: Through DMA controller.

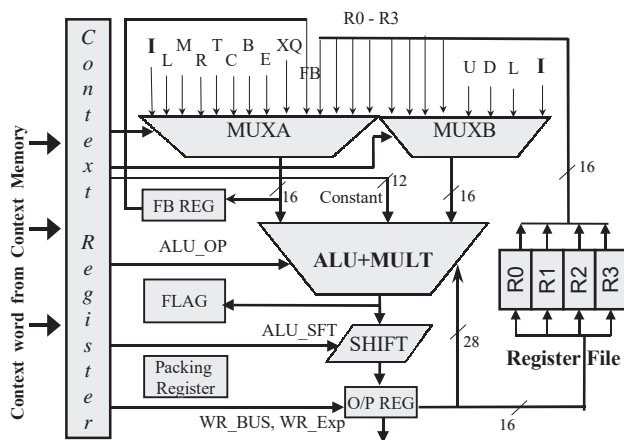
### 4.3. Tiny RISC instructions for MorphoSys

Several new instructions were introduced in the Tiny RISC instruction set for effective control of the MorphoSys RC Array operations. They perform the following functions:

- data transfer between main memory (SDRAM) and Frame Buffer,
- loading of context words from main memory into Context Memory, and
- control of execution of the RC Array.

### 4.4. Architecture of the Reconfigurable Cell

The reconfigurable cell (RC) array consists of an 8x8 array of identical Reconfigurable Cells (RC). As Fig. 3 shows, each RC comprises of an ALU-multiplier, a shift unit, and two multiplexers for ALU inputs. It has an output register, a feedback register, and a small register file. A context word, loaded from Context Memory and stored in the context register (Section 4.5), defines the ALU function, input multiplexer control bits, result storage, and the direction/amount of output shift. The context word can also specify an immediate value.



**Fig. 3: Architecture of a reconfigurable cell**

**ALU-Multiplier:** The ALU has 16-bit inputs, and the multiplier has 16 by 12 bit inputs. Two ALU ports, Port A and Port B are for data from input multiplexers. The third input (12 bits) is from the constant field in the context register (Fig. 4). The fourth port takes input from the output register. The ALU adder is 28 bits wide to prevent loss of precision during multiply-accumulate operation. Besides standard functions, the ALU has several additional functions e.g. absolute value of difference of two numbers and a single cycle multiply-accumulate operation.

**Multiplexer (Input):** The two input multiplexers (Fig. 3) select one of several inputs for the ALU. Mux A (16-to-1) provides values from the array data bus, outputs of the four nearest neighbors and other cells in the same row and

column and the register file. Mux B (8-to-1) takes its inputs from register file, an array data bus and three of the nearest neighbors.

### 4.5. Context Memory

The Context Memory is organized into two blocks (for row and column contexts) with each block having eight sets of sixteen context words. The RC Array configuration plane comprises eight context words (one from each set) from either the row or column block. Thus the Context Memory can store 32 configuration planes or 256 context words.

**Context register:** Each RC is configured through a context word stored in a 32-bit Context Register. It is a part of each RC, whereas the Context Memory is separate from the RC Array. The different fields for the context word are defined in Fig. 4. The field ALU\_OP specifies ALU function. The control bits for Mux A and Mux B are specified in the fields MUX\_A and MUX\_B. Other fields determine the registers to which the result of an operation is written (REG #), and the direction (RS\_LS) and amount of shift (ALU\_SFT) applied to output. The context includes a 12-bit field for the constant.



**Fig. 4: Definition of context word for RC**

The context word also specifies whether a particular RC writes to its row/column express lane (WR\_Exp). Depending upon the context, an RC can access the input of any other RC in its column or row within the same quadrant, or else select an input from its own register file.

**Context broadcast:** The context is broadcast in either of two modes: row-wise or column-wise. Thus, all eight RCs in a row (or column) execute the same context.

**Context Memory organization:** The Context memory has two blocks (one for each broadcast mode), each of which stores eight sets (since there are 8 rows/columns) of sixteen 32-bit context words.

**Selective context enabling:** It is possible to enable one specific row or column for operation in the RC Array. This feature is primarily useful in loading data into the RC Array, but also allows irregular operations in the RC Array, for e.g. zigzag re-arrangement of array elements.

### 4.6. Interconnection network

The RC interconnection network is comprised of three hierarchical levels.

**RC Array mesh:** The primary network in the array is a 2-D mesh that provides nearest neighbor connectivity.

**Complete row/column connectivity:** Within each quadrant, each cell can access the output of any other cell in its row and column.

**Express lane connectivity:** At the global level, there are buses for routing connections between adjacent quadrants. These buses, also called express lanes, run across rows as well as columns. These supply data from any one cell (out of four) in a row (column) of a quadrant to other cells in adjacent quadrant but in same row (column). The express lanes greatly enhance global connectivity. Even irregular communication patterns, that otherwise require extensive interconnections, can be handled quite efficiently. For e.g., an eight-point butterfly is accomplished in three cycles.

**Data bus:** A 128-bit data bus from Frame Buffer to RC array is linked to column elements of the array. It provides two eight bit operands to each of the eight column cells. Eight cycles are required to load the entire RC array.

**Context bus:** The context bus transmits context data to the Context Register in each cell in a row/column depending upon the broadcast mode. Each context word is 32 bits wide, and there are eight rows (columns), hence the context bus is 256 bits wide.

#### 4.7. MorphoSys program flow

The typical operation of MorphoSys M1 system is as follows: The Tiny RISC processor handles the general-purpose operations itself. Specific parts of applications, such as multimedia tasks, are mapped to the RC Array. The Tiny RISC processor initiates the loading of the context words (configuration data) and data from external memory into the Context Memory through the DMA Controller (Fig. 2).

Now, Tiny RISC issues one of several possible context broadcast instructions to start execution of the RC Array. While the RC Array is executing instructions, and using data from the first set of the Frame Buffer, the Tiny RISC initiates transfer of data for the next computation into the second set of the Frame Buffer. When the RC Array execution on the first data set completes, fresh data is available in the second set. Thus the RC Array does not have to wait for data load/store.

### 5. MorphoSys M1: Design implementation

In this section, we describe the physical design of major components of M1: the reconfigurable cell, the frame buffer, DMA controller, Tiny RISC and instr./data caches. The target technology is 0.35 microns (four layer metal). Target cycle time is 10 nanoseconds. The budgeted chip

area is 80 sq. mm. The chip layout will be completed by October, and it will then be fabricated through MOSIS.

#### 5.1. Design methodology

The design approach is two-fold. Components that offer great potential for optimization through a regular structure are custom designed (using Magic V.6.5), while other components, such as controllers, are synthesized with Synopsys behavioral synthesis tools. The resultant netlist is converted into a layout using standard cell libraries and Mentor Graphics tool Autocells. These layouts and custom layouts will then be combined for floor planning and global routing using MicroPlan and MicroRoute tools of Mentor Graphics Corp. The final design will be verified using Lsim simulator from Mentor Graphics.

#### 5.2. Custom component layout

**Reconfigurable Cell:** It is important to optimize the area for the ALU and multiplier, since there are 64 instances of this cell. The 28-bit ALU is designed with a carry-skip adder. Worst case delay is around 2.5 ns. The 16x12 multiplier has a delay close to 3.5 ns. The critical path is through the input multiplexers, multiplier, ALU, other muxes and a logarithmic shifter and adds to approx. 9 ns (within the target of 10 ns). The total area for one RC is close to 0.75 sq. mm.

**Frame Buffer:** This buffer or SRAM array has four banks of 64words by 64 bits. A unit of four SRAM cells was optimized for area, where each SRAM cell uses six transistors. The unit was then instantiated to obtain the array. Each bank has an address decoder (two-level 3 input decoders), input and output latches (64 and 128 bits) and a barrel shifter at the output for byte addressing. The access times of the SRAM array are within 3 ns. The area of the SRAM array and decoder is 0.7 sq. mm. The shifter and latches will need an additional area of 0.15 sq. mm.

**Tiny RISC:** This is implemented as a 32-bit (instruction and datapath) design, with the custom design of instruction cache (2KB), data cache (1KB), register file (16 registers) and ALU. Access times for the caches are 5 ns, while register file access time is 3 ns.

#### 5.3. Behavioral synthesis for controllers

**DMA controller:** This is represented as an FSM with a large number of states, and is suited for synthesis. The synthesis constraints are time cycle of 10 ns and minimum area. Estimated critical path delay is 7.5 ns. The netlist is read into Mentor Graphics tool Autocells, after formatting, and converted into a layout using standard cell libraries.

**Tiny RISC controller:** This will be synthesized using Synopsys tools and converted into layout using Autocells.



## 5.4. Design integration and verification

The custom layouts and synthesized layouts are input into MicroPlan for final placement and output pad connections. The standard cell layouts may be changed for better area utilization, but custom blocks are not affected. Once the floor-plan is accomplished, the design is input to Microroute for routing of global buses and Power and Ground buses from the pads to each block. The composite design is then complete and verified for correctness using Lsim, and through back-annotated synthesis re-runs.

**Table 1: Area and delay estimates for M1 components**

Component Name	Delay (ns)	Area (sq. mm)
Reconfigurable Cell	9.0	0.8
RC Array	--	55
Tiny RISC and caches	9 (max)	8.5
Frame Buffer	4.0	3.5
Context Memory	4.0	4.0
DMA Controller	7.5	7.5

## 6. Mapping applications to M1

In this section, we discuss the mapping of video compression and automatic target recognition (ATR) to the MorphoSys M1 architecture. MPEG standards for video compression are important for realization of digital video services, such as video conferencing, video-on-demand, HDTV and digital TV. The major functions required of a typical MPEG encoder are:

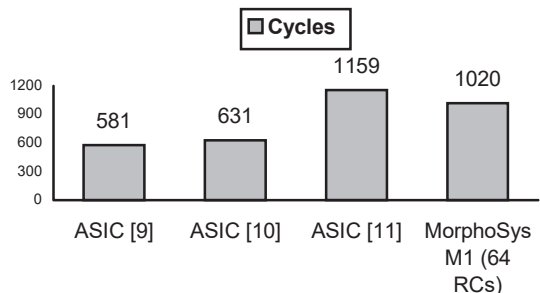
- Motion Estimation and Motion Compensation:
- DCT and Quantization:
- Inverse Quantization and Inverse DCT

### 6.1. Motion Estimation for MPEG

Motion estimation helps to identify redundancy between frames. The most popular technique for motion estimation is the block-matching algorithm [9]. Among the different block-matching methods, full search block matching (FSBM) involves the maximum computations but gives an optimal solution with low control overhead.

**Computation cost:** For a reference block size of 16x16, it takes 36 clock cycles to finish the matching of three candidate blocks. VHDL simulation results show that 5304 cycles are required to finish the matching of the whole search area. If the image size is 352x288 pixels at 30 frames per second (MPEG-2 main profile, low level), processing of an entire image frame would take  $2.1 \times 10^6$  cycles. At clock rate of 100 MHz for MorphoSys, the computation time is  $\approx 21.0$  ms.

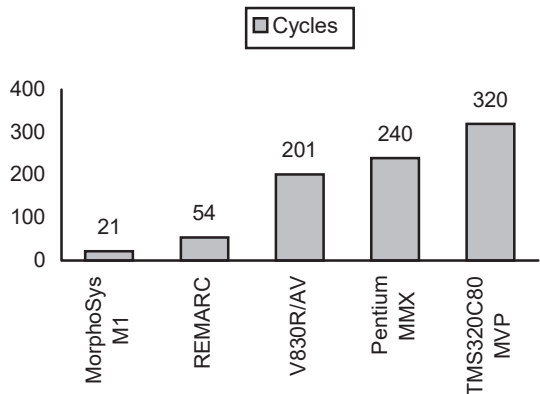
**Performance analysis:** MorphoSys M1 performance is compared with three ASIC architectures implemented in [9], [10], [11] for matching one 8x8 reference block against its search area of 8 pixels displacement.



**Fig. 5: Performance comparison for Motion Estimation**

### 6.2. Discrete cosine transform (DCT) for MPEG

The forward and inverse DCT are used in MPEG encoders and decoders. In the following analysis, we consider an algorithm for fast 8-point 1-D DCT [13]. The eight row (column) DCTs may be computed in parallel. For row (column) mode of operation, the configuration context is broadcast along columns (rows). The coefficients needed for the computation are provided as constants in context words. When 1-D DCT along rows (columns) is complete, the 1-D DCT along columns (rows) are computed in a similar manner.



**Fig. 6: Performance comparison for DCT (cycles)**

**Performance analysis:** MorphoSys requires 21 cycles to complete 2-D DCT (or IDCT) on 8x8 block of pixel data. This is in contrast to 240 cycles required by Pentium MMX™ [12]. The relative performance figures for MorphoSys and other implementations are given in Fig 6.

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

## LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

## FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

## E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.