

**BAA97-46**  
**Section I: Administrative**

Technical Topic Area: Intelligent Memories.

Lead Organization: Obsidian Technology

Type of Business: OTHER SMALL BUSINESS

**Title: rRAM: Intelligent RAM Device. An  
Alternative Solution to Adaptive Computing using  
Conventional LSI Building Blocks**

(A collaborative research program with Obsidian Technology,  
and Staff and Students from the University of California at Irvine)

Costs for Year One	\$361,980
Costs for Year Two	\$438,394
Costs for Year Three	\$179,686
Total Costs	\$980,060
Obsidian Cost Sharing	\$10,000
<b>Total Funds Requested</b>	<b>\$970,060</b>

Technical Contact  
Robert Heaton  
Obsidian Technology  
Phone (714) 363-7982.  
Fax (714) 247-2167.  
Email: rheaton@ece.uci.edu  
37 Niguel Pointe Drive,  
Laguna Niguel  
CA 92677.

Administrative Contact  
Cindy Ragognetti  
Obsidian Technology  
Phone (714) 249-8958.  
Fax (714) 247-2167.  
Email: cindyr@deltanet.com  
37 Niguel Pointe Drive,  
Laguna Niguel  
CA 92677.



**OBSIDIAN TECHNOLOGY**

# Section II Contents

A. Innovative Claims .....	4
B. Technical Rationale.....	5
B.1. Background .....	5
B.2. RISC Cores are Scaling to a Small Size.....	6
B.3. A Note on DRAM .....	6
B.4. What is rRAM? .....	7
B.5. Configuration Switching .....	9
B.6. Example Data Flow .....	9
B.7. The Function of the Global Controller (GC).....	10
B.8. Inter Process Unit Communication .....	10
B.9. How are rRAM Process Units Programmed? .....	11
B.10. Road Map .....	12
B.11. Where Does rRAM Fit? .....	13
B.12. Does the rRAM meet DARPA Goals? .....	13
B.13. Choice of Processor.....	14
B.14. Power Dissipation Issues.....	14
B.15. Plan of Attack.....	15
B.15.1. Device C Code Model .....	15
B.15.2. Architectural Simulation .....	15
B.15.3. Algorithm Mapping.....	16
B.15.4. Programming Tools.....	16
B.15.5. Device Implementation .....	16
B.15.6. Development System Implementation .....	16
B.15.7. Automated Code Partitioning.....	17
B.15.8. Real Application Testing / Proliferation .....	17
B.16. The Origin of the rRAM concept.....	17
C. Deliverables.....	18
C.1. Year One (Architectural Design) .....	18
C.2. Year Two (Device Implementation and Demo Board Development).....	18
C.3. Year Three (Application Testing and rRAM Proliferation).....	18
C.4. Proprietary Claims.....	19
D. Statement of Work (SOW).....	20
D.1. Proposed Tasks .....	20
D.1.1. Architecture Development .....	20
D.1.2. Hardware Implementation.....	20
D.1.3. Programming Tools.....	20
D.2. Contractors .....	21
E. Schedule of Milestones .....	21
F. Technology Transfer.....	21
G. Comparison with other Approaches.....	22
G.1. IRAM .....	22
G.2. Xilinx 6200 Series.....	22
G.3. BRASS .....	23
G.4. MorphoSys (UCI DARPA Funded Program for BAA97-06).....	23



H. Key Personnel .....	24
H.1. Robert Heaton .....	24
H.2. Prof. Nader Bagherzadeh (University of California at Irvine) .....	25
H.3. Dr. Fadi J. Kurdahi (University of California at Irvine) .....	25
H.4. Soheil Shams Ph.D. (BioDiscovery).....	26
H.5. Professor Tomas Lang .....	27
H.6. Muzaffer Kal .....	27
I. Description of Facilities .....	27
J. Costs .....	28
J.1. Summary of Costs .....	28
J.2. Cost Sharing .....	28
J.3. Contract Options .....	28
J.4. Year One Cost Detail .....	29
J.5. Year Two Cost Detail.....	30
J.6. Year Three Cost Detail (6 months of operation).....	31
K. Section III Additional Information .....	32
K.1. References .....	32

## A. Innovative Claims

This proposal seeks to deliver a VLSI device with orders of magnitude increases in the computing performance for two-dimensional processing applications while retaining a maximum use of existing, mainstream, hardware and software architectures. It also offers performance comparable with FPGA and Dynamically Programmable Array architectures without requiring the complexity of hardware place and route support CAD.

The rRAM is a single silicon device with a *hierarchical* array of N by N standard RISC processor cores, a significant quantity of RAM local to each processor, and a fast industry standard bus interface per figure 2. The key innovation in this approach is the hierarchical organization of address and interrupt space such that very high average bandwidth can be achieved between each processor and the RAM, while also maintaining the high IO bandwidth required for such arduous processing applications. This approach has a number of benefits:

- Achieves orders of magnitude reduction in the time required, as compared to FPGA based solutions, to translate software descriptions onto ACS hardware. Translation to hardware objects and placement/routing are not required.
- Takes advantage of spatial address locality of many data processing applications increasing computational speed by almost  $N^2$  over a single processor system.
- The computational model of each CPU is conventional and a standard RTOS may be employed.
- It is scalable. Improved process technologies will offer larger processor arrays and RAMs without changing computing model. We estimate that a 0.08u CMOS process will be able to support 256 ARM RISC processors and a mega byte or more of RAM on a single device at a foundry cost of around \$35.
- The device hardware is simple to implement. Existing RISC cores, RAM, and bus interface cells will be combined with relatively straightforward custom arbitration, interrupt, and DMA logic to form the device.
- Performance, familiarity, cost and simplicity will attract industry adoption. We believe that by providing familiar architectures and standard interfaces in the device, industry and government will be incentivized to adapt existing OS and applications to the rRAM concept.
- It is generalized. While not offering the application specific performance of some finer grained configurable machines [SoftMPEG] the rRAM is a general purpose ACS co-processing element flexible enough for adaptation as a general embedded computing element.
- High memory access bandwidth. Compared with approaches such as IRAM, (See Ref [IRAM]) in which a small number of processing elements access a very large RAM, this approach offers more scalable memory access bandwidth per processor.



## B. Technical Rationale

### ***B.1. Background***

The continuing reduction in silicon device geometries poses problems for the scalability of existing processing architectures and opens great potential for the emergence of radical new architectures. Processing architectures to emerge, which take advantage of increased density, include the following:

- Coprocessors based on conventional FPGA Devices.
- Bit Level Dynamically Programmable Arrays.
- Word level Dynamically Programmable Arrays.
- Intelligent RAM or DRAM Integrations.
- Massively parallel processors. (MPP)

Each approach has its associated characteristics of granularity, context switch speed, support software complexity, and user familiarity of programming model. Broadly, these architectures fall into two categories:

1. Dynamically programmable logic.
2. Multiple processors with access to integrated memory.

Although there is some overlap in these categories, we have chosen to explore the latter in this research.

In choosing an architecture for this proposal we focused particularly on the following DARPA goals (See Ref: [ACS]):

1. Performance of  $7 \times 10^{12}$  IntegerOps/Sec in six devices in order to perform Clear Target template matching by the year 2000.
2. 1,000,000x reduction in device configuration time. (Vs today's FPGAs.)
3. 100x reduction in compilation time.
4. 100x performance gain over contemporary microprocessors.
5. 10x performance gain over contemporary DSP.

It is our contention that the rRAM architecture, though simple to implement, will largely meet these goals.

### B.2. RISC Cores are Scaling to a Small Size

Clearly, for this architecture to be practical, it must be possible to integrate architectures like Figure 2 on a single device. The ARM™ or MIPS™ RISC processor cores are now available from a number of ASIC vendors and have reached a device area of about  $1.5\text{mm}^2 - 2\text{mm}^2$  in  $0.35\mu$  drawn CMOS technology. (See reference [TinyRISC]) Since practical silicon die sizes range up to  $100\text{mm}^2$  there is a strong suggestion that multi RISC arrays are already practical. See Figure 1.

Further, as silicon technologies continue to scale it is clear that large numbers of RISC processor cores can be integrated. For example by the year 2002  $0.08\mu$  CMOS technologies will be available and by this time a RISC processor core will have an area of around  $0.15\text{mm}^2$  clearly suggesting that it will be possible to integrate *hundreds or thousands of conventional RISC* cores on a single device along with supporting circuitry. RAM and DRAM are undergoing even more radical area reductions. (Ref [IRAM])

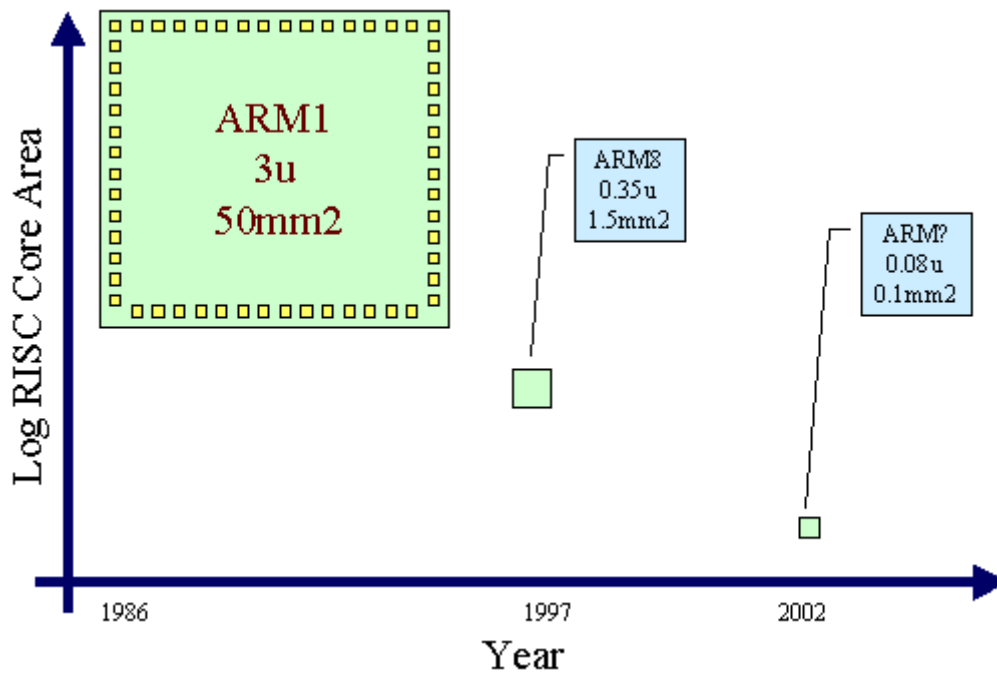


Figure 1. RISC Core Area over Time.

### B.3. A Note on DRAM

Throughout this document we use the term RAM (Random Access Memory). However part of the research will be to determine if Dynamic RAM (DRAM) is a better solution. A number of compromises between density, speed of logic, and RAM access need to be considered.

#### B.4. What is rRAM?

rRAM is an array of conventional RISC processors and RAM on a single silicon device interconnected such that many two dimensional algorithms can take full advantage of their parallelism. In this way the performance of an FPGA coprocessor can be approached without having to map the algorithms into hardware which is a complex and time consuming process. The basic architecture for the proposed experimental device is shown in Figure 2.

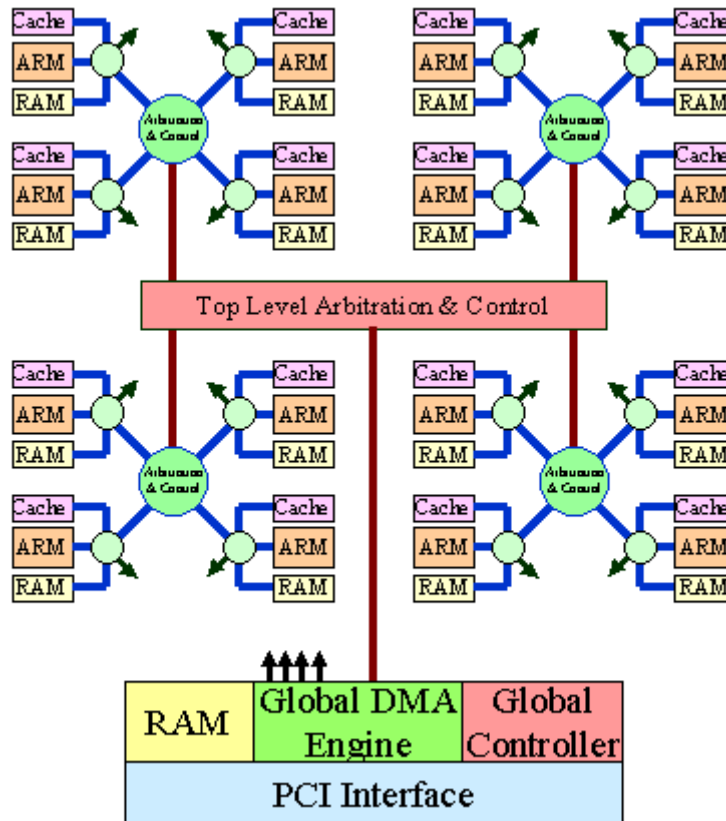


Figure 2. 16 Processor rRAM Architecture Example.

Key to the success of this architecture is access to RAM. Each RISC core has a local RAM or DRAM that can be accessed through arbitration in one of three ways:

1. Local RISC core.
2. Access through a hierarchical arbitration system.
3. Flat access from the host interface though direct memory mapped transfers or through the on chip DMA engine.

The Process Unit (or PU) that constitutes one element of the array is shown in Figure 3. Its arbitration and control unit (AU) is responsible for the following functions:

1. Arbitrating RAM between access sources.

2. Controlling a small instruction cache such that the local RISC core can access instructions while another PU accesses its data.
3. Providing interrupt (and optional synchronization) hardware support.
4. Providing simple programmable memory protections.
5. Provision of address translation such that the local CPU absolute memory vector locations appear in the correct place to the CPU (usually starting address zero) but appear within the allocated space to the top level system. This must be done to ensure each processor has independent indirection vectors.
6. Simple ‘call back’ timers.
7. CPU power management.

Through the arbitration logic each RISC core will be able to “see” the full memory map of the device including the local address space of all the other PUs as shown in the figure. However, the arbitration system is statistically more likely to force wait states for PU accesses outside its local RAM. Statistically the number of wait states required will be proportional to the hierarchical distance of traversal. Hence access within the local 2x2 PU cluster will typically be faster than access through the hierarchy to devices in the local 4x4 PU cluster and so on.

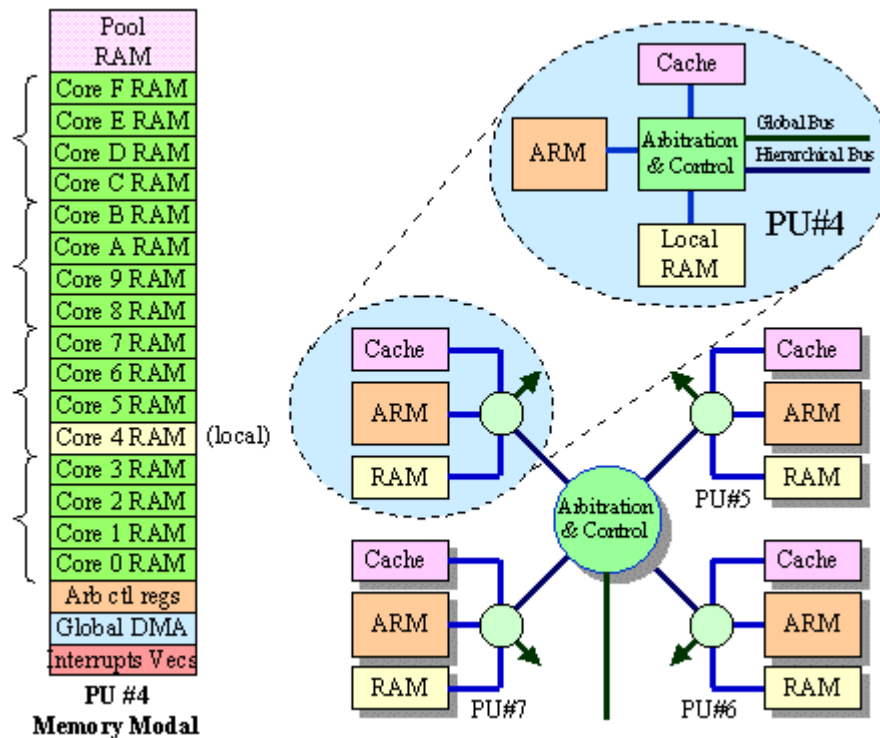


Figure 3. Process Unit (PU) Connections.

We envision that applications which work well with this architecture will have a lot of “spatial locality”. This is to say each PU spends most of its time accessing local program and data space. Fortunately the computationally intensive elements of many two or three-dimensional algorithms require tight nested loops. (Eg, DCT, motion estimation, and DSP

filters). This also allows a very small instruction cache to be effective for reducing local RAM accesses.

### B.5. Configuration Switching

The rRAM is a very flexible element that can change its function on the fly if its application requires it. This makes it possible to *bring the program to the data* rather than the conventional model of bringing the data to the program.

The specific implementation of Configuration Switching will require some research effort. However we know of four different methods for achieving this and we expect to choose some variant of these:

1. Halt the affected processors gracefully. Write a new program start vector to the new routine. Reset the PU.
2. Halt PU. Re-map the program address space for the new code. Local PU reset.
3. IRQ the processor (via a command) and pass it the new routine start address.
4. Halt all PUs. Load new program code. Perform global reset.

These vary in performance, but provided the new configuration is preloaded, methods 1 through 3 will permit configuration switch within an average of about 5 processor cycles, or **50nS** at 100MHz.

### B.6. Example Data Flow

rRAM is designed to be very flexible and it is not limited to a particular data flow model. However, a very simple data flow is shown in Figure 5 as an illustration.

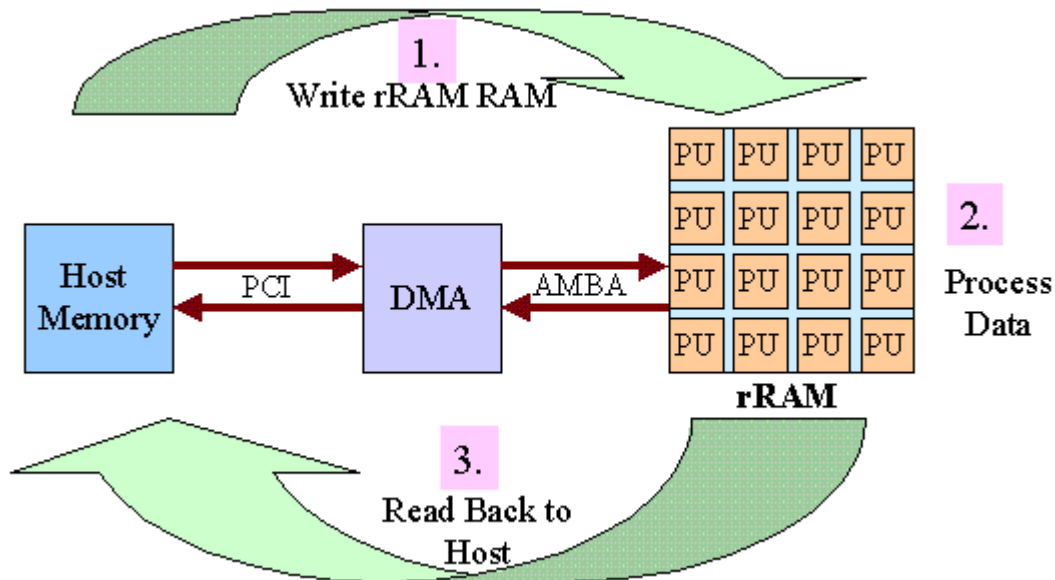


Figure 5. rRAM Example Data Flow.

Step 1. In this example the data from the host is translated through a DMA engine onto the rRAM memory map. This may be under the control of the on chip global processor or the host. A DMA descriptor table in on-chip RAM permits on the fly memory translation to most effectively use PU RAM space. Note that the bandwidth of the PCI bus is high compared to the expected data rates for video and other likely applications.

Step 2. Once the data has been loaded into the rRAM the PUs can be started. Note that this process can be pipelined such that each PU is started by the Global Controller (GC) as soon as its required data is loaded. The code in each processor then executes upon the data in memory. It need never execute any system calls apart from signaling the GC or host. Hence our claim that the architecture is largely OS independent.

The program running on the PU may simply complete (or more likely) signal the Global Controller then stop. At this point the Global Controller may take actions such as signal the PU to change configuration and run a different program on the PUs data.

Step 3. The resulting data or result is sent back through the DMA to host as needed.

### ***B.7. The Function of the Global Controller (GC)***

The rRAM will be memory mapped into PCI host address space like other PCI peripherals. Hence the GC usage will be optional since all its resources are available to the host processor. However, it is included in the overall architecture because it can offload the host CPU of a number of mundane functions which include the following:

- Performing global (top level) actions of an algorithm. I.e. synchronizing and managing the results of the parallel operations in the array.
- Controlling the DMA engine. In order to be successful this architecture must be able to translate complex two dimensional data objects on the fly. While a processor is not strictly necessary (or particularly efficient) for this it does give an added dimension of flexibility to data movement and translation; and of course its silicon area is modest.
- Multiple data or program copies. When a new configuration is to be loaded from the host many applications will require a copy of a similar program in each PU. Using the GC reduces PCI bus traffic.

In many respects the GC will resemble the PU in that it will likely use the same processor and have a similar program environment.

### ***B.8. Inter Process Unit Communication***

This will be a crucial element of the research since the difficulty of programming each PU will depend largely on this. Several mechanisms will probably be supported. While investigating the feasibility of this architecture we considered the following:

- Simple polling via post boxes in conjunction with local PU timers.
- A scheme whereby the GC or the host receives interrupts from, and sends interrupts to, the PUs. Hence the GC would supervise all the inter PU traffic.



- A scheme whereby any PU (or the GC) can send **commands** to any other PU (or the GC).

Of these options, we favor the latter. Inter PU **command** may work as follows if PU#1 is sending a command to PU#2:

1. Within the local address space of each PU, one location is reserved uniquely for a command from every other PU.
2. PU#1 writes a command word into its unique location in PU#2.
3. PU#1 IRQs PU#2 through hierarchical interrupt arbitration. A PU can initiate an IRQ to another PU by writing its memory mapped Arbitration Unit (AU) registers with PU address, command, and IRQ enable bits. A returning ACK appears in AU status space.

### B.9. How are rRAM Process Units Programmed?

rRAM is targeted towards the very computationally intensive parts of algorithms which frequently have small routine size and locality of data. The more complex but computationally less critical algorithmic elements can be run partitioned off to the GC or the host. This leaves the PU programmer with relatively simple task of writing small routines for each PU. For many application the program in each PU will be almost identical.

By design, the programming model for each PU is very simple and consists of a memory map as shown in Figure 6 and a small number of calls to an elementary monitor program we call “miniMon”.

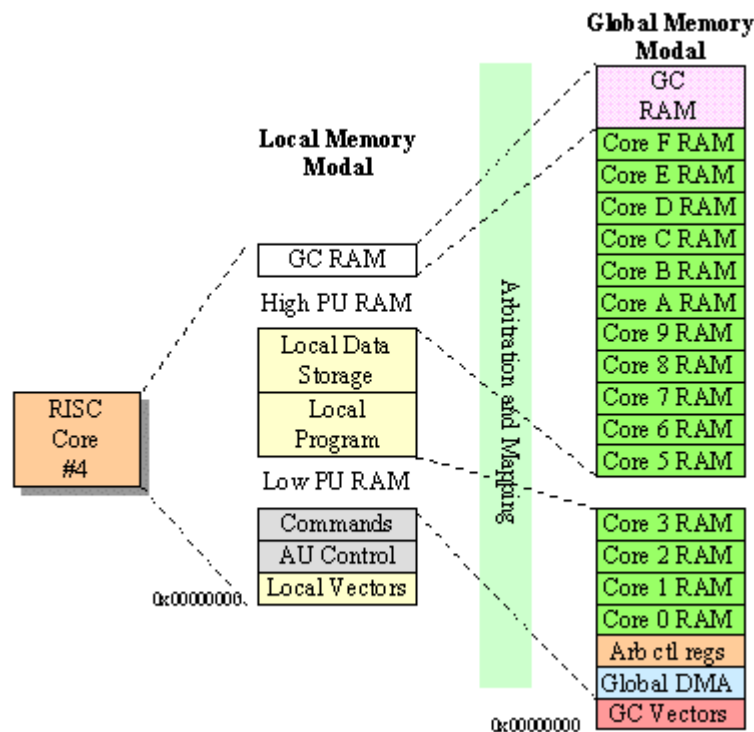


Figure 6. Process Unit Memory Model.

The PU programmer will write C programs which mostly operate within local memory space but can “see” (i.e. access) the whole device memory if the protections are appropriately set. Generally the PU program will not be responsible for the transport of data but will just perform a mundane but computationally intensive processing on data which is moved in and out by the DMA or GC.

This means that the PU programs will generally be rather small. For example a DCT algorithm requires about 130 ARM instructions or about 260 bytes of program. IEEE MPEG2 motion estimations requires about 2K bytes of program space. Hence each PU will often simply have a copy of the program. The remainder of the RAM space can be filled by the programmer with as many (say) 8x8 pixel blocks as will fit.

We expect the “miniMon” which runs on each processor to be very simple and take around 500 bytes of program code. Since calls to the miniMon are rare this code could be made reentrant and be placed in global RAM space. Much research work is required to properly define the miniMon. However in looking at feasibility the following command and routines were considered:

<b>Routine</b>	<b>Function</b>
HaltTillIRQ	Power reduction. Halt the CPU till an IRQ is received.
AckCommand(PUNum , command, seqnum)	Acknowledge a command
SendCommand(PUNum,command,arg)	Send a command to another PU.
AUsetup(...)	Configure the local arbitration unit (AU)

<b>Command</b>	<b>Function</b>
ResetPU(PUNum)	Reset a PU
Reconfigure(PUNum, offset)	Reconfigure (switch PU context via IRQ)
Error(PUNum, type)	Signal a protection error.
MemReq(PUNum, range)	Request graceful memory access to the local RAM space of another PU.
MemRelease(PUNum, range)	Release access to the memory of another PU gracefully.
MemCopy(PUNum,offset,length,R/W)	Graceful memory copy between PUs

Obviously some tools are needed to help the programmer consolidate, load, and check complete rRAM programs. Early in the development a template based C++ code model will be developed with embedded code for moving data, inserting miniMon, and checking for resource violations. The programmer will simply insert the C code for each PU and GC into the template. The whole source can then be compiled, linked, and run on a host machine.

### **B.10. Road Map**

The continuing reduction in CMOS device geometries will permit the rRAM architecture to rapidly evolve in performance per Figure 7.

Device	Process	CPUs	Clock	RAM	Bus IF	Int-Ops/S
rRAM16	0.25u	16	150MHz	256KB	PCI	$4.8 \times 10^9$
rRAM64	0.18u	64	200MHz	1000KB	RBUS	$2.5 \times 10^{10}$
rRAM256	0.08u	256	250MHz	4000KB	RBUS	$1.3 \times 10^{11}$

Figure 7. rRAM Roadmap.

**B.11. Where Does rRAM Fit?**

Per Figure 8, we see rRAM as a coprocessor positioned between conventional microcontroller approaches and the various hardware approaches. Since the designer programs in C code directly we consider that rRAM based applications will, in general, be implemented more rapidly than approaches requiring hardware mapping. rRAMs parallelism should lead to higher performance than conventional microcontroller but not reach the performance of dedicated logic.

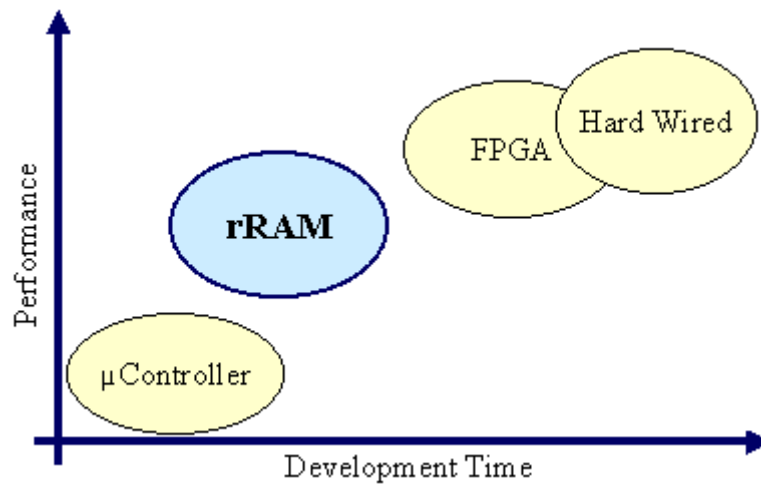


Figure 8. rRAM Positioning.

**B.12. Does the rRAM meet DARPA Goals?**

The following are some estimates (which are open to some debate) on how close the rRAM architecture comes in reaching some selected DARPA goals. In estimating performance in the year 2000 we need to make some assumptions such as 0.2μ five level metal CMOS processing, and 250MHz RISC core performance at 10mW power dissipation. At this process density it is feasible to integrate 128 RISC cores on a single chip. The following comments on the rRAMs ability to meet selected DARPA ACS goals:

- **100x reduction in compilation time.** Compared to today's FPGA technologies, this goal is easily met because a conventional C compiler can be used for rRAM rather than the process of place and route, timing verify, etc, that typically takes several hours or days.
- **1,000,000x reduction in configuration time.** Compared to today's FPGAs this goal is almost met. The rRAM can switch to an internally stored configuration in about 5 clock cycles which at 250MHz is 20nS. RAM based FPGAs require 1-2mS or more to configure because the configuration has to be loaded across a bus. This gives about 100,000x reduction in configuration time. Close to the goal.
- **$7 \times 10^{12}$  IntegerOps/S with six devices.** Given that today's RISC can perform a number of operations in one cycle an assumption is needed. Given an effective instruction cache assume a worst case of 2 integerOps per clock cycle. This translates to  $6.4 \times 10^{10}$  32-bit integerOps/S per chip. Hence 16 devices (rather than 6) are needed to meet the  $7 \times 10^{12}$  integerOps/S goal so this goal is not quite met and further research is needed. See "Choice of Processor" below.
- **100x Performance gain over contemporary microprocessors.** This goal is broadly met due to the processor parallelism and appropriate application.
- **10x performance gain over contemporary DSP.** This is very dependent on interpretation. However, since DSP systems are generally able to partition well in this architecture we think that the 128x parallelism will meet the 10x goal for 32 bit integer DSP systems.

### ***B.13. Choice of Processor***

We are focusing here on the ARM and MIPS architectures here because they are mature architectures with low silicon areas and low power dissipations. They also have a wide support infrastructure in place which is key to the rapid development and deployment of this concept.

That notwithstanding the simulation and modeling of the architecture may point in other directions. For example if a smaller processor unit is employed, such as an 8086 or a 6502 it would be possible to greatly increase the processor array size for a given chip area. This is likely to result in a large increase in integerOps/S per chip.

### ***B.14. Power Dissipation Issues***

Many MIPS and ARM cores have been optimized for battery operated equipment and have power dissipations in the 5-10mW range. [TinyRISC]. They also support zero power halt modes. Hence they are a good choice for large integrations and it may be feasible to integrate 128 processors or more on a single die with power dissipations of around one watt. This is within the power range of low cost plastic packages.

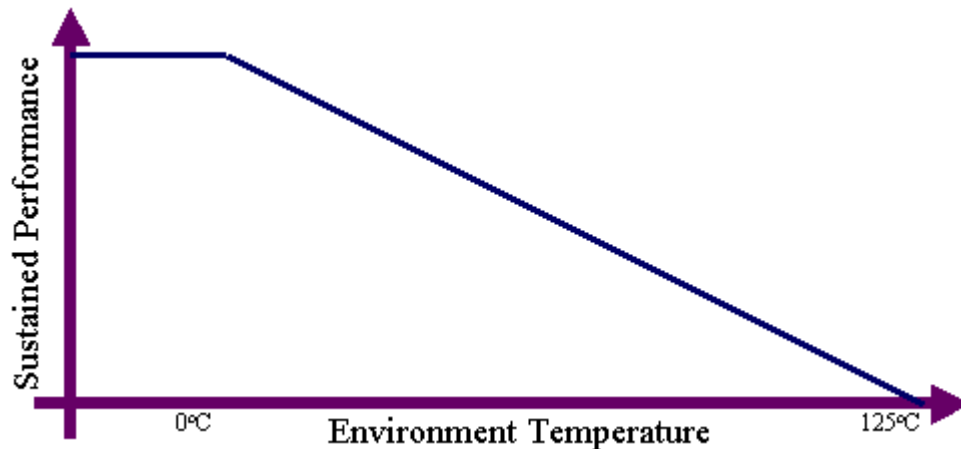
That notwithstanding, we would like to research the concept of "Thermally Constrained Clock Throttling". Statistically microprocessors do useful processing in short bursts since there are typically delays for peripheral accesses or user input. Most CPU's however have to be designed into a thermal environment which assumes maximum environmental temperature



and 100% processor utilization. This generally requires increased cost packaging and/or specially cooling arrangements.

The rRAM will have an integrated temperature sensor. When the die temperature reaches a preset limit the maximum clock rate is reduced. Since maximum ambient temperatures and continuous operation occur infrequently the device can be packaged in standard packaging without significantly reducing performance.

Clock throttling permits the chip to operate to its maximum thermal capability in any environment whereas in conventional processors a significant safety margin is required to ensure functionality over a range of operating environments. This concept is embodied in Figure 9.



**Figure 9. Thermally Constrained Performance.**

### ***B.15. Plan of Attack***

See also Deliverables section. This research is split into the following major elements:

#### **B.15.1. Device C Code Model**

Firstly we will implement a simple device code model to enable us to make early discoveries of functional or performance problems with the rRAM architecture. This will be in the form of a template based C++ program. It will permit us to quickly import real C based algorithms, debug, and run them while estimating real performance.

#### **B.15.2. Architectural Simulation**

A VHDL model of the rRAM16, a 16 PU version of the architecture will be built. Existing models of the processor and PCI bus interface will be used. VHDL models of the DMA Engine and Arbitration Units (AUs) will be coded in synthesizable form.

For verification purposes the simulation and code model results will be compared.

### B.15.3. Algorithm Mapping

The device code model will be used by several students to map various target applications to the architecture and measure performance with respect to other approaches.

### B.15.4. Programming Tools

It will be necessary to adapt an existing RISC C compiler to produce suitable object code for the PUs and the GC. With appropriate switches set we are hopeful that an existing compiler may be used without modification.

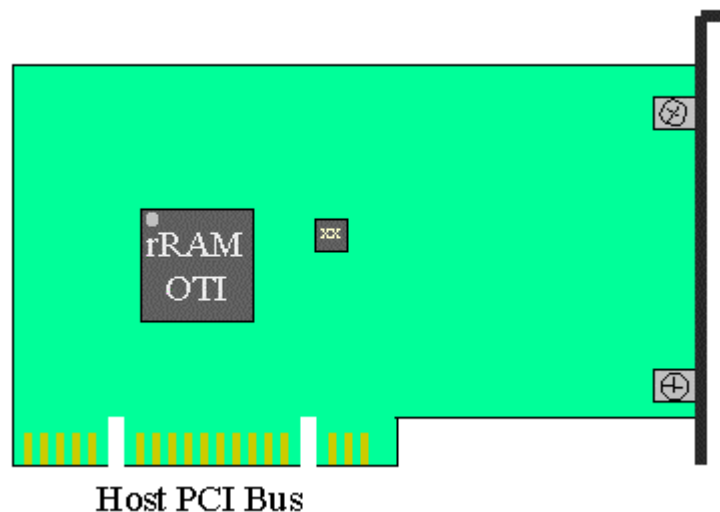
It will be necessary to write a loader to combine PU and GC compiled objects together. However this will be done manually during the initial investigation.

### B.15.5. Device Implementation

With the code model and simulation completed the device will be implemented through an ASIC vendor, probably VLSI Technology. The logic parts of the VHDL model will be synthesized and other cells mapped into the vendors library. A rigorous top down verification will be used ensure first pass success.

### B.15.6. Development System Implementation

A live platform is needed to demonstrate the rRAM device in real time. Initially this will be a simple PCI peripheral board with one rRAM device and a package of basic development software.



**Figure 10. Initial Development Board.**

### B.15.7. Automated Code Partitioning.

This is a well researched area which has experienced only modest success. However, we will research tools for optimizing the partitioning of small programs between PUs although this is a speculative area. We believe enforcing a clear structure on the source could facilitate good results.

### B.15.8. Real Application Testing / Proliferation

The most important indicator of research success is its adoption by Government and Industry. In order to facilitate this, there will be an effort to reach out to find military and commercial applications to implement on the development platform. Hence we expect to work closely with DARPA customers and potential commercial licensors during this activity.

## ***B.16. The Origin of the rRAM concept.***

In researching the most efficient solution for the Morphosys Dynamically Reprogrammable Architecture (DPA), a number of factors favored an increasingly rich matrix processing element. These factors were:

1. In order to have each element in the matrix conditionally and dynamically control its own context, significant control circuitry is required. If the data path is not made rich (i.e. wide and multi functional) then the control circuit, swamps the data area in the results is inefficient use of silicon area.
2. In order to do local conditional operations within each Process Unit the data processed by the unit has to be meaningful. I.e. Byte, word, etc.
3. With increased data width comes another issue. Meaningful operations on this type of data include ALU, shift, and multiply operations. These functions are very inefficient to partition between different PUs. The effect of this is to further increase the most efficient data width so that partitioning will rarely be needed.
4. In considering the applications of the PU it became apparent that many more than four configurations were affordable since each configuration for a wide data path was about 20 bits including control of the routing matrix.

A wide conditional processing unit *is* a CPU! Having reached this point other issues appeared. The PU was now very computationally *rich* but data *poor* since only two or three variables were immediately available to it. This suggested local to give fast access to more data.

Hierarchical memory and interrupt access is an analog of the routing scheme first proposed for Morphosys.



## **C. Deliverables.**

### ***C.1. Year One (Architectural Design)***

During the first year of research a thorough examination of the proposed architecture will be undertaken at a bit true level. The deliverables for the first year are:

1. A C++ based code model system of the rRAM16 to allow program development to begin, and architectural issues to be discovered, very early in the project. Performance monitors will be included in this code.
2. A report detailing the relative performance issue of rRAM vs DPA and other market and research architectures.
3. A full chip bit true VHDL model using existing RISC, RAM, and PCI bus models. Arbitration blocks will be implemented in synthesizable VHDL code.
4. Some primitive translation tools to allow existing RISC compiler object code to be loaded into the models program space.
5. Coding of a basic sample application in C.
6. A high level VHDL test bench which will execute the basic sample application, testing functionality and performance.

### ***C.2. Year Two (Device Implementation and Demo Board Development)***

The deliverable of the first year's work comes close to completion of device development. During the second year a 16 processor rRAM device will be implemented and placed on a PCI circuit board such that it can be tested in a system and provided as a development board to early evaluators and users. Therefore the deliverables are:

1. Work with silicon vendor to implement small quantities of rRAM16 devices.
2. Develop a four layer PCI printed circuit board and metal bracket for the rRAM16.
3. Develop "miniMon" per-CPU primitive monitor and communication program.
4. Basic supporting PC or UNIX system software for loading and executing rRAM16 applications.
5. A complete demo application.
6. Arrange for a DARPA customer to provide a real processing example.

### ***C.3. Year Three (Application Testing and rRAM Proliferation)***

Promotion of the rRAM concept will continue actively through all phases of the project. However during the year the hardware will be available for demonstrating the technology and developing real applications. Improved system software support and development of optimized algorithms for the architecture will be an important aspect of this phase. Hence the deliverables are:

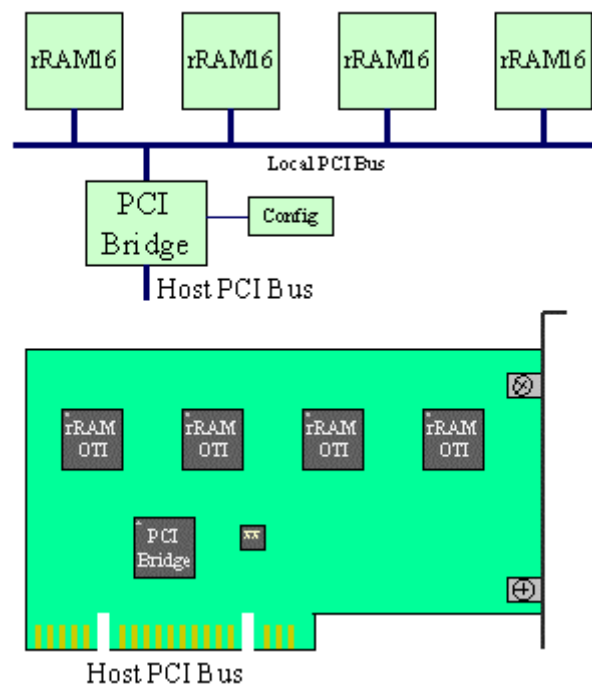
1. Implementation and evaluation of a DARPA customer derived application.



2. Implementation and system support software for a CPU card with 4 rRAM16 chips (That is 64 RISC processors) and a higher integration development board. See Figure 11.
3. Code and simulate architectural refinements for rRAM64 and rRAM128 in readiness for rRAM commercialization.

#### **C.4. Proprietary Claims.**

Obsidian Technology will retain title to the hierarchical arbitration scheme described herein. Also “thermally constrained computing”. Intellectual property rights will be made available on an equal and non-discriminatory basis. Obsidian will also retain ownership of any prototypes and development systems it directly funds.



**Figure 11. 64 Processor Development Board.**

## **D. Statement of Work (SOW)**

See also “Plan of Attack” and “Deliverables”.

### ***D.1. Proposed Tasks***

The development and evaluation proposed here is divided into three main tasks that correspond to the three major components of the proposed project. Each task consists of one or more sub-tasks:

#### **D.1.1. Architecture Development**

The architecture of rRAM will be developed jointly by Obsidian Technology, staff and students from the University of California at Irvine, and Soheil Shams of the BioDiscovery company. The following subtasks will be accomplished:

1. A rRAM C++ template based code model will be developed by Robert Heaton, Nader Bagherzadeh, a consultant, and two students.
2. An HDL model of the architecture will be developed by Dr Fadi Kurdahi, Robert Heaton, a consultant, and two students.
3. Mapping of typical algorithms onto the rRAM architecture and a study of its performance with respect to other architectures. This will be performed by Dr Tomas Lang, Dr Soheil Shams, and supporting students. This activity will continue throughout the length of the program so that applications may be tested in real time.

#### **D.1.2. Hardware Implementation**

A 16 processor demonstration device will be developed by Robert Heaton of Obsidian Technology with the support of students and consultants from UCI. These subtasks will be performed:

1. Conversion of HDL model to ASIC vendor cells.
2. Provision of a fabrication package to the ASIC vendor.
3. Development of a PCI printed circuit board.
4. Evaluation the device resulting in a test report.
5. Revision of the device as necessary.

#### **D.1.3. Programming Tools**

A number of tools will be developed to facilitate in depth testing of the architecture and more rapid commercial and military exploitation. This part of the program will be managed by Fadi Kurdahi of UCI and use graduate student labor. The elements of this will be:

1. Any required modifications to existing RISC compilers to allow use by rRAM.
2. A loader program for rRAM.
3. Research into a pragmatic user guided granularity partitioning tool.
4. The miniMon monitor program to run on each PU.
5. Simple system software to run on the Global Controller.
6. PC or Sun based system evaluation software.



## D.2. Contractors

Aside from the contributors listed under section II.H (Key Personnel) a number of talented Masters students from UCI will be employed for two or three days a week on the project.

## E. Schedule of Milestones

A schedule of milestones is shown below in Figure 12.

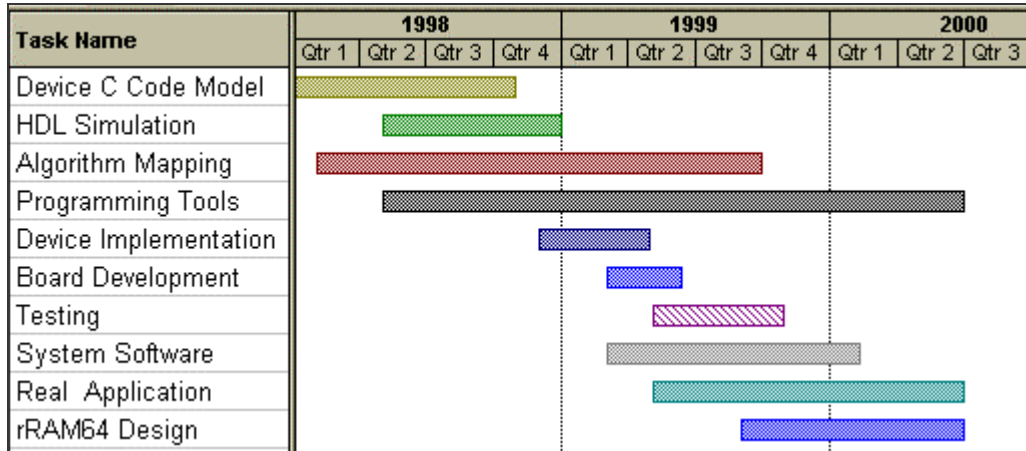


Figure 12. Key Milestone and Activities.

## F. Technology Transfer

The Investigators are committed to using a number of both traditional and novel means to transfer the rRAM. The means include, but are not limited to, the following:

- Development of Key Industrial Sponsors. Obsidian Technology currently has the rRAM concept under review with ARM (Advanced RISC Machines), VLSI Technology, LSI Logic, and Integrated Device Technology. Our object is to have rRAM manufactured and promoted by one of these companies.
- Independent software vendors will also be sought to provide support applications. Discussions are underway with Morphologic who have good background in this area.
- Through an industrial sponsor or through Obsidian Technology DARPA customers within the military will be approached to focus the research on specific real applications.
- Data dissemination through published reports to DARPA, journal and conference publications, and technical reports made available on the WWW.

## **G. Comparison with other Approaches.**

### **G.1. IRAM**

rRAM and IRAM are targeted at rather different applications. IRAM focuses on being a main processor while rRAM focuses on two dimensional and DSP applications as a coprocessor. IRAM does have some similarities with the rRAM in that multiple process units and RAM are arrayed on a single device. However in the current literature IRAM has very large RAMs and a smaller number of process units. We believe this makes it unsuitable for very computationally intense applications with spatial locality. We maintain the advantages of the rRAM are as follows:

1. Higher performance for two dimensional and DSP applications for a given device size. This is because the ratio of processors to memory is higher in the rRAM.
2. rRAM has faster RAM access because each process unit is located physically close to its associated memory.
3. rRAM will use conventional RISC processors leveraging existing tools and programmer training. It is also likely to be available earlier due to the reuse of existing technology.

### **G.2. Xilinx 6200 Series**

The 6200 series is a re-configurable FPGA that can be used as a coprocessor. It is a fine grained FPGA and can be used in similar applications to the rRAM. As a platform for implementing fine grained logic implementations, it exhibits more efficient mapping especially for unconventional data manipulations that take many processor cycles to achieve with a CPU. The 6200 also has flexible FPGA IO cells which give great flexibility of interface. However the rRAM has some significant advantages:

1. rRAM runs compiled C and does not require the process of logic synthesis, placement and routing, timing analysis which requires hours or days for the 6200 series. Hence rRAM is a faster platform for prototype coprocessor development.
2. The rRAM can be reconfigured several orders of magnitude faster than the 6200 because alternative configurations (contexts) can be stored on board the device and switched to very quickly.
3. rRAM has the intelligence to be able to reconfigure itself without intervention by a host processor.
4. rRAM computational elements are usually faster. FPGA logic is slow due to its multiply routed flexibility. RISC processors generally have clocks which run about twice as fast as an FPGA in an equivalent technology.

It is unclear to this stage which approach is denser. The PU elements in the rRAM are less flexible but they are implemented in hand crafted custom logic and hence make up for this somewhat.

### **G.3. BRASS**

The Berkeley BRASS project is a combination of a RISC processor and a configurable computational array. When the application is well tuned to the format of the computational array it can produce performance close to hardwired logic. For such specific, static applications the BRASS architecture is likely to out perform the rRAM as a coprocessor.

The rRAM architecture is focused somewhat differently, however, and has the following relative benefits:

1. rRAM runs compiled C and does not require the process of logic synthesis, placement and routing, and timing analysis which requires significantly more process cycle. Hence rRAM is a faster platform for coprocessor program development. BRASS has an excellent library based approach to reduce compilation time and there are still many degrees of freedom in the program which will make a fast, reliable compiler and hardware mapping very difficult to achieve.
2. More generalized performance. When the BRASS array is not well tuned to the application its performance falls to that of the single RISC processor. On the other hand a wider variety of algorithms will map into multiple processor code although with a fair performance.
3. The configuration speed of the rRAM, just a few clock cycles, is faster making on the fly adaptation more practical.
4. Conditional on the fly adaptation in BRASS would need to be very limited. This is because of the extra compilation overhead to provide timing tested mapping of all possible conditions. In rRAM this is little more than a program branch.

### **G.4. MorphoSys (UCI DARPA Funded Program for BAA97-06)**

Some principal investigators in rRAM are also involved in BAA97-06. MorphoSys has a number of common elements with BRASS. We are attempting to improve the configuration scheme and configuration software. However the qualities of MorphoSys with respect to rRAM are similar to those of BRASS.



## H. Key Personnel.

### ***H.1. Robert Heaton***

Robert Heaton established and managed the VLSI Design Group for Acorn Computers in 1983. He was responsible for all aspects of the group including buying the equipment and software, hiring the staff and establishing the design flow that led the effort to build the ARM, the first RISC processor chip offered for sale. Working with the team, Robert contributed the following specific architectural innovations:

1. Triple ported asymmetrical register file.
2. Semi asynchronous data processing timing flow.
3. Fast high density ALU.
4. Techniques of mixing static and half static logic.

These innovations lead to a leadership position for the ARM architecture in terms of low power and silicon density that persists to this day. The ARM processor and peripheral devices were fully functional at first silicon. The ARM was implemented with an effort of just five man years. Robert not only lead the team but was an active member of it, designing and laying out many of the key cells personally. These included the ALU, the register file, the control PLA's, the status control and address generation logic.

Since leaving Acorn during 1986, Robert has established and managed a number of engineering groups and was Senior Director of Engineering for Standard Microsystems from 1990 to 1996 where he managed a staff of 60. During this time he established a close relationship with the engineering staff in the Advanced Computer Architecture group at UC Irvine and is currently Chairman of UCI's Engineering Corporate Affiliates group.

Robert has continued to be a technical innovator with a Geometric Compression US patent #5,216,726 and a Fast Ethernet Local Area Network US patent #5,544,323. It was while working in a networking environment that Robert noticed the limitations of existing processor architectures. He established Obsidian Technology as an individual consultant during 1996 and has taken consulting contracts with Rockwell Semiconductor Systems and Standard Microsystems Corp. Additional help for this project will be drawn from UCI staff and students in the form of constancy and part time employment.

In March of this year Robert (and Obsidian Technology) was awarded a BAA 97-06 grant together with his associates at UCI.

Robert will lead the research and be supported for 24 hours per week during the length of the project. He is committed to about 12 hours per week on project BAA97-06 and has a pending proposal for a 128-bit RISC architecture at NSF.

For more information: <http://members.home.com/heaton/resume.html>



## ***H.2. Prof. Nader Bagherzadeh (University of California at Irvine)***

Nader Bagherzadeh will be employed as a consultant to support the development of the architecture and to supervise students that work as consultants at the UCI campus. He is budgeted for 20 days per year during this research. Nader is also supported by BAA97-06 and has a 20 days/year role in a pending NSF solicitation.

Nader has been involved in the design of advanced microprocessor architectures for the past ten years. He was the leading designer of the first 4-issue VLIW microprocessor called VIPER. The VIPER architecture demonstrated pioneering work in the areas of processor optimization for routing operand data through bypass circuitry. Moreover, it utilized a unique pipeline architecture with a refined addressing mode that mitigated the development of future VLIW processors by industry. Later on he worked on all out of order issue Superscaler microprocessor called SDSP targeted for multimedia computation intensive applications. This study led to several new discoveries in the areas of instruction fetching and register renaming hardware optimization.

Currently he has been combining his experience with out-of-order issue Superscaler with multi threaded architectures. In this work he is the first researcher that demonstrated the design of a register renaming scheme based on the reorder buffer and instruction window that call handle multiple threads. This work will open the way for exploiting instruction level parallelism beyond the wide issue Superscaler model. Finally, he is involved in the hardware design of a 500Mhz reorder/buffer that utilizes the True Single Phase clocking scheme.

For more information: <http://www.eng.uci.edu/comp.arch/nader/index.html>

## ***H.3. Dr. Fadi J. Kurdahi (University of California at Irvine)***

Fadi Kurdahi will be employed as a consultant to support the team's computer aided design (CAD) requirements and to supervise graduate students developing VHDL models for the project. He is budgeted for 20 days per year during the research. Fadi is also supported by BAA97-06 and has a 20 days/year role in a pending NSF solicitation.

Fadi Kurdahi has been doing research in Design Automation and VLSI design for over 13 years. Specifically, he has researched and developed CAD tools for Architectural synthesis and estimation and has published numerous papers on the subject of linking layout information to synthesis. This becomes particularly crucial for the projected fabrication technologies that are likely to implement the proposed DRA. Recently, his work has focused on FPGAs as target implementations. His work on estimation for FPGA resulted in a suite of prediction tools that are highly accurate yet runtime efficient. Architectural synthesis techniques such as scheduling and binding are being developed to utilize these estimator and generate "first-time-correct" silicon.

Since 1987, he has been a faculty at the Department of Electrical & Computer Engineering at the University of California, Irvine, where he is currently an Associate Professor. His research interests are in the areas of Computer-Aided Design of VLSI circuits, high-level synthesis, and design methodology of large scale systems. His research addresses the issues



of linking High Level Synthesis to subsequent levels of design, namely logic and physical levels. He has published papers dealing with various aspects of chip and system synthesis at international conferences and journals. He has organized and participated in tutorials on high level synthesis at international conferences such as the Design Automation Conference in 1990. Prof. Kurdahi was an Associate Editor for IEEE Transactions on Circuits and Systems II in 1993-1995, and was a guest editor for a special issue of the VLSI Design Journal on "Linking Behavioral, Structural and Physical Models of Hardware". He was Organization Chairman of the Sixth International High Level Synthesis Workshop, and served on the program committees of several conferences and workshops such as: the international High Level Synthesis Workshop, the international Symposium OIL System Level Synthesis, the ACM/IEEE Physical Design Workshop, and the European Design and Test Conference. He received the Research Initiation Award from the National Science Foundation in 1989, and the ACM/SIGDA Fellowship in 1991 and 1992. Prof. Kurdahi is a member of IEEE and ACM.

For more information: <http://www.eng.uci.edu/faculty/kurdahi/fadi.html>

#### **H.4. Soheil Shams Ph.D. (BioDiscovery)**

Soheil Shams will be employed as a consultant to support the team's investigations into mapping both new and conventional array mapped algorithms onto the rRAM architecture. Soheil's small company, BioDiscovery, is engaged in algorithm research for extracting gene information (and other data mining) from images. As such, BioDiscovery is a potential user of the rRAM PC accelerator board. Soheil is budgeted for 50 days per year during the research.

Dr Shams has an extensive background in image processing. Working for Hughes Space & Communications over a ten-year period he was the Principle Investigator for numerous Target Tracking, Target Recognition, and data encryption projects. He has a number of patents pertinent to this project which include:

1. S. Shams, "Target Deghosting and Tracking Using the Multiple Elastic Feature Nets Algorithm" (Patent filed 1994).
2. S. Shams & D. Shu, "A Dynamically Reconfigurable Switch Design for Interprocessor Communications in Multi-Processing Systems" (Patent filed 1993).
3. S. Shams, "A Variable Accuracy Indirect Addressing Scheme For SIMD Multi-Processors", U.S. Patent No. 5,526,501, Issued 6/11/96.

Soheil has been much honored for his work including Hughes Outstanding Paper Award for 1996 and the Hughes Electronics Innovation and Excellence Award for 1996. He also presents classes at UCLA for Machine Perception and Artificial Intelligence.

Among Soheil's many publications are the following:

1. S. Shams, "A Self-Organizing Model for Multi-Invariant Object Recognition," *Proc. of the Engr. Applications of Neural Networks Conf.*, London, UK, pp. 359-362, 1996.



2. S. Shams, "Simultaneous Recognition of Multiple Objects Using the MEM Model", *Proc. of the Inter. Workshop on Artificial Neural Networks '95*, Malaga, Spain. pp. 919-925, 1995.
3. S. Shams and K. W. Przytula. "Implementation of Multilayer Neural Networks on Parallel Programmable Digital Computers." In Parallel Algorithms and Architectures for DSP Applications. Ed. M. Bayoumi, Kluwer Academic Publishers, pp. 225-253, 1991.

### **H.5. Professor Tomas Lang**

Tomas Lang is a Professor in the Department of Electrical and Computer Engineering at the University of California, Irvine. Previously he was a Professor in the Computer Architecture Department of the Polytechnic University of Catalonia, Spain, and a faculty member of the Computer Science Department at the University of California, Los Angeles. He received an Electrical Engineering degree from the Universidad de Chile in 1965, a M.S from the University of California (Berkeley) in 1966 and the Ph.D. from Stanford University in 1974. Dr. Lang's primary research and teaching interests are in digital design and computer architecture with current emphasis on high-speed and low-power numerical processors and multiprocessors. He is co-author of two textbooks on digital systems, two research monographs, one IEEE Tutorial, and author or co-author of research contributions to scholarly publications and technical conferences.

Dr. Lang has performed professional consulting for IBM's Scientific Center, Los Angeles, for Hughes Aircraft Co. and for Jet Propulsion Labs. He has taught courses to professionals in the areas of computer architecture and design and of high-performance computer systems for UCLA Extension, University Consortium for Continuing Education, IBM, Burroughs, and Vitesse.

### **H.6. Muzaffer Kal**

Muzaffer Kal has extensive experience with UNIX and windows operating systems device drivers gain while in the employ is Microsoft. He also has extensive experience with silicon development in the networking World as a Principle of Soft-Mixed-Signal Corp. He brings a wealth of practical software and hardware skills to the project of which he will be a key member. Additionally he has practical high speed adaptive equalizer, DFE, and DSP background.

## **I. Description of Facilities**

The Facilities to be used are those available at the premises of Obsidian Technology and those available to the consultants and students. Should these prove to be inadequate contract services from UCI will be purchased for access to high-end computer servers and other equipment.

Obsidian Technology has three workstations, a 500Kb/S internet data link, a complete set of Tanner IC design tools, C++ compilers and ancillary equipment. QuickHDL™ from Mentor



will be used for in house VHDL modeling. An ASIC vendor's software and facilities will be used for converting the VHDL implementation into a fabrication package.

## **J. Costs**

### ***J.1. Summary of Costs***

Costs for Year One	\$361,980
Costs for Year Two	\$438,394
Costs for Year Three	\$179,686
Total Costs	\$980,060
Obsidian Cost Sharing	\$10,000
<b>Total Funds Requested</b>	<b>\$970,060</b>

### ***J.2. Cost Sharing***

Obsidian Technology will contribute \$10,000 towards the total cost of the program.

### ***J.3. Contract Options***

It is difficult to separate the major elements of the project (hardware, software and algorithm development). However, the first year is most separable from the remainder of the project since the deliverables constitute a thorough development of the concept and a complete high level design. Hence this is a potential funding option.

**J.4. Year One Cost Detail**

		Hours	Rate	O/H Rate %	O/H	Total
<b>Staff</b>	PI	1200	\$70	10.00%	\$8,400	\$92,400
	Staff Engineer	600	\$50	5.00%	\$1,500	\$31,500
	Consultant1	400	\$70	5.00%	\$1,400	\$29,400
	Consultant2	400	\$70	5.00%	\$1,400	\$29,400
	UCI Staff1	160	\$70	51.00%	\$5,712	\$16,912
	UCI Staff2	160	\$70	51.00%	\$5,712	\$16,912
	UCI Student1	960	\$18	51.00%	\$8,813	\$26,093
	UCI Student2	960	\$12	51.00%	\$5,875	\$17,395
	UCI Student3	200	\$12	51.00%	\$1,224	\$3,624
	Student4	960	\$18	5.00%	\$864	\$18,144
	Student5	960	\$15	5.00%	\$720	\$15,120
	Student6	400	\$15	5.00%	\$300	\$6,300
	Admin1	120	\$30	5.00%	\$180	\$3,780
	<b>Total Staff</b>					<b>\$306,980</b>
<b>Equipment</b>	Workstation Upgrades					\$7,000
<b>&amp; Misc</b>	CAD Software					\$16,500
	PCI Bus Model					\$18,000
	Material and Supplies					\$3,500
	<b>Total Equipment &amp; Misc.</b>					<b>\$45,000</b>
<b>Travel</b>	DAPRA Conferences etc.					\$10,000
<b>Total for Year 1</b>						<b>\$361,980</b>
Note: 51% overhead is the rate charged by University (UCI) for contracts.						

**J.5. Year Two Cost Detail**

		<b>Hours</b>	<b>Rate</b>	<b>O/H Rate %</b>	<b>O/H</b>	<b>Total</b>
<b>Staff</b>	PI	1200	\$70	10.00%	\$8,400	\$92,400
	Staff Engineer	600	\$50	5.00%	\$1,500	\$31,500
	Consultant1	400	\$70	5.00%	\$1,400	\$29,400
	Consultant2	400	\$70	5.00%	\$1,400	\$29,400
	UCI Staff1	160	\$70	51.00%	\$5,712	\$16,912
	UCI Staff2	160	\$70	51.00%	\$5,712	\$16,912
	UCI Student1	960	\$18	51.00%	\$8,813	\$26,093
	UCI Student2	960	\$12	51.00%	\$5,875	\$17,395
	UCI Student3	200	\$12	51.00%	\$1,224	\$3,624
	Student4	960	\$18	5.00%	\$864	\$18,144
	Student5	960	\$18	5.00%	\$864	\$18,144
	Student6	400	\$15	5.00%	\$300	\$6,300
	Admin1	180	\$30	5.00%	\$270	\$5,670
	<b>Total Staff</b>					<b>\$311,894</b>
<b>Equipment</b>						
<b>&amp; Misc.</b>	Device Fabrication					\$85,000
	Fab Design services					\$28,000
	Material and Supplies					\$3,500
	<b>Total Equipment &amp; Misc.</b>					<b>\$116,500</b>
<b>Travel</b>	DAPRA Conferences etc.					\$10,000
<b>Total for Year 2</b>						<b>\$438,394</b>
Note: 51% overhead is the rate charged by University (UCI) for contracts.						

**J.6. Year Three Cost Detail (6 months of operation)**

		Hours	Rate	O/H Rate %	O/H	Total
<b>Staff</b>	PI	700	\$70	10.00%	\$4,900	\$53,900
	Staff Engineer	120	\$50	5.00%	\$300	\$6,300
	UCI Consultant1	300	\$70	5.00%	\$1,050	\$22,050
	UCI Staff1	100	\$70	51.00%	\$3,570	\$10,570
	UCI Staff2	100	\$70	51.00%	\$3,570	\$10,570
	UCI Student1	460	\$18	51.00%	\$4,223	\$12,503
	UCI Student2	460	\$12	51.00%	\$2,815	\$8,335
	UCI Student3	200	\$12	51.00%	\$1,224	\$3,624
	Student4	460	\$18	5.00%	\$414	\$8,694
	Student5	460	\$15	5.00%	\$345	\$7,245
	Student6	460	\$15	5.00%	\$345	\$7,245
	Admin1	100	\$30	5.00%	\$150	\$3,150
	<b>Total Staff</b>					<b>\$154,186</b>
<b>Equipment &amp; Misc</b>						
	PCB Boards					17000
	Material and Supplies					3500
	<b>Total Equipment &amp; Misc.</b>					<b>\$20,500</b>
<b>Travel</b>	DAPRA Conferences etc.					\$5,000
	<b>Total for Year 3 (6 months)</b>					<b>\$179,686</b>
	Note: 51% overhead is the rate charged by University (UCI) for contracts.					

## K. Section III Additional Information

### K.1. References

[SoftMPEG]

Performance of software-based MPEG-2 video encoder on parallel and distributed systems. Author(s): Akramullah, Shahriar M.; Ahmad, Ishfaq; Liou, Ming L. Source: IEEE Transactions on Circuits and Systems for Video Technology Aug 1997 IEEE Piscataway NJ USA p 687-695 ISSN: 1051-8215 CODEN: ITCTEM

[TinyRISC]

LSI Logic Introduces A COMPRESSED CODE 16/32-bit TinyRISC™ Microprocessor. Press release: <http://www.lsil.com/mediakit/pr0028.html>.

[IRAM]

IRAM Project. See IRAM Talks and Publications:  
<http://iram.cs.berkeley.edu/publications.html>  
<http://infopad.eecs.berkeley.edu/~pering/iram/project3.html>

[CLAY]

National Semiconductor ACS Architecture:  
<http://www.national.com/appinfo/milaero/napa1000/index.html>

[BRASS]

Berkeley Project. <http://HTTP.CS.Berkeley.edu/projects/brass/>

[ACS]

DARPA Adaptive Computer Systems Challenge Problems:  
<http://www.ito.darpa.mil/ResearchAreas96/ACSchallengeproblems.html>

[MIT]

Dynamically Programmable Gate Arrays: A Step Toward Increased Computational Density (FPD '96). [http://www.ai.mit.edu/projects/transit/dpga\\_prototype\\_documents.html](http://www.ai.mit.edu/projects/transit/dpga_prototype_documents.html)

