THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON DC

CHREC
NSF Center for High-Performance Reconfigurable Computing

High-Performance Reconfigurable Computing

Tarek El-Ghazawi

Director, Institute for Massively Parallel Applications and Computing Technology (IMPACT)
Co-Director, NSF Center for High-Performance Reconfigurable Computing (CHREC)

The George Washington University

Acknowledgements

- ARSC, AMI, Cray, DoD, HPTi, NASA, NSF/CHREC, SGI, SRC, Star Bridge, Xtreme Data, many others

Outline

- Architectures and Systems
- Tools and Programming
- Applications
- Performance
- Wrap-up

Reconfigurable Supercomputing (RSC)

- Efficient high performance computing using parallel and distributed systems of both reconfigurable hardware resources and conventional microprocessors
- This tutorial establishes the current status, the direction taken, and the potential for RSC

Top 500 Supercomputers

Rank	Site	Computer	Processors	Year	R_{max}	R_{peak}
1	DOE/NNSA/LLNL United States	eServer Blue Gene Solution IBM	212992	2007	**478200**	596378
2	Forschungszentrum Juelich (FZJ) Germany	Blue Gene/P Solution IBM	65536	2007	**167300**	222822
3	SGI/New Mexico Computing Applications Center (NMCAC) United States	SGI Altix ICE 8200, Xeon quad core 3.0 GHz SGI	14336	2007	**126900**	172032
4	Computational Research Laboratories, TATA SONS India	Cluster Platform 3000 BL460c, Xeon 53xx 3GHz, Infiniband HP	14240	2007	**117900**	170880
5	Government Agency Sweden	Cluster Platform 3000 BL460c, Xeon 53xx 2.66GHz, Infiniband HP	13728	2007	**102800**	146430

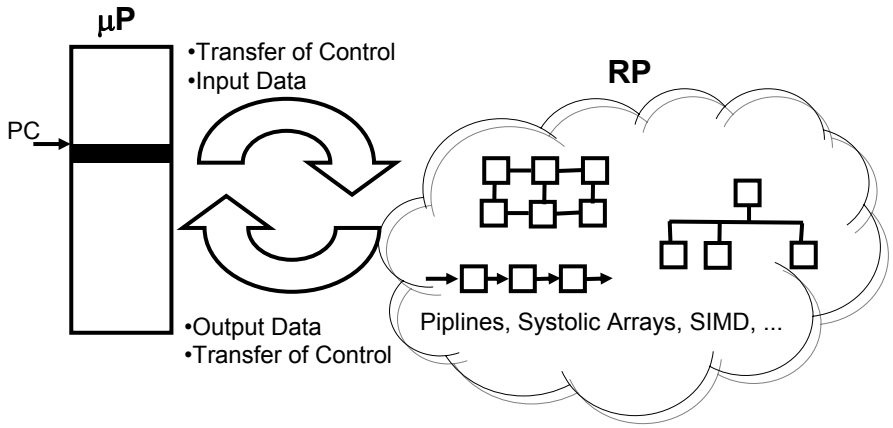
Reconfigurable Computers

The microchip that rewires itself

Source: [Sci97]

- **Scientific American – June 1997**
  - **Computers that modify their hardware circuits as they operate are opening a new era in computer design.**
  - **Reconfigurable computers architecture is based on FPGAs (Field Programmable Gate Arrays)**

Execution Model for HPRCs

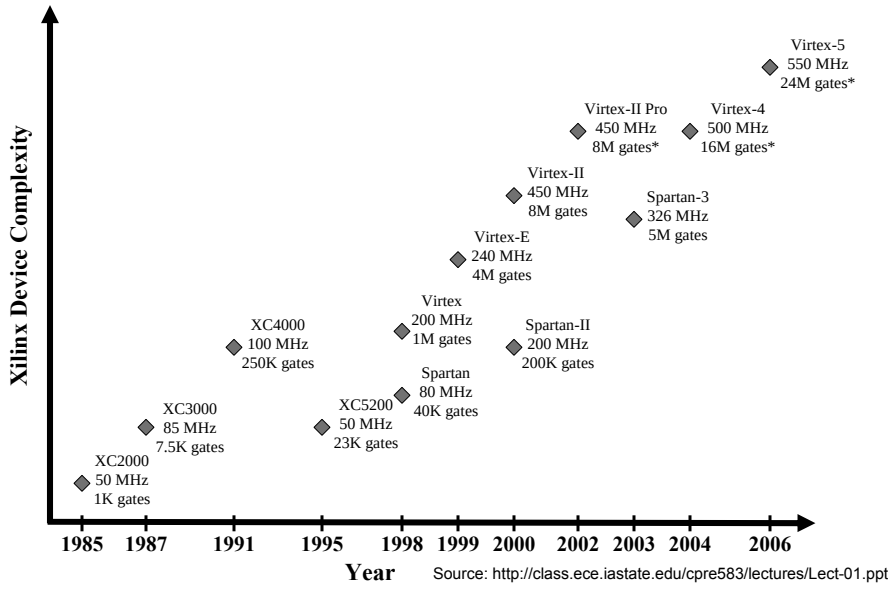


- **Fine grain computations with the RP, others with the MP**
- **Interaction between RP and MP can be blocking or asynhronous**
- **This scenario is replicated across the whole system and standard HPC parallel programming paradigms used for interactions**

Synergism between μP and RPs

	μP	RP(FPGA-based)
Processing Style	Software→Control Flow (von Neumann) Temporal – reuse of fixed hardware	Hardware→Data Flow Spatial – Unfolding parallel operations with changeable hardware
Parallelism Exploited	Coarse-Grain	Fine-Grain
Clocking Rate	Very Fast Saturating Rate	Relatively Slow Increasing Speed
Applications Partitioning	Relatively Easy (S.W./Parallel Programming)	Harder
Commercial Availability	COTS, multipurpose	COTS, multipurpose

Capacity Trends



Source: http://class.ece.iastate.edu/cpre583/lectures/Lect-01.ppt

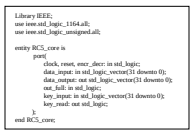
WHAT'S NEW IN THE VIRTEX-5 FPGA FAMILY

Feature/capability *LX Platform*	Virtex-5 family	Virtex-4 family	Virtex-5 benefit
Process Technology	65nm, 1.0v V_{cc} Triple-oxide	90nm, 1.2v V_{cc} Triple-oxide	Higher density and performance with lower power and cost
LUT	Real 6-input LUT with 6 independent inputs	4-input LUT	Fewer logic levels—higher density and speed and lower power
Distributed RAM	256 bits per CLB	64 bits per CLB	More memory
Shift Registers (SRL)	128-bit in one CLB	64-bit in one CLB	Deeper pipelines
Interconnect	New diagonal routing	Segmented routing	Fast, predictable routing
Clock Management	550 MHz PLL and DCM	500 MHz DCM	Higher speed PLL: lower jitter DCM: flexible clock synthesis
Block RAM/FIFO with ECC	550 MHz 36 Kbits per block (2 x 18Kb) with power saving circuits	500 MHz 18 Kbits per block	Higher speed More memory, low power
DSP Blocks	550 MHz 25 x 18-bit MAC, plus bit-wise comparator 1.38 mW/100MHz @ 38% toggle rate	500 MHz 18 x 18-bit MAC 2.3 mW/100MHz @ 38% toggle rate	Higher performance Higher precision using 50% fewer slices Lower power

The Design Cycle
(That we want you to avoid !)

Design and implement a simple encryption unit with RC5 cipher with fixed key

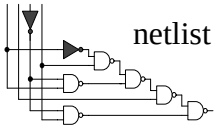
Specification



HDL (Hardware Description Language) model

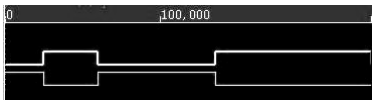
Functional simulation

Synthesis



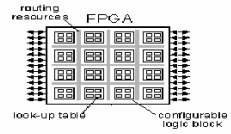
netlist

Post-synthesis simulation



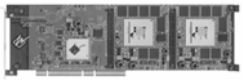
The Design Cycle
(That we want you to avoid !)

Implementation (Mapping, Placing & Routing)



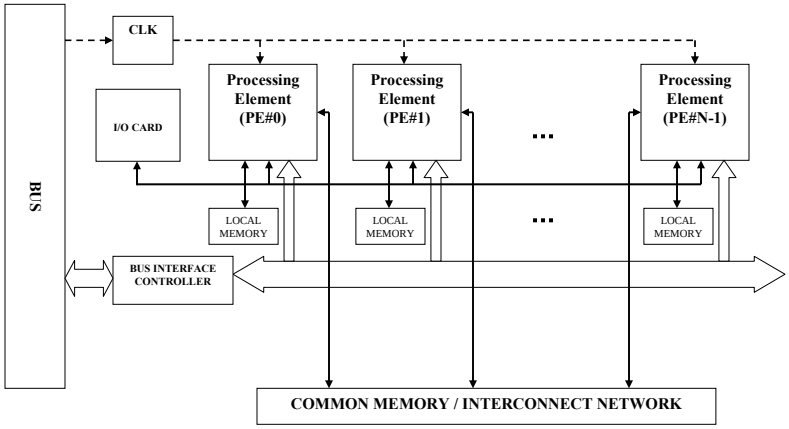
Timing simulation

Downloading and Testing



On board testing

General Architecture of an FPGA-Based Board



Reconfigurable Computing Boards
(Accelerators)

- Boards may have one or several interconnected FPGA chips
- Support different bus standards, e.g. PCI, PCI-X, VME
- May have direct real-time data I/O through a daughter board
- Boards may have local onboard memory (OBM) to handle large data while avoiding the system bus (e.g. PCI) bottleneck

Reconfigurable Computing Boards
(Accelerators)

- **Many boards per node can be supported**
- **Host program (e.g. C) to interface user (and μP) with board via a board API**
- **Driver API functions may include functionalities such as Reset, Open, Close, Set Clocks, DMA, Read, Write, Download Configurations, Interrupt, Readback**

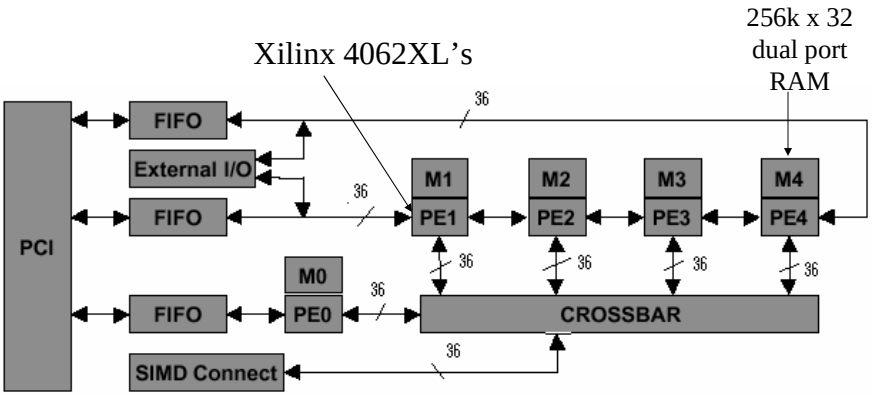
Some Reconfigurable Boards Vendors

- ANNAPOLIS MICRO SYSTEMS, INC**. (http://www.annapmicro.com)**
- University of Southern California -USC/ISI **(http://www.east.isi.edu)**
- AMONTEC **(http://www.amontec.com/chameleon.shtml)**
- XESS Corporation **(http://www.xess.com)**
- CELOXICA **(http://www.celoxica.com)**
- CESYS **(http://www.cesys.com)**
- TRAQUAIR **(http://www.traquair.com)**
- SILICON SOFTWARE: **(http://www.silicon-software.com)**
- ALPHA DATA: **(http://www.alpha-data.com)**
- Associated Professional Systems: **(http://www.associatedpro.com)**
- NALLATECH: **(http://www.nallatech.com)**

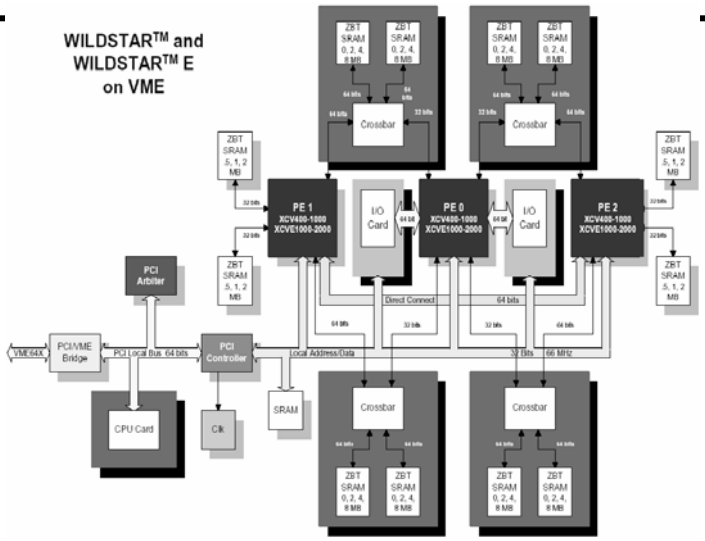
Representative Example Boards

From Annapolis Micro Systems (AMI)
http://www.annapmicro.com
&
Nallatech
http://www.nallatech.com

WILDFORCE™

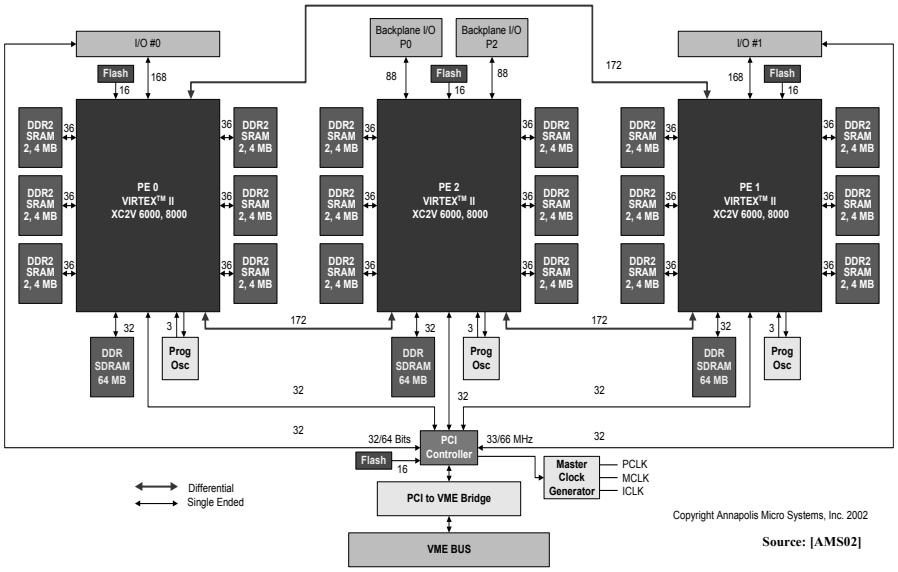


Source: [AMS02]



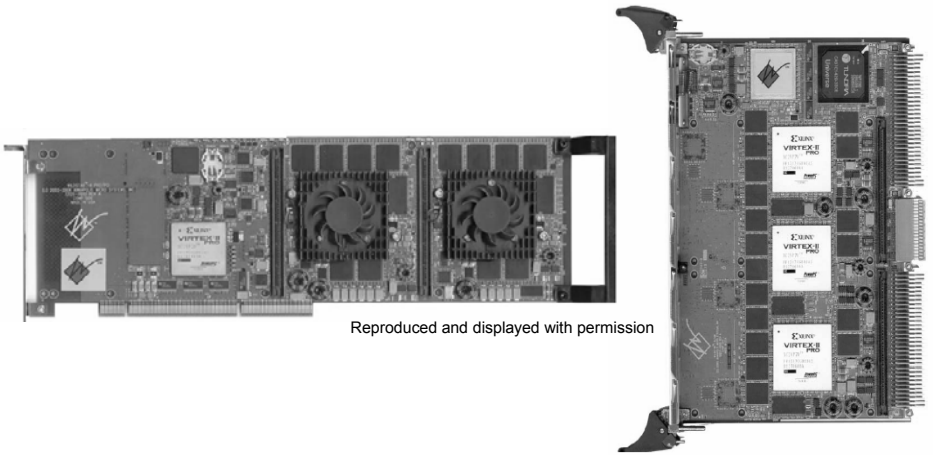
Source: [AMS02]

WILDSTAR™ II for VME



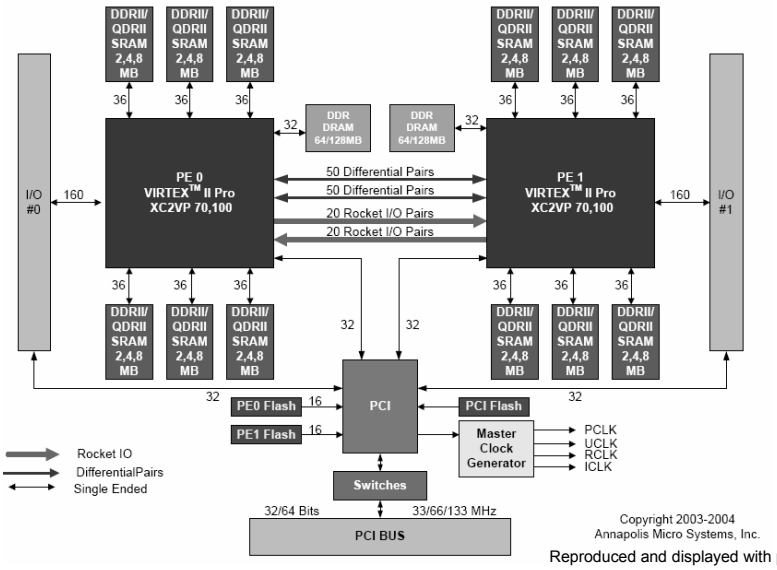
Source: [AMS02]

WILDSTAR™ II Pro



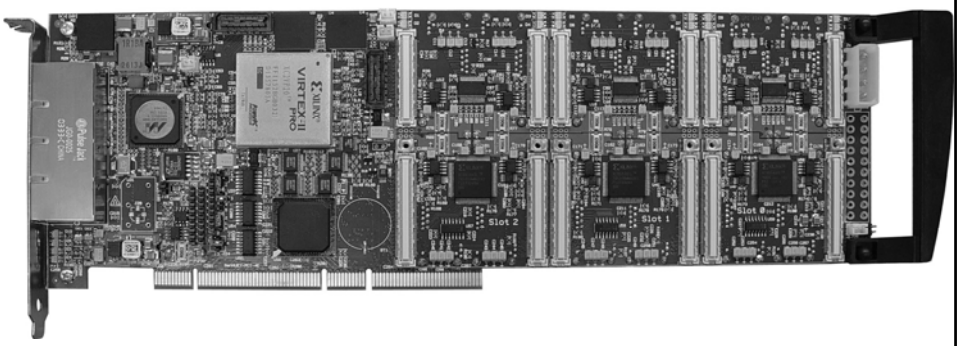
Reproduced and displayed with permission

WILDSTAR™ II Pro



Reproduced and displayed with permission

Nallatech's BenNUEY-PCI-4E



Clusters and Networks of Reconfigurable Computers (NORCs)

Networks of Reconfigurable Computers (NORCs)

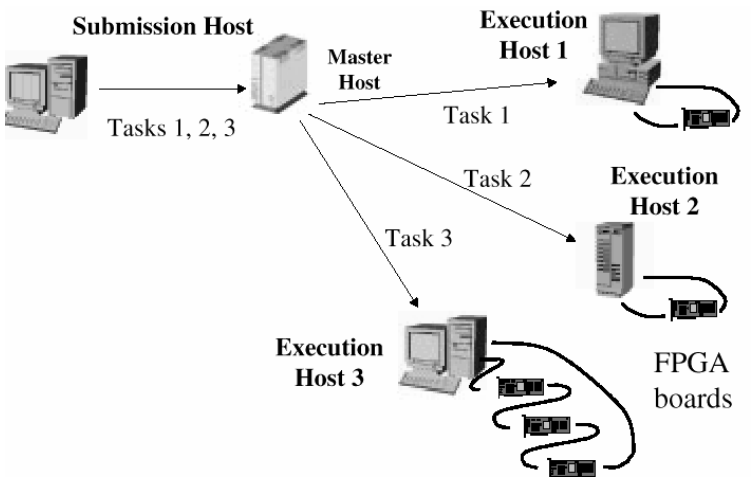
- Expensive reconfigurable workstations can be at times underutilized
- Large problems may need to be spread over a number of workstations
- Many problems may need a high throughput environment
- So, need S/W system to remotely schedule and monitor reconfigurable tasks, send data and bit-streams, and collect results

Example : GWU/GMU Extended JMS

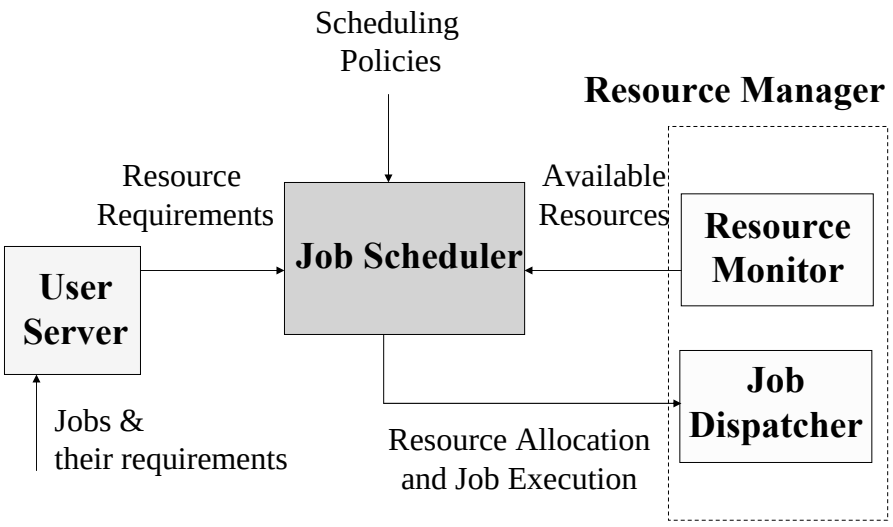
http://www.gwu.edu/~hpc/lsf/
http://ece.gmu.edu/lucite/

- Team from GWU and GMU with DoD support
- Considered extending job management systems (JMS's) to recognize reconfigurable computing resources and support the needed functionalities
- Evaluated many implementations of JMS's and selected LSF (Load Sharing Facility) for implementation

Networked Reconfigurable Resources Management System



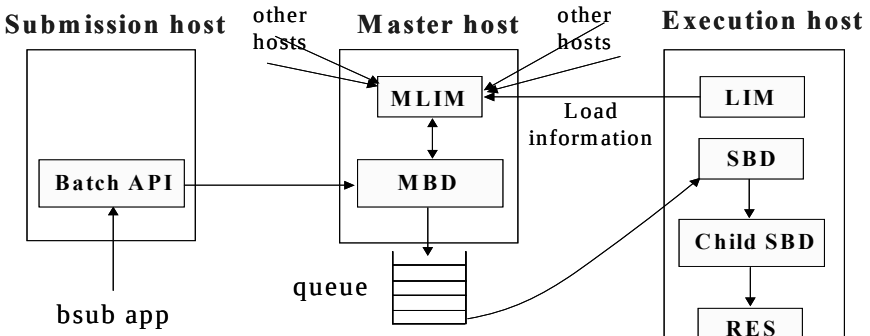
Architecture of a typical Job Management System



LSF

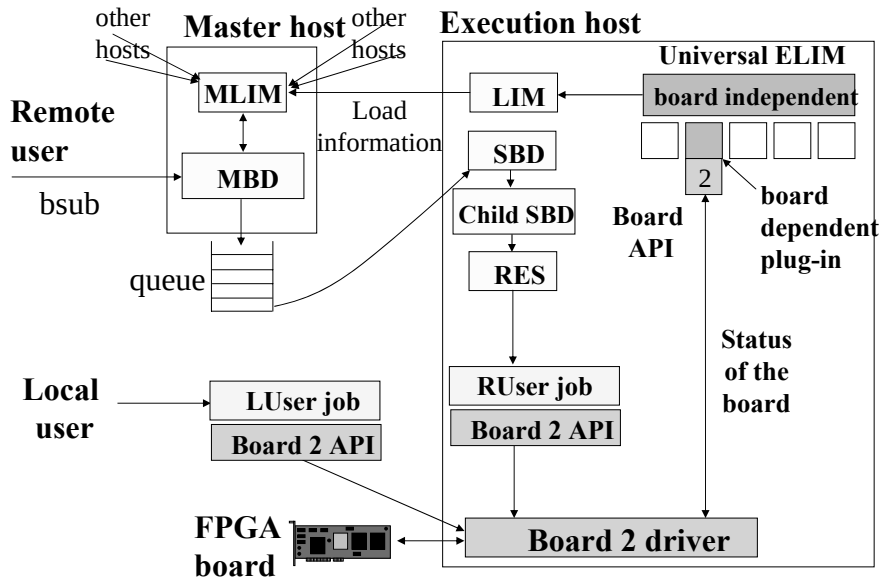
- LSF (Load Sharing Facility) is the product of Platform Computing
- LSF is a layer of software services on top of UNIX and Windows NT operating systems
- The LSF Suite is a set of software modules that manage distributed computing resources and workloads
- LSF creates a single system view on a network of heterogeneous computers so that the whole network of computing resources can be utilized effectively and managed easily

General Architecture of LSF

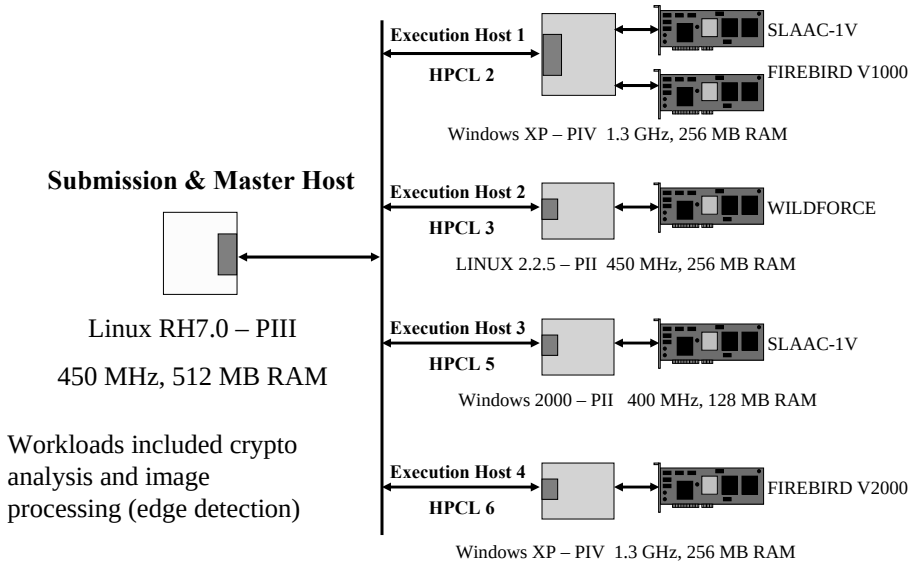


LIM – Load Information Manager
MLIM – Master LIM
MBD – Master Batch Daemon
SBD – Slave Batch Daemon
RES – Remote Execution Server

Extension of LSF to Reconfigurable Hardware

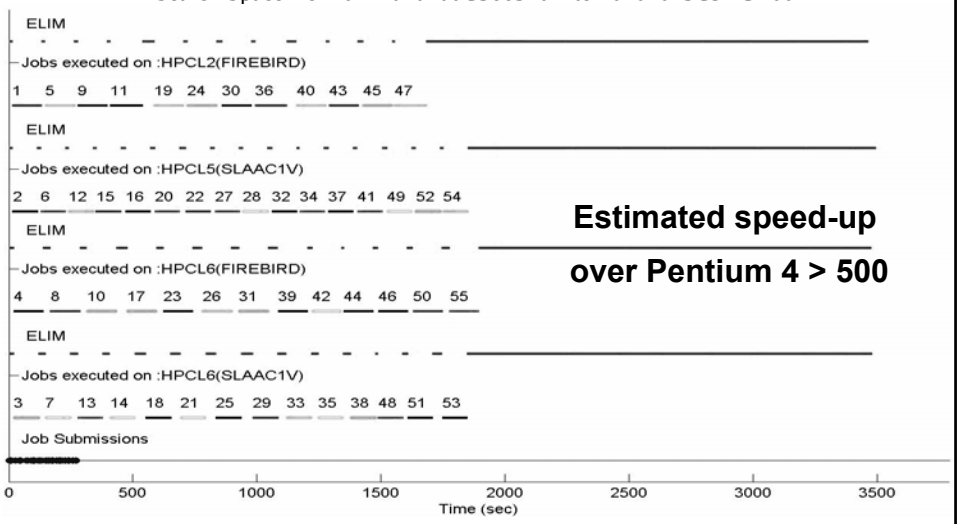


GWU/GMU NORCs Testbed Used in Experiments



Parallel DES Breaker

Ciphertext=0X 8CA64DE9C1B123A7 Plaintext=0X 0000000000000000
Search Space from 0X 1010100C5663702 to 10101013C9BCB00



Estimated speed-up over Pentium 4 > 500

Reconfigurable Computing Clusters

Reconfigurable Beowulf Clusters

- "Do-It-Yourself Supercomputers" - Science 1996
- Built around:
  - Pile of PCs (POP)
  - Dedicated Commodity Network
    - LAN
    - Myrinet, Infiniband, …..
  - Free Unix: Linux
  - Free and COTS Parallel Programming and performance Tools
- COTS Hardware permits rapid development and technology tracking
- COTS reconfigurable boards as accelerators at each node

Example 1: HPTi Solution

Delivered and Benchmarked

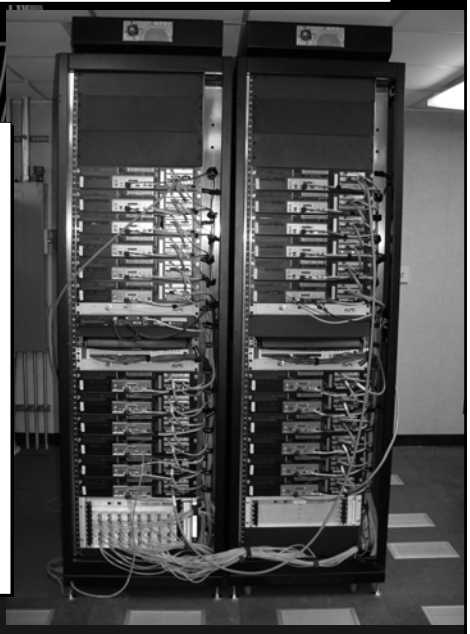
http://www.hpti.com/

Source: [HPTi, MAPLD04]

Delivered and Benchmarked

HPTi

- ☑ 48 nodes
- ☑ 2u, back-to-back (net 1u/node)
- ☑ 96 FPGA's
- ☑ Annapolis Micro
- ☑ Xilinx Virtex II
- ☑ 34 Tera-Ops
- ☑ *In use today*
- ☑ *All Commodity Parts*



Tower of Power

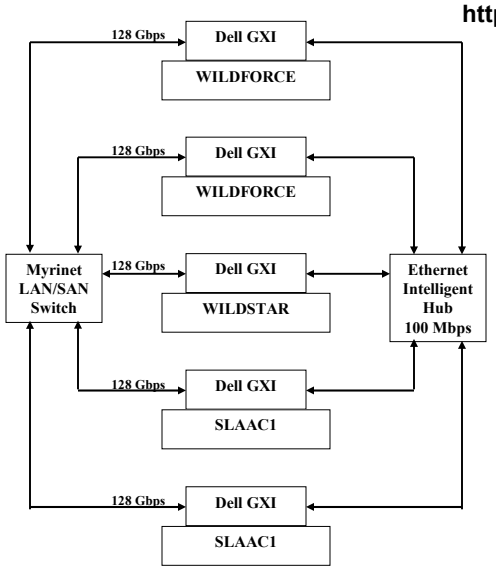
http://ccm.ece.vt.edu/

- **16-node cluster of PCs**
- **WILDFORCE board on each PC**
- **Myrinet network connecting all PCs**
- **Runs ACS API (platform independent API for the configuration and control of multi-board systems)**

Source: [ACS01]

SLAAC RRP (Research Reference Platform)

http://www.east.isi.edu/



Source: [ISI 01]

Scalable Reconfigurable Systems

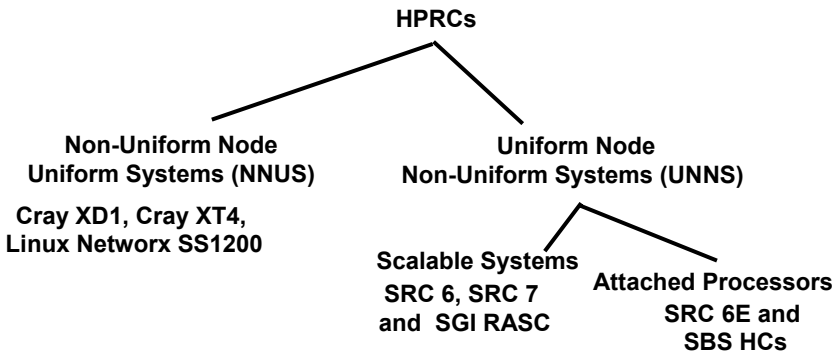
Reconfigurable Supercomputers at GWU

Scalable Reconfigurable Systems

- Large numbers of reconfigurable processors and microprocessors
- Everything can be configured
  - Functional units
  - Interconnects
  - Interfaces
- High-level of scalability
- Suitable for a wide range of applications
- Everything can be reconfigured over and over at run time (Run-Time Reconfiguration) to suite underlying applications
- Can be easily programmed by application scientists, at least in the same way of programming conventional parallel computers

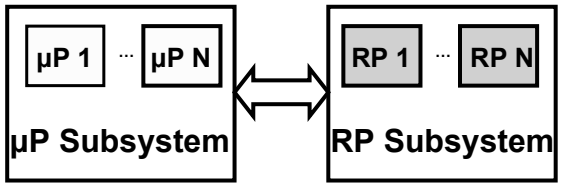
An Architectural Classification for High-Performance Reconfigurable Computers

A Classification for High-Performance Reconfigurable Computers (HPRCs)



Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, Kris Gaj, Volodymyr Kindratenko, Duncan Buell, "The Promise of High-Performance Reconfigurable Computing", IEEE Computer (In Press).

1. Uniform Node Non-Uniform Systems (UNNS)

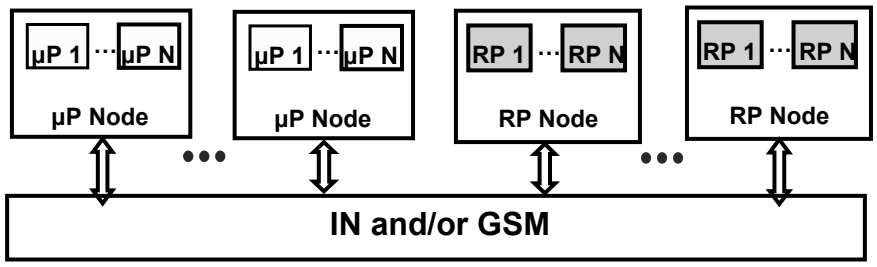


a. Non-Scalable (or Attached Processor) Architecture

Examples: SRC 6E and SBS HC

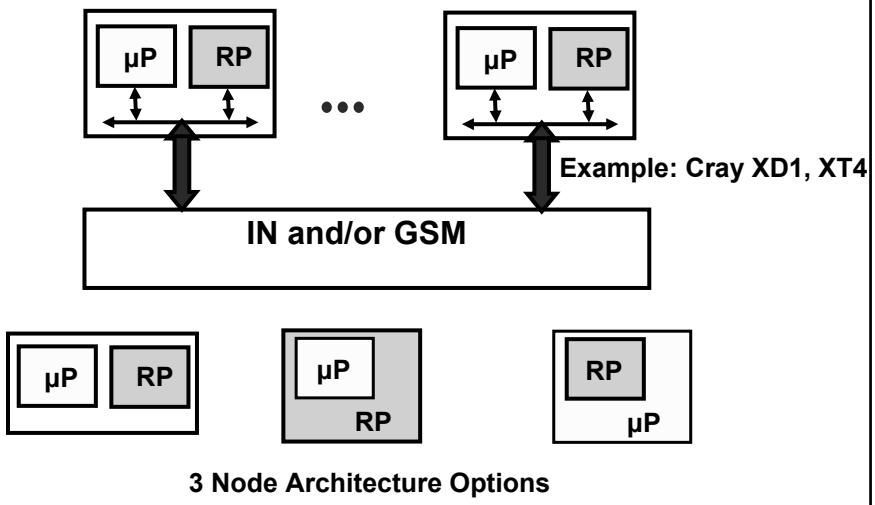
1. Uniform Node Non-Uniform Systems (UNNS)

b. Scalable System



Examples: SRC 6, SGI Altix/RASC

2. Non-Uniform Node Uniform Systems (NNUS)



3 Node Architecture Options

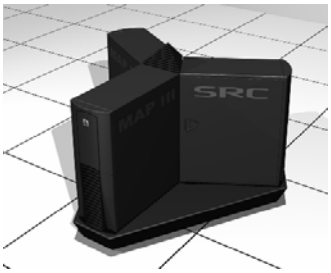
Example1: SRC Systems

http://www.srccomp.com/

Source: [SRC, MAPLD04]

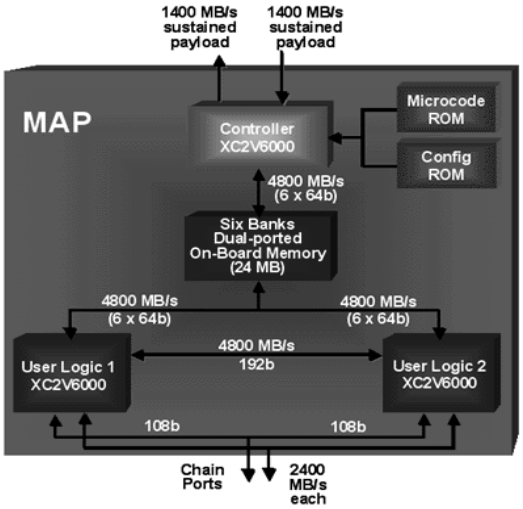
Example1: SRC 6 System

http://www.srccomp.com/



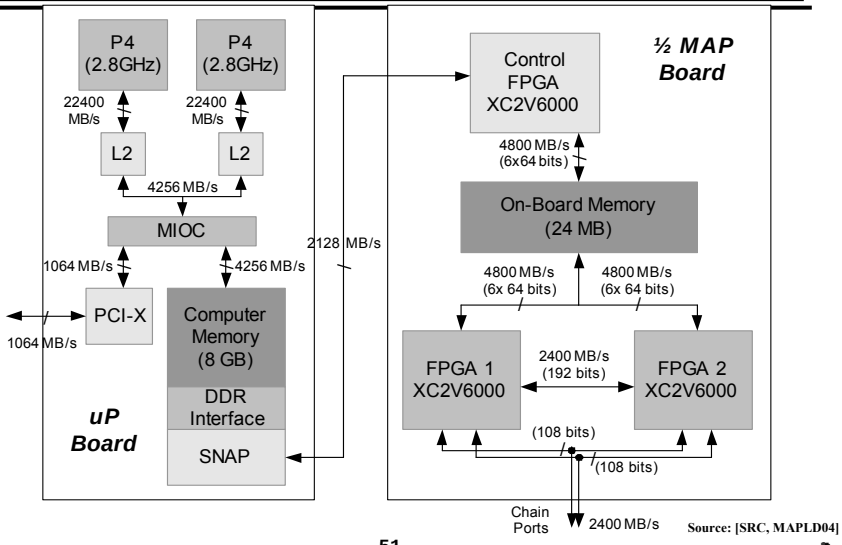
Source: [SRC, MAPLD04]

SRC MAP™ Reconfigurable Processor



Source: [SRC, MAPLD04]

SRC Hardware Architecture



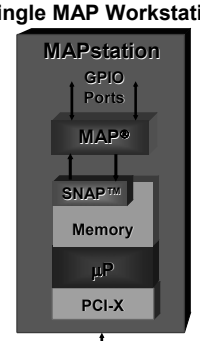
Source: [SRC, MAPLD04]

51

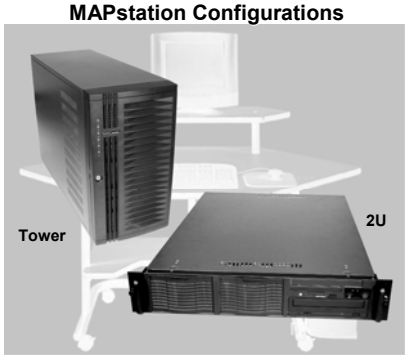
SRC MAPstation™

SRC-6 uses standard external network connections

Single MAP Workstation



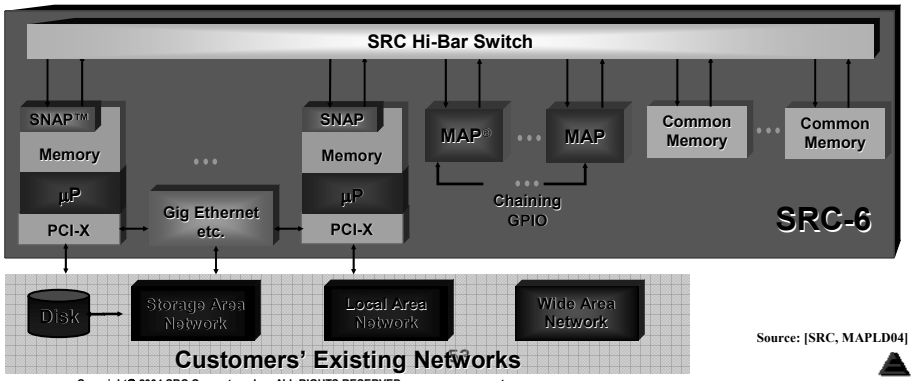
MAPstation Configurations



Source: [SRC, MAPLD04]

SRC Hi-Bar™ Based Systems

- Hi-Bar sustains 1.4 GB/s per port
- Up to 256 input and 256 output ports
- Common Memory (CM) has controller with DMA capability
- Up to 8 GB DDR SDRAM supported per CM node

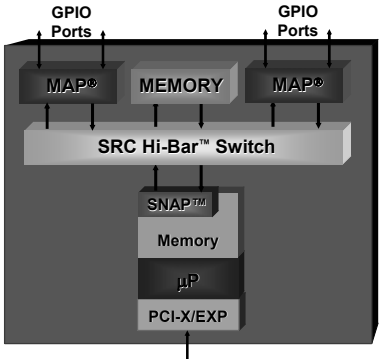


Source: [SRC, MAPLD04]

SRC MAPstation™ with Hi-Bar™

MAPstation towers hold up to 3 MAP or memory nodes

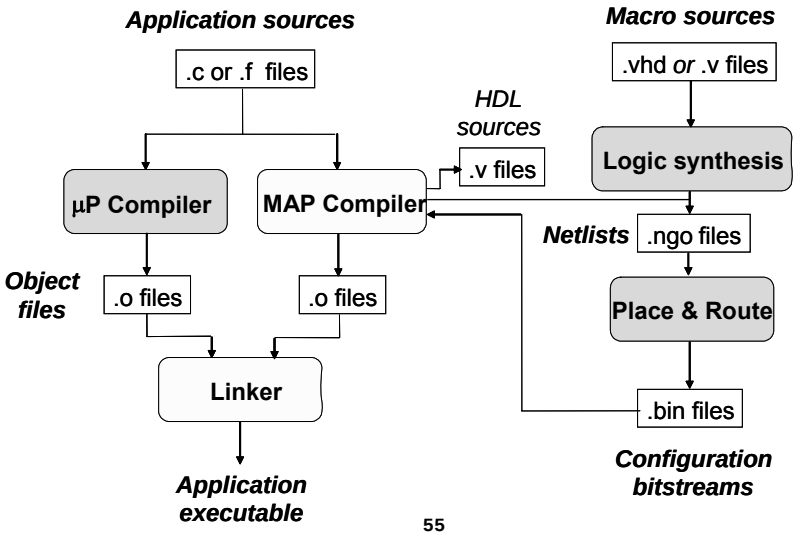
MAPstation with 2 MAPs and Common Memory



MAPstation Tower

Source: [SRC, MAPLD04]

SRC Compilation Process

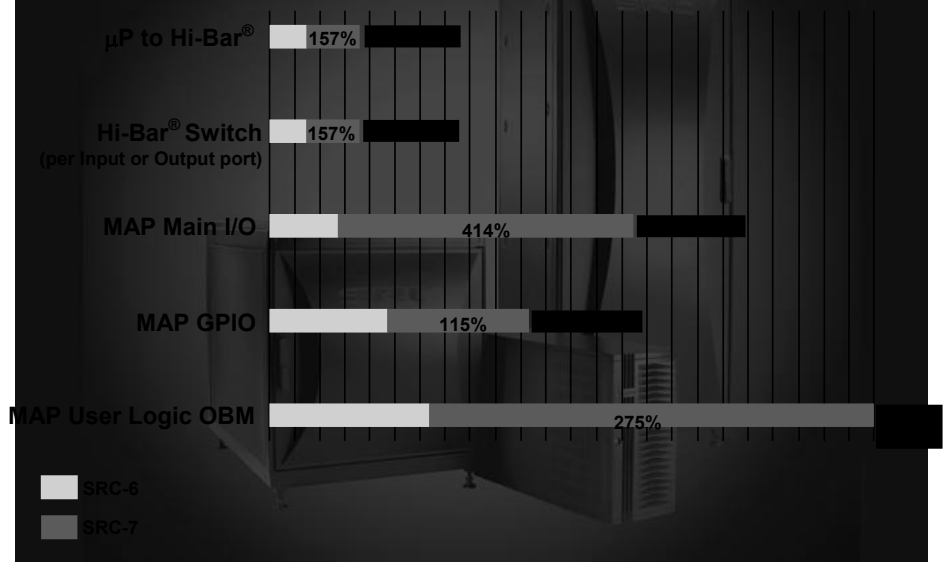


55

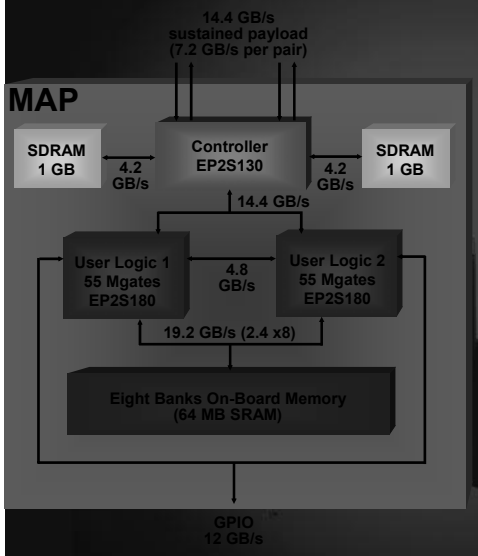
SRC-7

- **Focus on higher bandwidth**
  - **Faster interconnect**
  - **More memory accesses**
- **Maintains software compatibility**

Sustained Payload Bandwidths



Series H MAP®

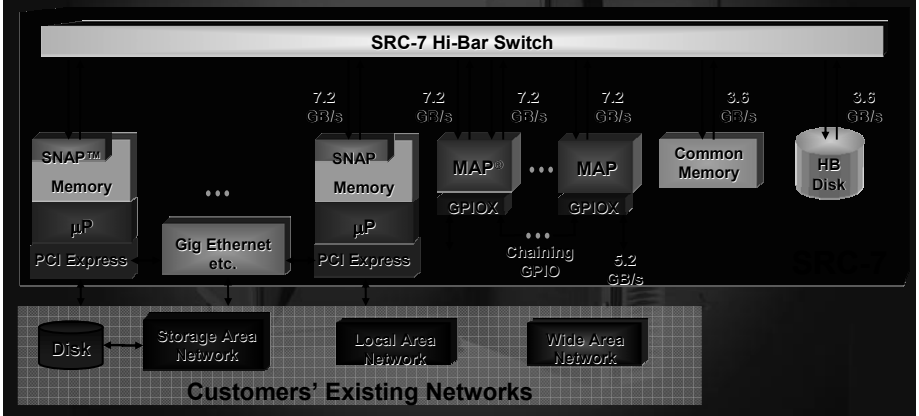


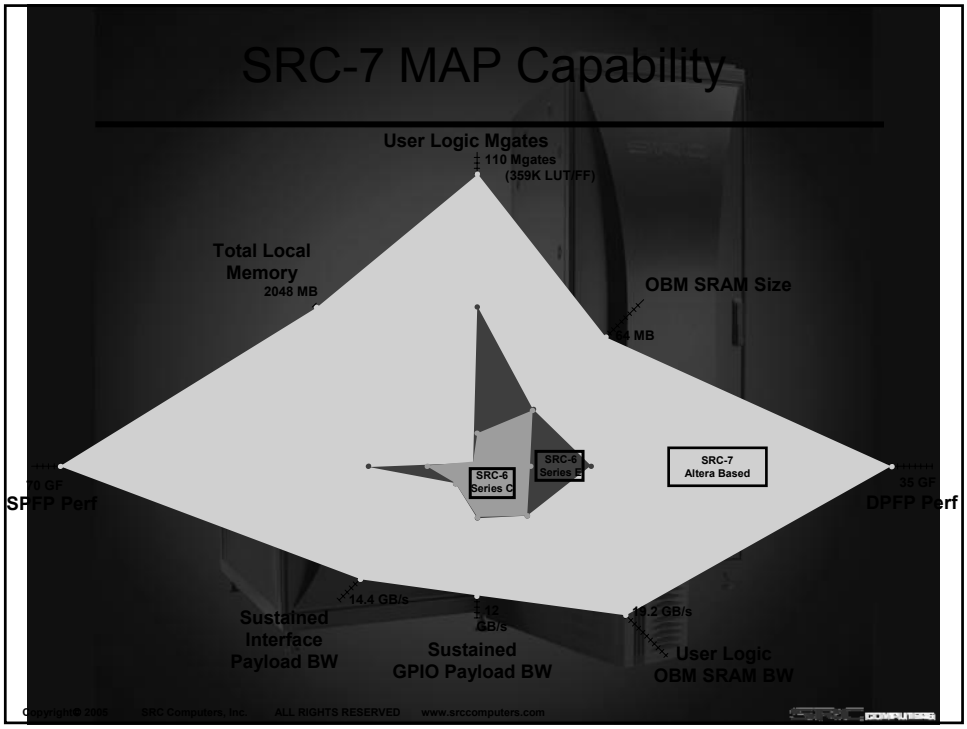
- 1 or 2 LVDS main I/O ports
- 150 MHz nominal User Logic speed
- 16 simultaneous SRAM OBM references
- 2 dedicated 64 bit Bridge Ports
- 2 simultaneously accessible DDR2 SDRAM OBCM banks
  - Initial release is 512 MB, 1 GB to follow
- Streaming supported between I/O, OBCM, User Logic, OBM and GPIOX
- Simultaneous input and output DMAs
- GPIO eXpansion (GPIOX) cards

Series H MAP
5.25" Drive Bay Enclosure

SRC-7 Hi-Bar™ Based Systems

- Hi-Bar sustains 3.6 GB/s payload per path with 180 ns latency per tier
- 2 tiers support 256 nodes
- MAP can use 1 or 2 Hi-Bar ports
- GPIO can be chained or used for direct data input to MAP



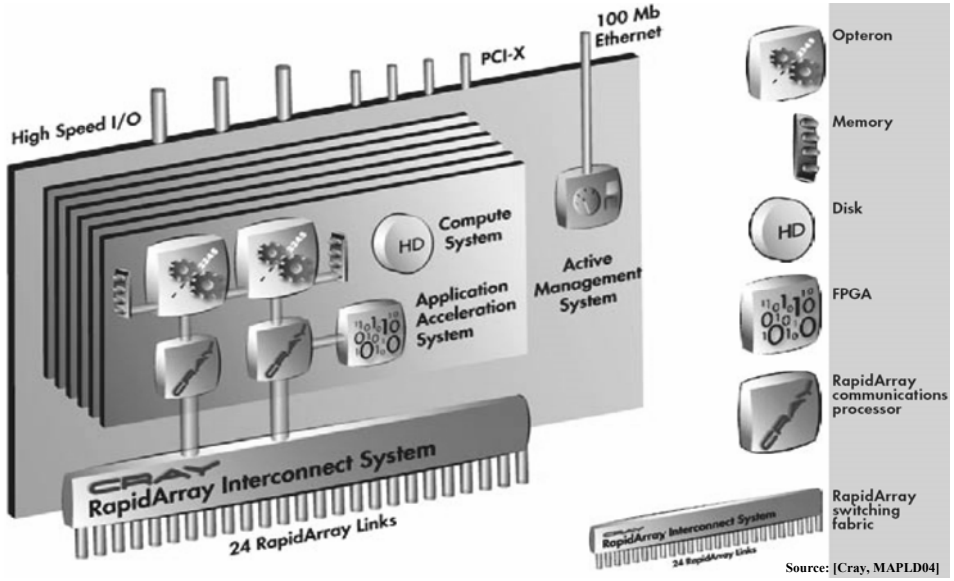


Example 2: Cray XD1 (OctigaBay 12K)

http://www.cray.com

Source: [Cray, MAPLD04]

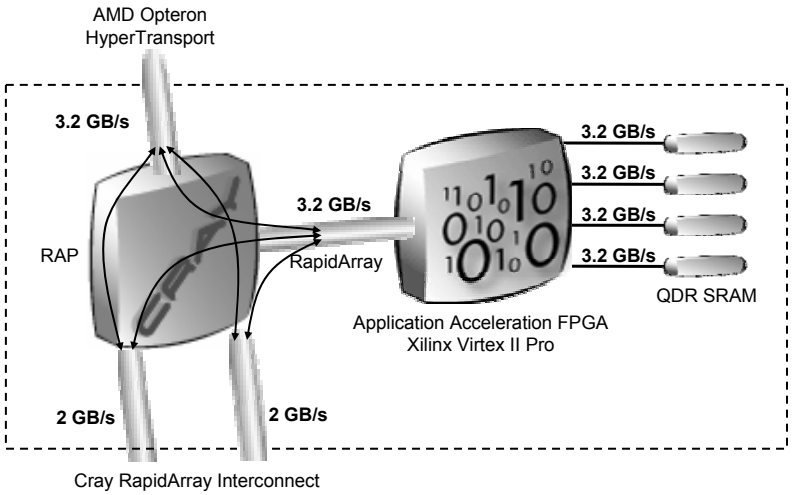
Cray XD1 (OctigaBay 12K)



Source: [Cray, MAPLD04]

63

Application Acceleration Co-Processor



Source: [Cray, MAPLD04]

64

Cray XD1 Solutions

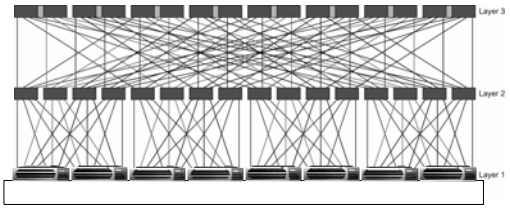
OctigaBay 12K	Rack	Supercomputer
1 Shelf	12 Shelves	1,000 Shelves
12 Processors	144 Processors	12,000 Processors
58 GFLOPS	**691 GFLOPS**	**58 TFLOPS**
Up to 96 GB Memory	Up to 1.2 TB memory	Up to 96 TB
8 GB/s, 1 microsecond latency	8 GB/s, 1 microsecond latency	8 GB/s, 1 microsecond latency
1 Tb/s switch fabric	12 Tb/s switch fabric	1 Pb/s switch fabric
Active management	Active management	Active management

Modular computing power with high availability

Source: [Cray, MAPLD04]

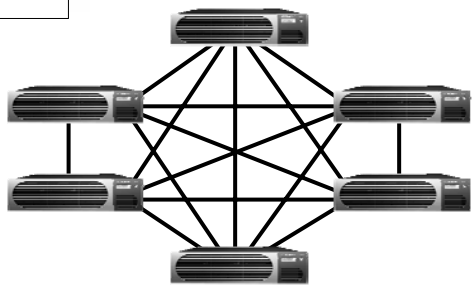
65

Cray XD1 System



Fat Tree

Direct Connect

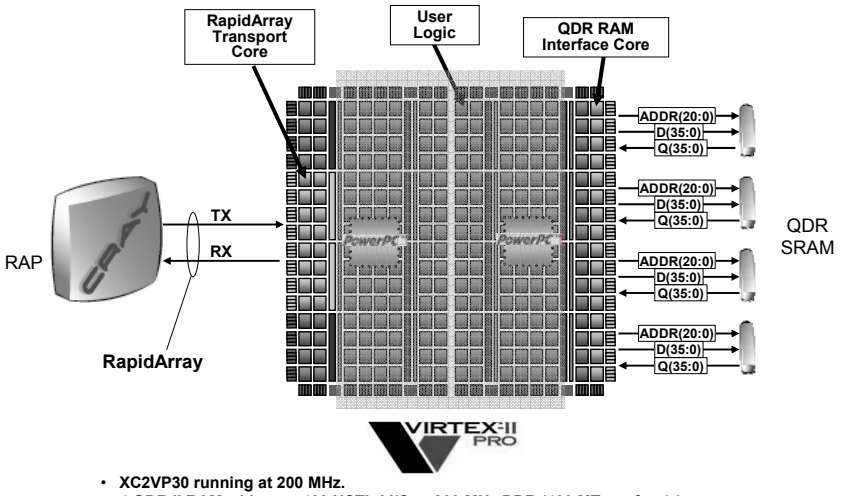


Multiple Chassis Connected to RapidArray Fabric

Source: [Cray, MAPLD04]

66

Application Acceleration Interface



- XC2VP30 running at 200 MHz.
- 4 QDR II RAM with over 400 HSTL-I I/O at 200 MHz DDR (400 MTransfers/s).
- 16 bit RapidArray I/F at 400 MHz DDR (800 MTransfers/s.)
- QDR and RapidArray I/F take up <20 % of XC2VP30. The rest is available for user applications.

Source: [Cray, MAPLD04]

67

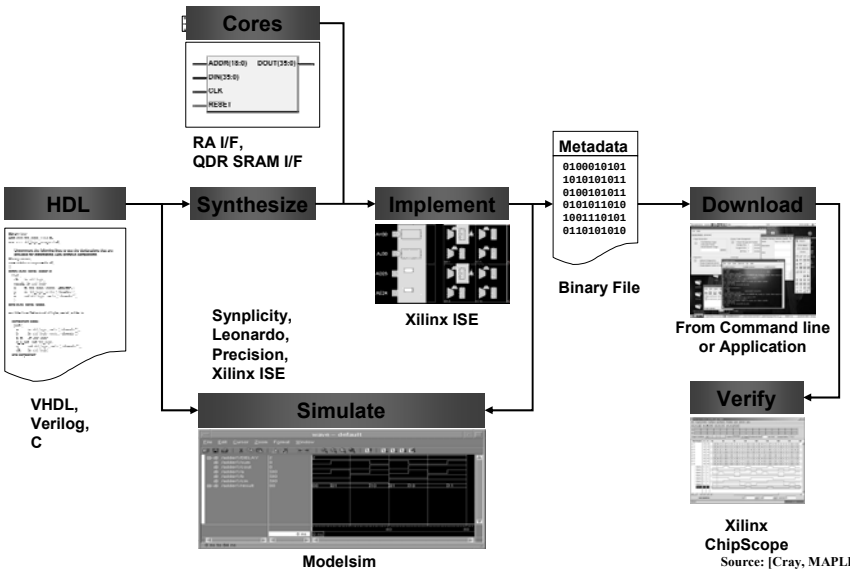
FPGA Linux API

- Admininstration Commands
  - fpga_open - allocates and opens fpga
  - fpga_close - closes allocated fpga
  - fpga_load - loads binary into fpga
  - fpga_is_loaded - queries the programming state of the FPGA
  - fpga_uload - clears the configuration in FPGA (hard-reset)
- Operation/Control Commands
  - fpga_start - start fpga (release from reset)
  - fpga_reset - soft-resets the FPGA
- Mapping Commands
  - fpga_set_ftrmem - maps application virtual address to allow access by FPGA
  - fpga_memmap - maps FPGA ram into application virtual space
  - fpga_mem_sync - forces completion of outstanding transactions to mapped FPGA memory
- Data Commands
  - fpga_wrt_appif_val - writes data into application interface (register space)
  - fpga_rd_appif_val - reads data from application interface (register space)
- Status Commands
  - fpga_status - gets status of fpga

Source: [Cray, MAPLD04]

68

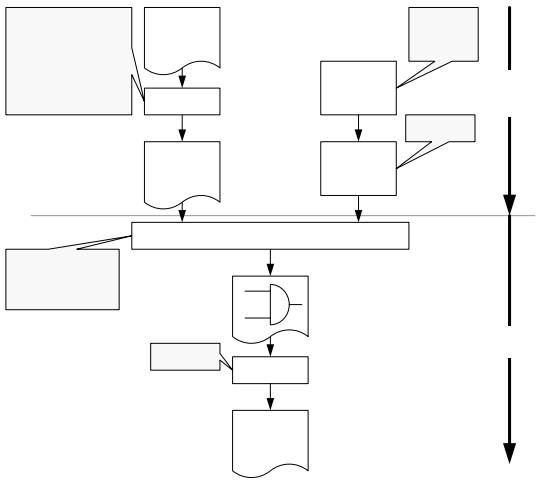
Standard FPGA Development Tools



Source: [Cray, MAPLD04]

69

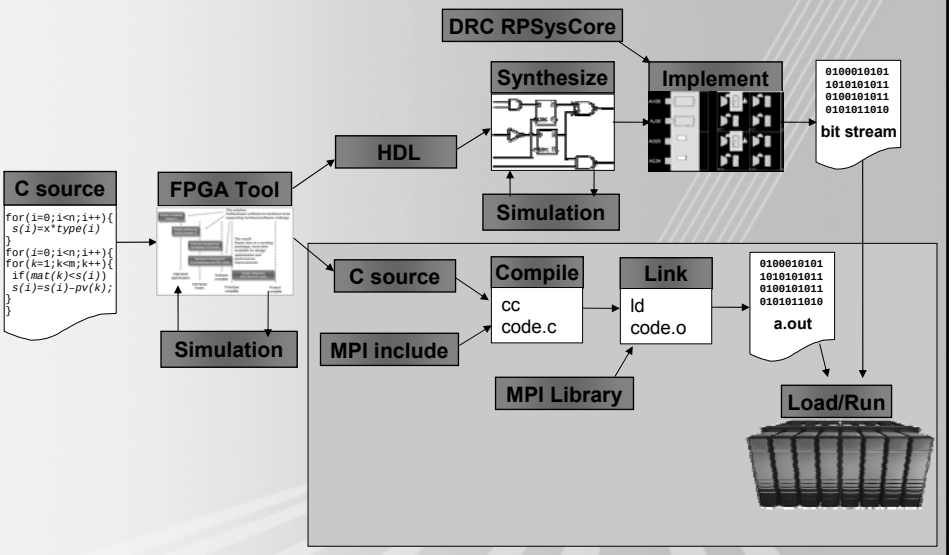
Additional High Level Tools



Source: [Cray, MAPLD04]

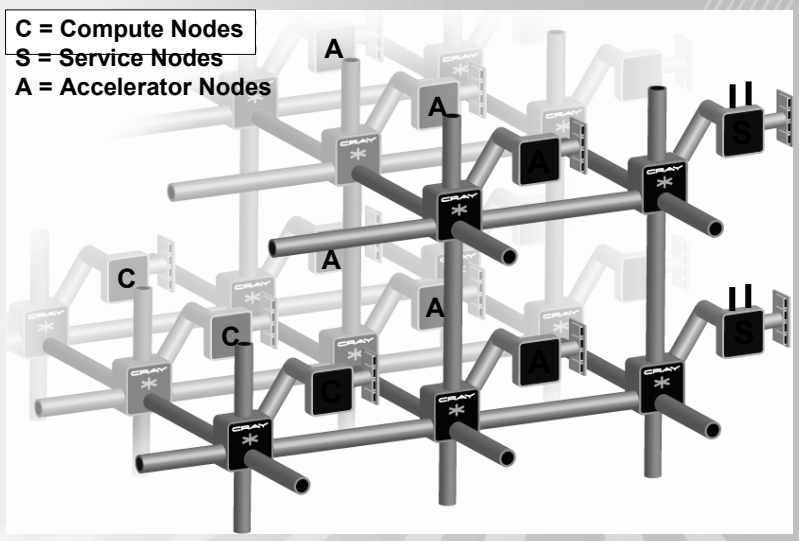
70

Software Development Flow



Source: [Cray, SC07]

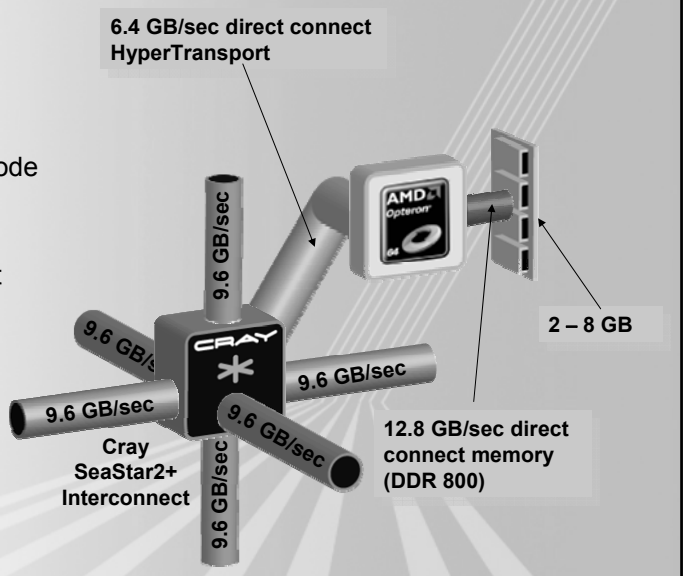
XT4 Scalable Interconnect



Source: [Cray, SC07]

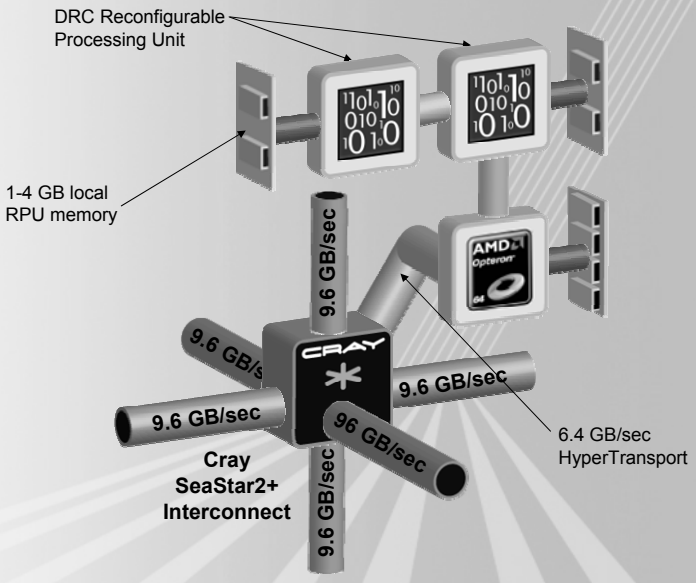
Cray XT4 Node

- 4-way SMP
- >35 Gflops per node
- Up to 8 GB per node
- OpenMP Support within socket



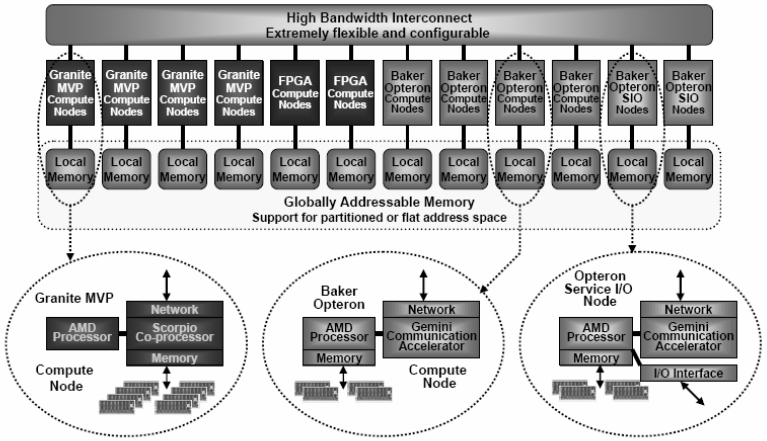
Source: [Cray, SC07]

Cray XR1 Reconfigurable Blade



Source: [Cray, SC07]

Cascade System Architecture



- Globally addressable memory with unified addressing architecture
- Configurable network, memory, processing and I/O
- Heterogeneous processing across node types, and within MVP nodes
- Can adapt at configuration time, compile time, run time

Example 3: SGI Altix

http://www.sgi.com/servers/altix/

Source: [SGI, MAPLD04]

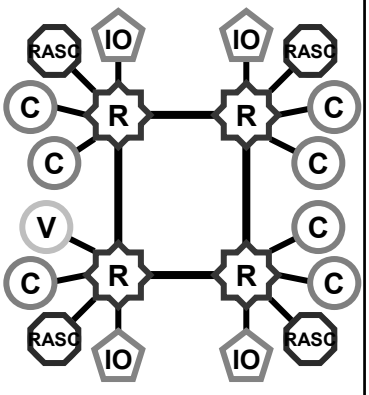
SGI Systems

http://www.sgi.com

sgi

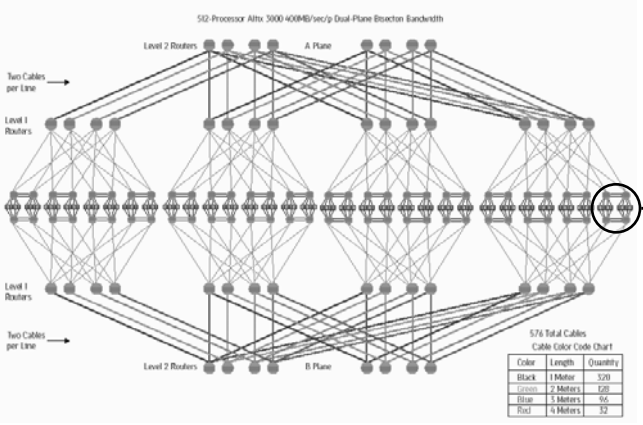
System Architecture

- R • NUMAlink system interconnect
- C • General-purpose compute nodes
- IO • Peer-attached general purpose I/O
- V • Integrated graphics/visualization
- RASC • Reconfigurable Application Specific Computing

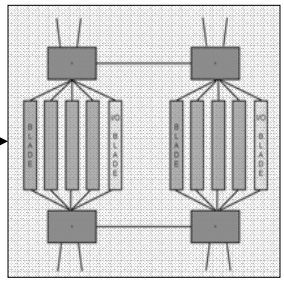


sgi

NUMALink Topology



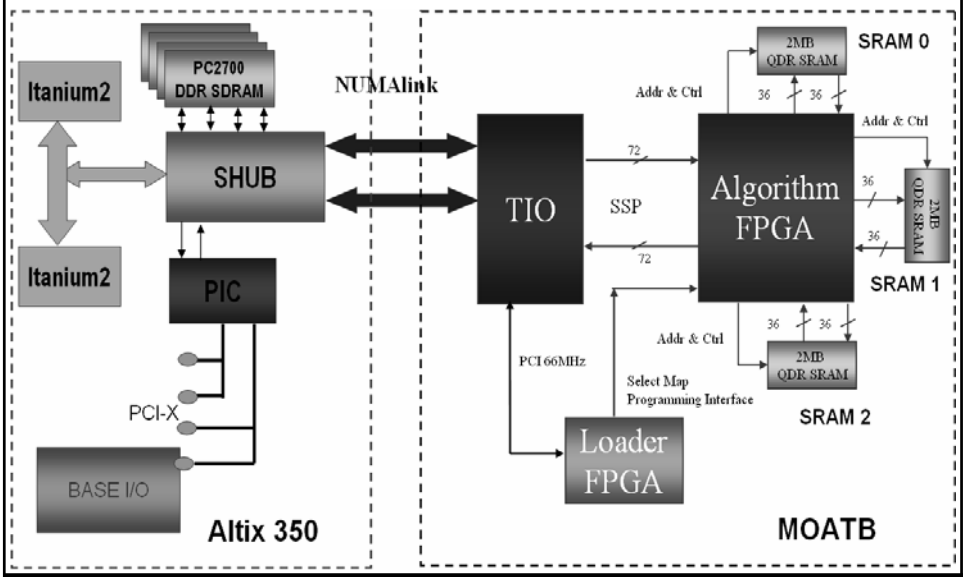
NUMALink topology of a 512-processor dual "Fat-Tree"



NUMAlink interconnect at the lowest hierarchy level

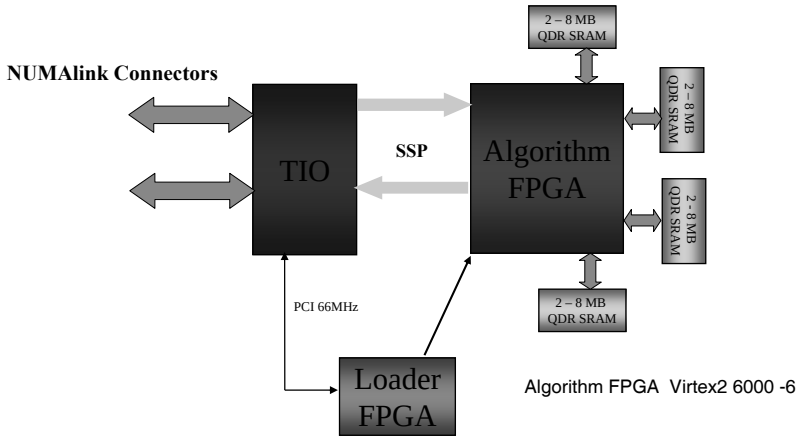
sgi

RASC Architecture



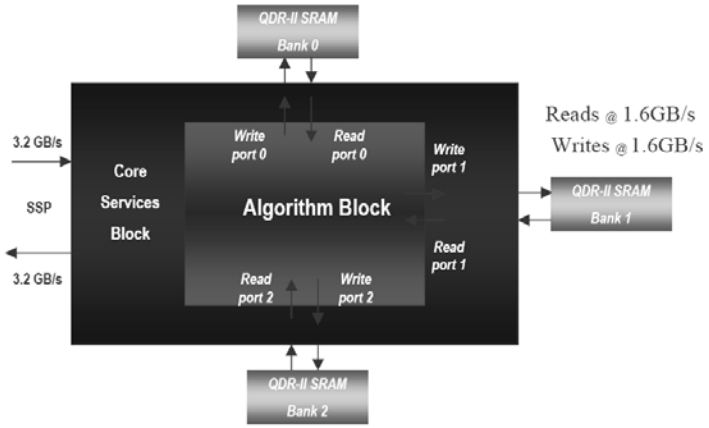
sgi

Current Product – SGI® RASC™ Technology (Athena)



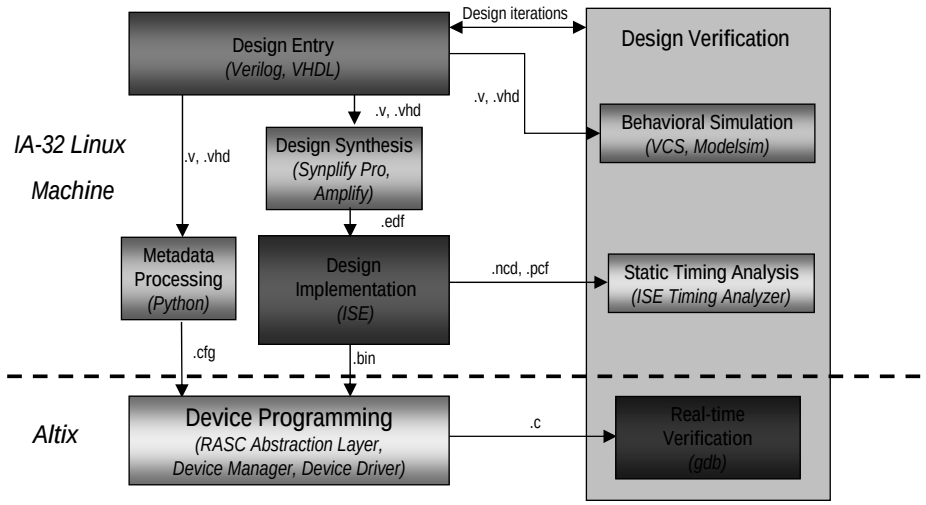
sgi

FPGA Architecture Overview



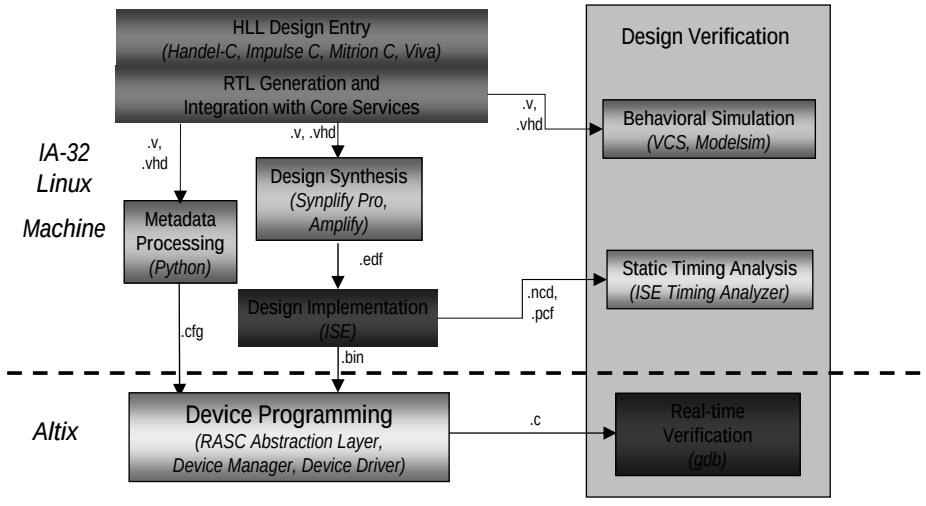
sgi

Design Flow
(HDLs)



sgi

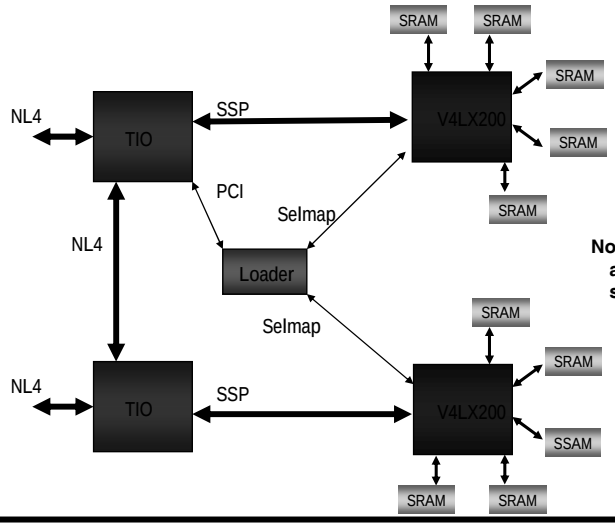
Design Flow
(HLLs)



sgi

SGI® RASC™ RC100 Blade

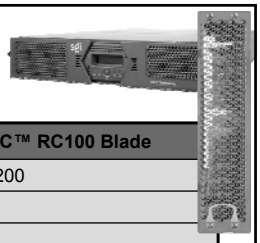
Computation Blade



Note: both QDR SRAM and DDR2 SDRAM supported

sgi

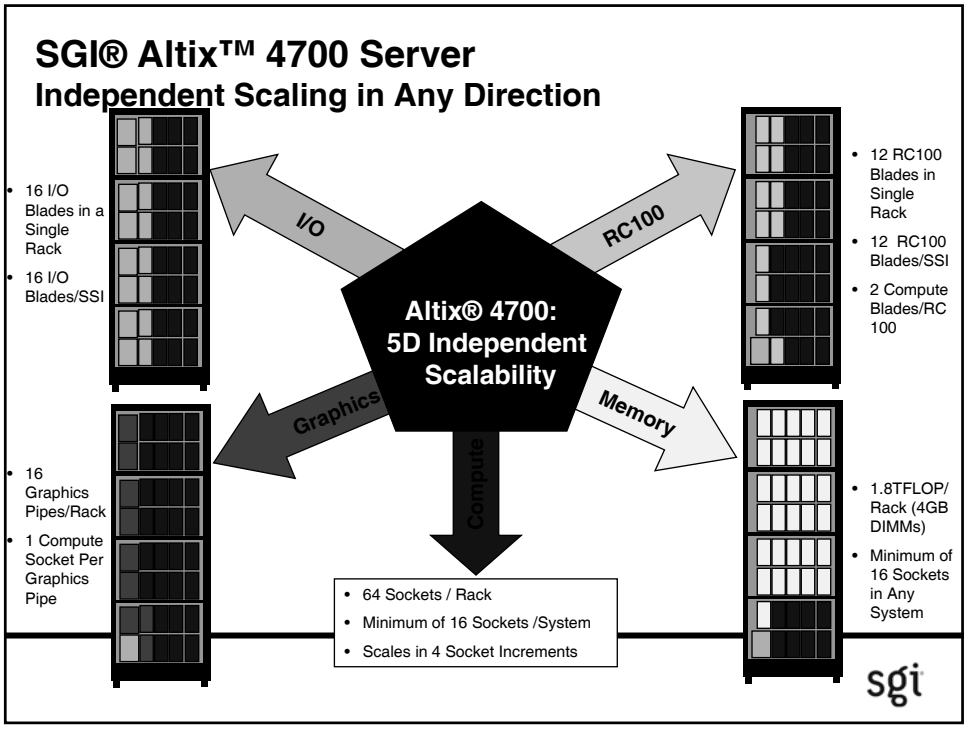
SGI® RASC™ Specifications



	SGI® RASC™ Module (Ver. 1)	SGI® RASC™ RC100 Blade
FPGA	Xilinx Virtex II-6000	Xilinx Virtex-4 LX200
No. of FPGAs	One per brick	Two per blade
Host System	SGI® Altix® 3700 Bx2 or 350 Silicon Graphics Prism™	SGI® Altix® 4000 SGI® Altix® 3700 Bx2 or 350 * Silicon Graphics Prism™*+
Memory	16MB QDR SRAM	80MB QDR SRAM **OR** 20GB DDR2 SDRAM
I/O	Dual NUMAlink™ 4 ports	Dual NUMAlink™ 4 ports
Max Config	Up to 2 units per system	Up to 8 RC100 blades per system More available with custom configuration
Dimensions	Rack-Mountable Form Factor ▪EIA slide-mountable ▪2U (3.5" H x 19"W x 26"D)	Blade Form Factor ▪10-U Altix® 4000 IRU ▪Up to 8 RC100 blades per IRU Rack-Mountable Form Factor ▪2 blade slot chassis ▪3U (5.25" H x 19"W x 26"D)
O/S	Linux® OS (on host server)	Linux® OS (on host server)

* with available 2 blade slot upgrade chassis
+ rack mounted version only

sgi



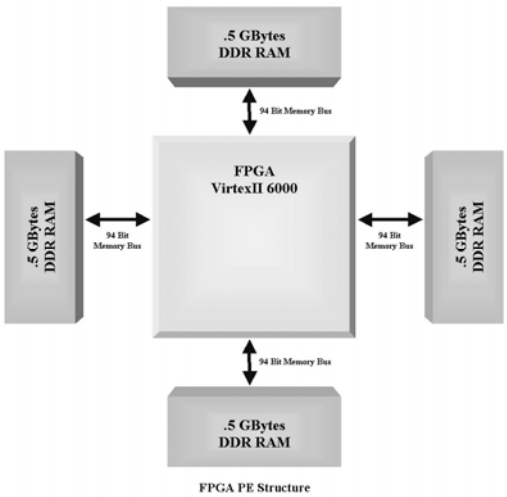
Example 4: The Starbridge Hybrid Computer 62m http://www.starbridgesystems.com/

Source: [SGI, MAPLD04]

Hypercomputers: High-Performance FPGA Accelerators

Source: [SBS, MAPLD04]

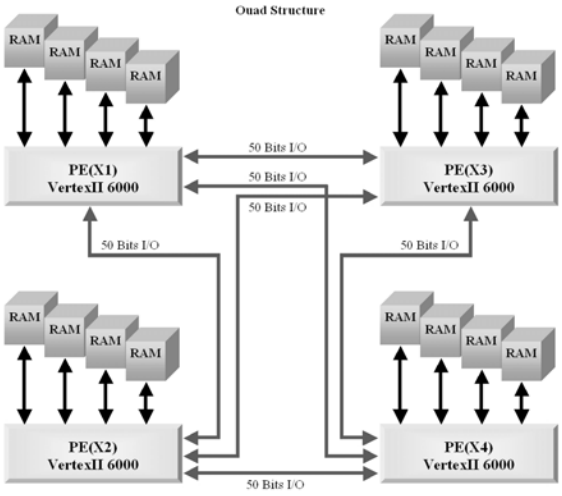
Structure of an FPGA Processing Element



FPGA PE Structure

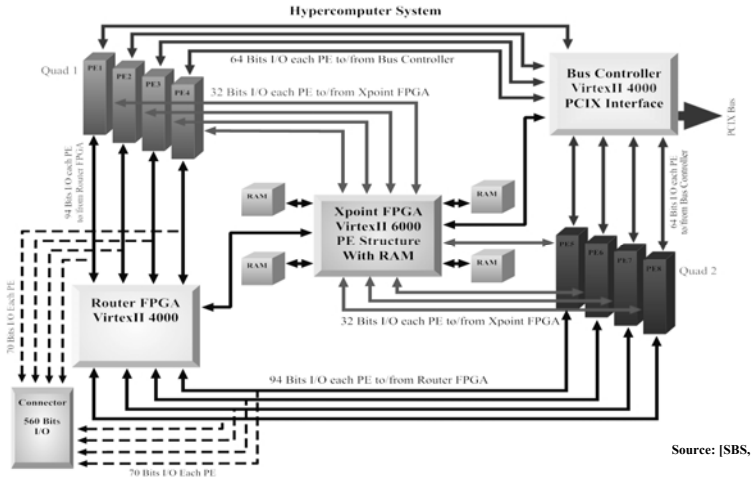
Source: [SBS, MAPLD04]

Structure of a Processing Element Quad



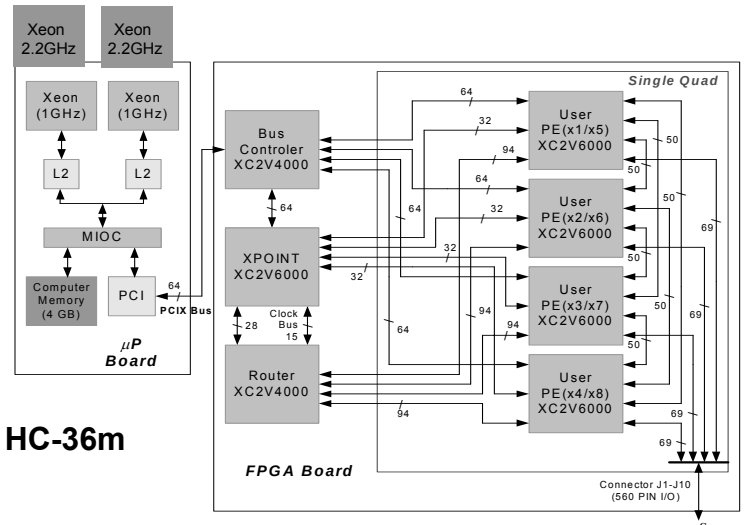
Source: [SBS, MAPLD04]

Hypercomputer Architecture



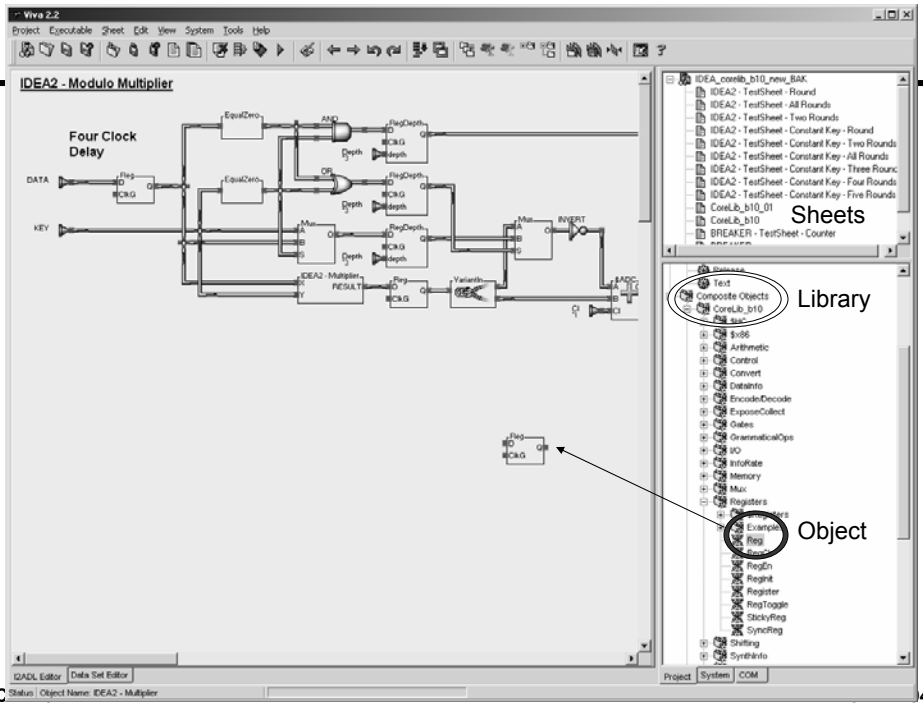
Source: [SBS, MAPLD04]

Star Bridge Hardware Architecture

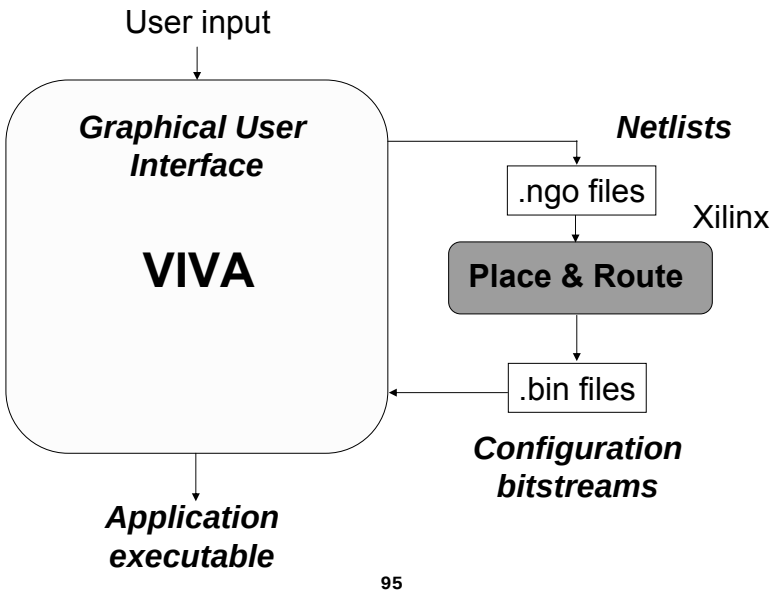


HC-36m

Source: [SBS, MAPLD04]



Star Bridge Software Environment



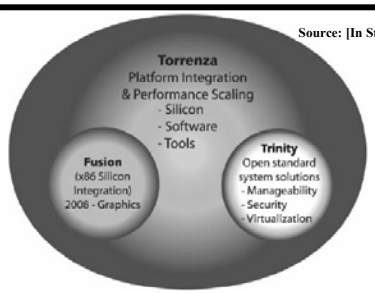
Emerging Directions

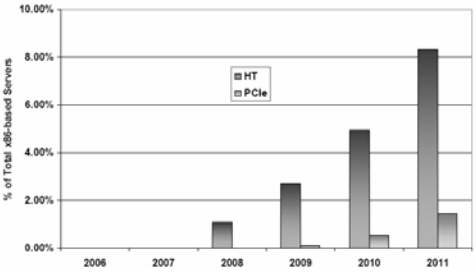
AMD Torrenza

(http://enterprise.amd.com/us-en/AMD-Business/Technology-Home/Torrenza.aspx)

- **Fit into existing AMD Opteron Socket**
- **Leverages HyperTransport Link**

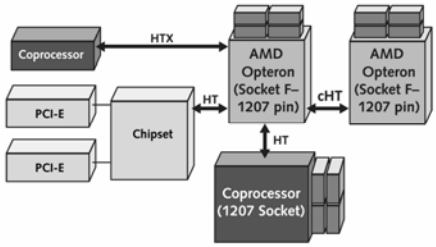
Source: [In Stat, 5/07]





Source: [In Stat, 5/07]

Projected Growth of HT and PCIe Coprocessing in x86-based Servers (Excluding GPUs)



Source: [AMD, 5/07]

AMD Torrenza

(http://enterprise.amd.com/us-en/AMD-Business/Technology-Home/Torrenza.aspx)

	X86-only	Proprietary	x86 Custom/ Proprietary	Torrenza
TCO	Low	High	Medium-High	Low-Medium
Flexibility	Low	High	Medium	High
Scalability	High	Low	Medium	High
Manageability	High	Medium	Medium	High
Performance	Low-Medium	High	Medium-High	Medium-High

Source: [In Stat, 5/07]

Public Torrenza Participants

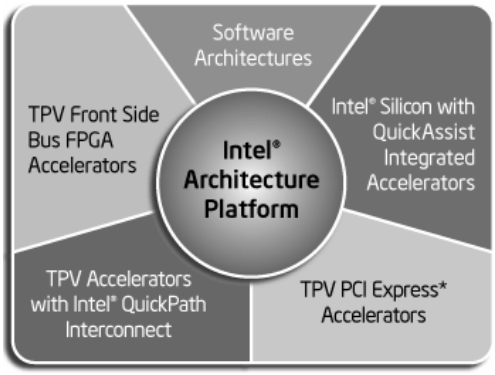
Company	Market Segment	Coherent License	Product(s) in Development
3Leaf Systems	Systems	Yes	Virtual I/O server
ACTIV Financial	Software	No	Market data applications
AMI	Software	No	BIOS & software development tools
Altera	Silicon	No	FPGAs
Bay Microsystems	Silicon	No	Network processors
Cadence	Design	No	IP for HT interface & design tools for 90nm, 65nm, and 45nm
Celoxica	Software	No	Software compiler, RTS, & FPGA programming tools
Commex Technologies	Silicon	No	Core-logic chipsets
Cray	Systems	Yes	Coprocessors & HPC systems
DRC Computer	Silicon	No	Coprocessors
Flextronics	Design/Manufacturing	No	Design & manufacturing services
HP	Systems	No	Servers with HTX slots
IBM	Systems	No	Systems
Lattice Semiconductor	Silicon	No	FPGAs
Liquid Computing	Systems	No	Scalable systems
Microway	Systems	No	Systems using DRC FPGAs
NetLogic	Silicon	No	NET7 content accelerator
Newisys	Silicon	No	Coherent HT fabric
Panta Systems	Systems	No	Scalable systems
Phoenix Technologies	Software	No	BIOS & software development tools
Qlogic	Silicon	No	Inifiniband I/O
RapidMind	Software	No	Development suite
Raza Microelectronics	Silicon	Yes	MIPS-based processors
SRC Computers	Silicon	No	FPGAs
Sun Microsystems	Systems	No	Scalable systems
Tarari	Silicon	No	Content inspection & media processors
U. Mannheim	Silicon	Yes	HTX reference designs, HT & cHT open source cores
Xilinx	Silicon	No	FPGAs
XtremeData	Silicon	No	FPGAs

Source: [AMD, 5/07]

Intel® QuickAssist Technology

(http://www.intel.com/technology/platforms/quickassist/index.htm)

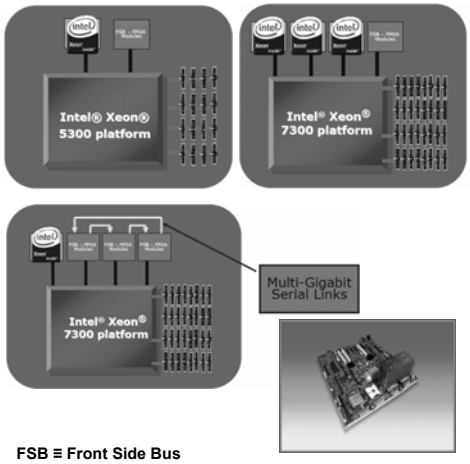
- **A comprehensive initiative**
  - **A family of interrelated Intel and industry standard technologies**
- **Enables optimized use and deployment of accelerators on Intel® platforms**
  - **Accelerated performance for demanding applications with Front Side Bus (FSB) attached Field Programmable Gate Arrays (FSB-FPGA) hardware modules**
  - **Fits into existing Xeon Socket**
  - **Leverages FSB link**

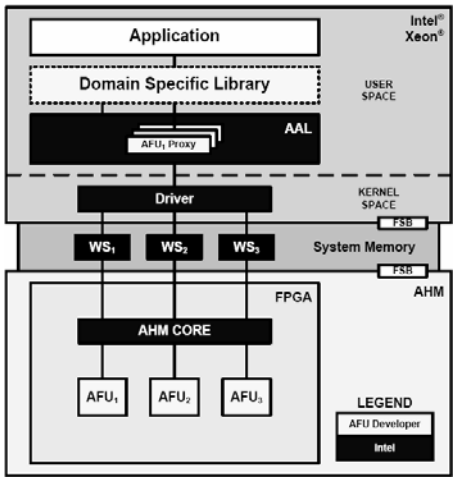


FSB ≡ Front Side Bus
TPV ≡ Third-Party Vendor

Intel® QuickAssist Technology

(http://www.intel.com/technology/platforms/quickassist/index.htm)

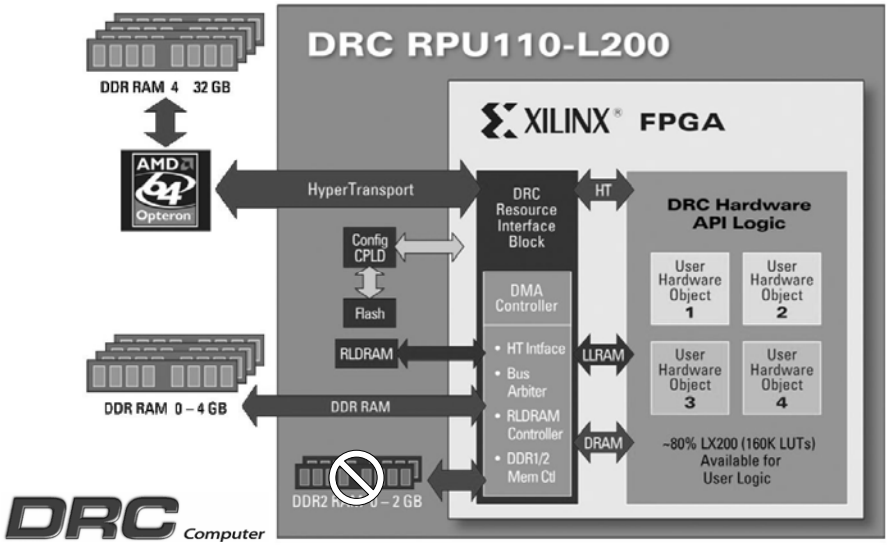




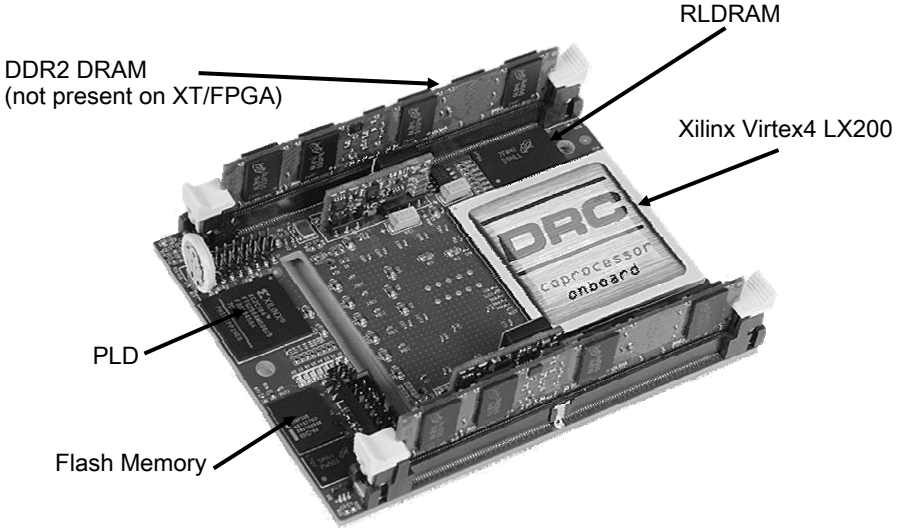
FAP System Architecture

FSB ≡ Front Side Bus
FAP ≡ FSB-FPGA Accelerator Platform
AHM ≡ Accelerator Hardware Module
AFU ≡ Accelerator Function Unit
AAL ≡ Accelerator Abstraction Layer
TPV ≡ Third-Party Vendor

DRC System
(http://www.drccomputer.com/)



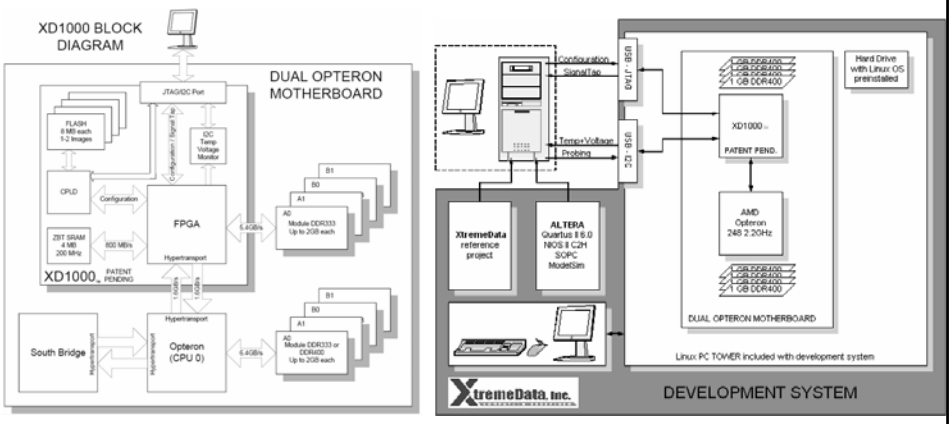
DRC Module Details



XtremeData

(http://www.xtremedatainc.com/)

- **FPGA uses all motherboard resources meant for CPU:**
  - **Intel Processors: Front Side Bus links, memory interface, power supply, heat-sink**
  - **AMD Processors: HyperTransport Links, Memory interface, power supply, heat-sink**
- **Usable with any compatible validated Intel® Xeon® or AMD Opteron server**
- **Mix and match modules and CPUs on quad-socket systems**



Current and Future of XtremeData….

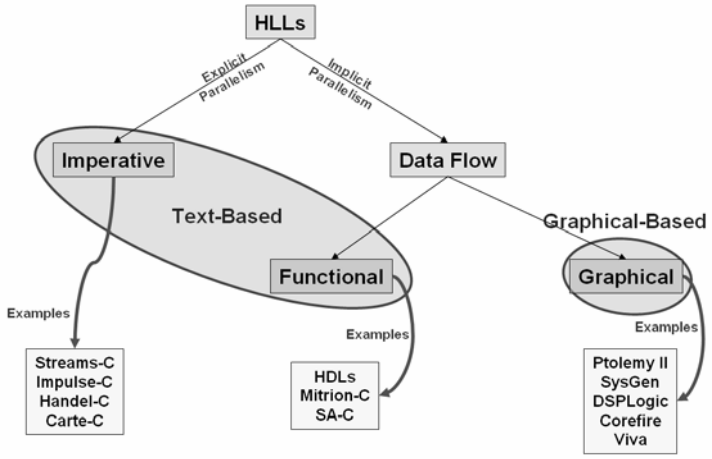
- **Only Company that supports AMD and Intel accelerators**
- **Chosen by Intel to receive FSB license**

Processor Socket		Module	Features	Availability
AMD	Socket E		2S180	Now
	Socket F		2S130 and 2S180 32MB QDRII 20MB/S Mem B/W	Q42007
Intel	Dual Processor		Scalable Footprint 3S80E – 3S340 Any combo of two + Bridge 17MB/S Mem B/W	Q42007
	Multi-processor		Scalable Footprint 3S80E – 3S340 Any combo of two + Bridge 17MB/S Mem B/W	Q12008

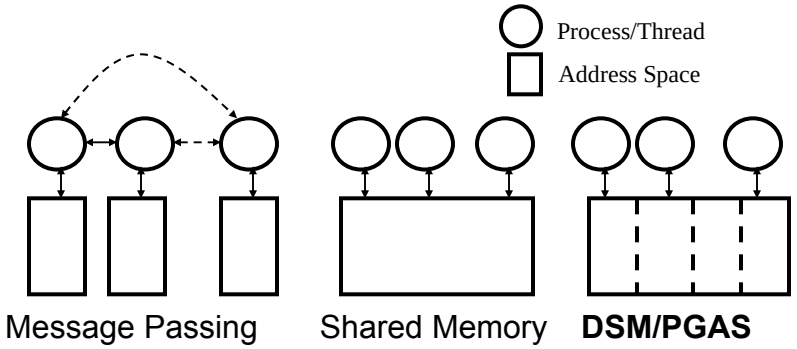
Outline

- Architectures and Systems
- Tools and Programming
- Applications
- Performance
- Wrap-up

HLLs Classification



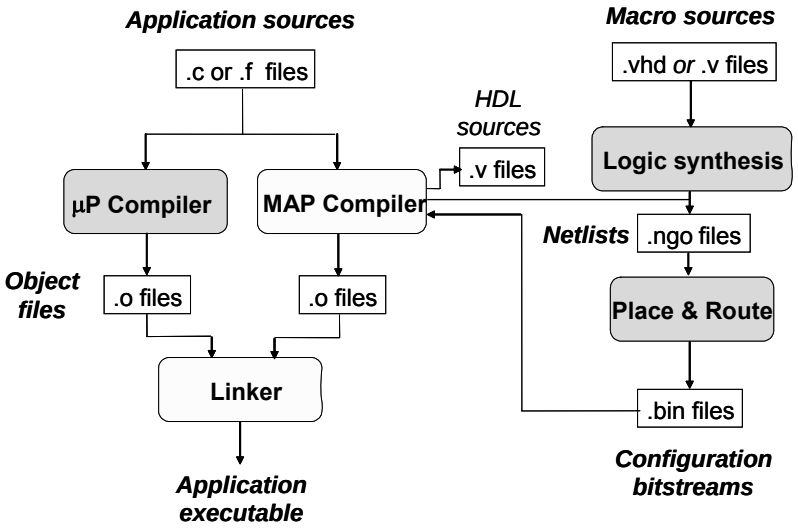
Programming Models: Expressing Parallelism and Locality in Imperative Languages



Introduction to MAP C

- **Available only for SRC machines**
- **MAP FORTRAN also exists**
- **MAP C differs from ANSI C**
  - Some ANSI C features are not available
    - No global variables
    - No external function calls
      - Other than user-defined or SRC-defined macros, or inlined functions
    - No structures
    - No switch statement, etc.
  - Extensive use of concepts and macros not present in C

SRC Compilation Process



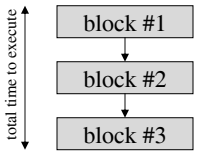
MAP Routines

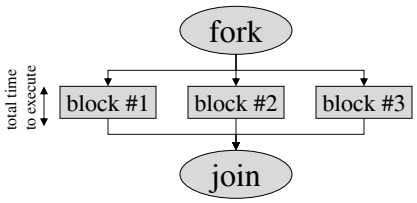
- **Microprocessor side**
  - .c File
  - Function prototype
    - void subr(int64_t*, int);
  - Allocation of MAP
    - int map_allocate(int nm);
    - int map_free(int nm);
  - Calling MAP function
    - subr(array, mapnum);
- **MAP side**
  - .mc File
  - Function implementation

```
void subr(int64_t A[], int mn)
{

// code goes here

}
```

Parallel Sections

- **Sequential code**
  - By default, only one code block is active at any time
- **Parallel code sections**
  - Multiple code sections can be active at the same time



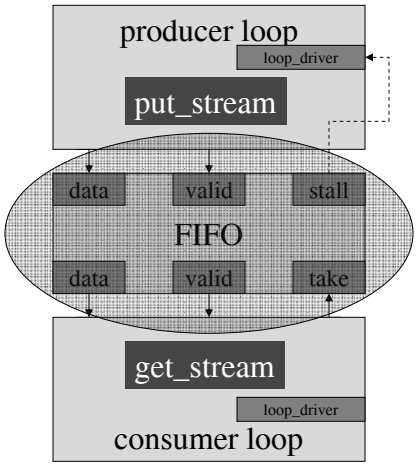


Parallel Sections

```
#pragma src parallel sections
{
   #pragma src section
   {
      sum1 = a + b;
   }

   #pragma src section
   {
      sum2 = a - b;
   }

   #pragma src section
   {
      prod1 = a * b;
   }
}

res = sum1 + sum2 + prod1;
```

Streams

- **Streams mechanics**



- **Code example**

```
Stream_64 S0;
#pragma src parallel sections
{
   #pragma src section
   {
      for (i=0;i<10;i++) {
         res1 = A[i] << 2;
         put_stream(&S0, res1, 1);
      }
   }

   #pragma src section
   {
      for (j=0;j<10;j++) {
         get_stream(&S0, &val);
         B[j] = val + 100;
      }
   }
}
```

Data storage

- **Scalar values can be stored in the "registers" – memory created on-chip from LUTs**
  - float val1, val2;
- **Arrays can be stored in OBM**
  - OBM_BANK_A (AL, long long, 128)
  - OBM_BANK_B_2_arrays (Bi, int64_t, 128, double Bd, 2048)
    - accessible as AL[i], Bi[j], Bd[k]
- **or BRAM**
  - int Ci[128];
  - float Cd[2048];
    - accessible as Ci[i], Cd[j]

Data movement

- **Scalar values via MAP function arguments**
  - void subr(int64_t A[], int n, int64_t *time, int mapnum)
- **Arrays via DMA transfer to OBM**
  - DMA_CPU (CM2OBM, AL, MAP_OBM_stripe(1,"A"), A, 1, n*sizeof(int64_t), 0);
- **Arrays via streams**
  - stream_dma_cpu(&S0, PORT_TO_STREAM, AL, DMA_A, A, 1, n*sizeof(int64_t));
  - get_stream(&S0, &val);

Data movement: DMA transfer

```
void subr(int64_t A[], int64_t C[], int n, int64_t *time, int mapnum)
{
   …
   OBM_BANK_A (AL, int64_t, MAX_OBM_SIZE)
   OBM_BANK_C (CL, int64_t, MAX_OBM_SIZE)
   …
   DMA_CPU (CM2OBM, AL, MAP_OBM_stripe(1,"A"),  A, 1, n*sizeof(int64_t), 0);
   wait_DMA (0);
   …
   // do something useful
   …
   *time = *time + (t1 - t0);
   …
   DMA_CPU (OBM2CM, CL, MAP_OBM_stripe(1,"C"), C, 1, n*sizeof(int64_t), 0);
   wait_DMA (0);
   …
}
```

Data movement: streaming

```
void subr(int64_t A[], int64_t C[], int n, int64_t *time, int mapnum) {
   …
   OBM_BANK_A (AL, int64_t, MAX_OBM_SIZE)
   int64_t a[SIZE];    // SIZE >= n
   Stream_64 S0;
   …
   #pragma src parallel sections
   {
      #pragma src section
      {
         stream_dma_cpu(&S0, PORT_TO_STREAM, AL, DMA_A, A, 1, n*sizeof(int64_t));
      }

      #pragma src section
      {
         for (i = 0; i < n; i++)
         {
            get_stream(&S0, &a[i]);
            .. Operate on Data ..
         }
      }
   }
```

Data packing/unpacking

- **Data access element size for OBM and streams is 64-bit wide**
- **Various split/combine macros allow splitting and combining scalar values, for example:**

```
int64_t v;
int i;
float f;
comb_32to64_int_flt(1234, 0.1234f, &v);
split_64to32_int_flt(v, &i, &f);
result:
   i = 1234
   f = 0.1234
```

Loops

```
for (i = 0; i < n; i++) {
    A[i] = B[i] * C[i];
}

while (i < n) {
    A[i] = B[i] * C[i];
    i++;
}

break
continue
```

```
do {
    A[i] = B[i] * C[i];
    i++;
} while ( i < n );


for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
        A[i] += B[i] * C[j];
    }
}
```

Introduction to Impulse-C

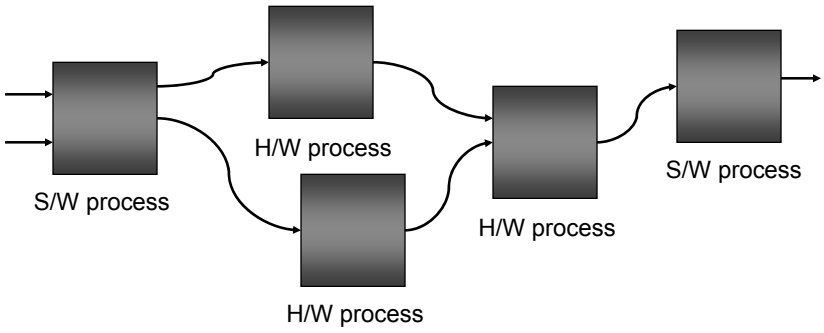
What is Impulse C?

- Not a new language
  - A Subset of ISO C + a library, just like MPI
- A library of functions compatible with standard C
  - Functions for application partitioning
  - Functions for creating and configuring the application architecture
    - Functions for creating processes and streams
    - Functions for connecting streams
    - Functions for mapping into the vendor platform
  - Functions for desktop simulation and instrumentation
- A software-to-hardware compiler

Impulse-C Programming Model

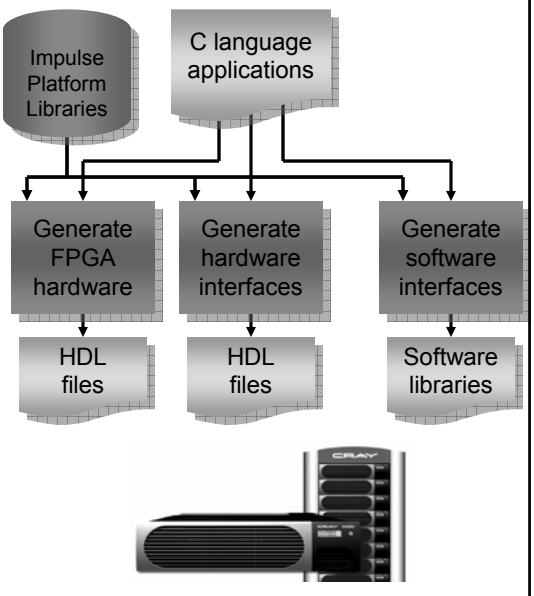
- Communicating Sequential Processes (CSP) Programming Model, also like MPI
  - Supports parallelism at the process level
- As much parallelism as possible is exploited within the processes via automated scheduling/pipelining by the compiler
- Streams for interprocess (inter-functional units) communications
  - Buffered communication channels (FIFOs)
  - In a sense similar to MPI messages, but may be more like Unix pipes

Programming Model



Programming with Impulse C

1. Use Impulse C functions to partition the application into hardware and software processes
   - Create the processes
   - Create input and output Streams
   - Connect the streams
2. Use Impulse C to compile hardware processes to HDL and generate hardware stream and memory interfaces.



Elements of an Impulse-C Application

- main()
  - Entry point for the software side of the application
- Configuration function
  - e.g. config()
  - Defines the parallel Impulse C processes
  - Creates streams
  - Connects stream
- co_initialize()
  - Creates the entire application H/W architecture targeting a specific platform
- co_execute()
  - Starts the parallel Impulse C processes
- One or more Impulse C processes
  - Define the behavior of the application, including test producer and consumer functions as required

Impulse C Process Coding Style

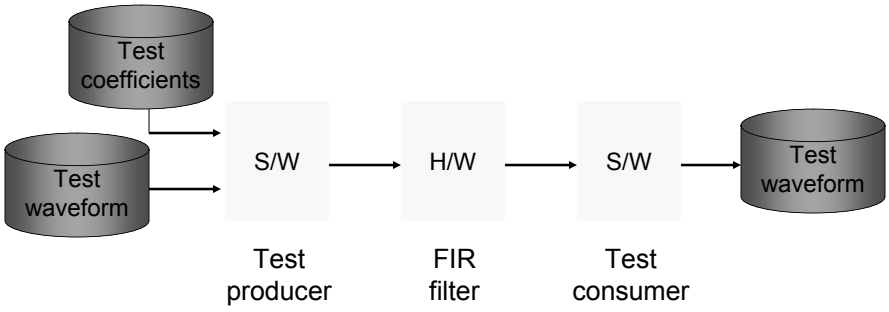
- Written as a C function
  - Accepts pointers to streams, signals, memories, etc.
  - Accepts optional compile-time parameters
  - No return value

```
void des_ic(co_stream filter_in, co_stream filter_out) {
    int32 data;
    co_stream_open(filter_in, O_RDONLY, INT_TYPE(32));
    co_stream_open(filter_out, O_WRONLY, INT_TYPE(32));
    while (co_stream_read(filter_in, &data, sizeof(int32))) {
    . . . // Process the data here
        co_stream_write(filter_out, &data, sizeof(int32));
    }
    co_stream_close(filter_in);
    co_stream_close(filter_out);
}
```

Example: FIR Filter

- Data and coefficients passed into filter via data stream
  - Could also use shared memory
- Algorithm written using untimed, hardware-independent C code
  - Using coding styles familiar to C programmers
- Software test bench written in C to test functionality
  - In software simulation
  - In actual hardware

FIR Filter Functional Test



This test can be performed in desktop simulation (using Visual Studio or some other C environment) or can be performed using an embedded processor for the producer/consumer modules.

impulse accelerated technologies

```
void fir(co_stream filter_in, co_stream filter_out) {
   int32 coef[TAPS]; int32 firbuffer[TAPS];
   int32 nSample, nFiltered, accum, tap;

   co_stream_open(filter_in, O_RDONLY, INT_TYPE(32));
   co_stream_open(filter_out, O_WRONLY, INT_TYPE(32));
   // First fill the coef array with the coefficients...
   for (tap = 0; tap < TAPS; tap++) {
      co_stream_read(filter_in, &nSample, sizeof(int32));
      coef[tap] = nSample;
   }
   // Now fill the firbuffer array with the first n values...
   for (tap = 1; tap < TAPS; tap++) {
      co_stream_read(filter_in, &nSample, sizeof(int32));
      firbuffer[tap-1] = nSample;
   }
   // Now we have an almost full buffer and can start processing waveform samples…
   while ( co_stream_read(filter_in, &nSample, sizeof(int32)) == co_err_none ) {
      firbuffer[TAPS-1] = nSample;
      for (accum = 0; tap = 0; tap < TAPS; tap++) {
         accum += firbuffer[tap] * coef[tap];
      }
      nFiltered = accum >> 2;
      co_stream_write(filter_out, &nFiltered, sizeof(int32));
      for (tap = 1; tap < TAPS; tap++) {
         firbuffer[tap-1] = firbuffer[tap];
      }
   }
   co_stream_close(filter_in);
   co_stream_close(filter_out);
}
```

Declare stream interfaces

Open the streams

Read in the coefficients

Read in the first *n* values

Process the incoming stream and perform the filter operation to generate outputs

When done, close the streams

Impulse C Configuration Function

```
void config_fir(void *arg)
{
   co_stream waveform_raw;
   co_stream waveform_filtered;

   co_process producer_process;
   co_process fir_process;
   co_process consumer_process;

   waveform_raw = co_stream_create("waveform_raw", INT_TYPE(32), BUFSIZE);
   waveform_filtered = co_stream_create("waveform_filtered", INT_TYPE(32), BUFSIZE);

   producer_process = co_process_create("producer_process", (co_function)test_producer,
                                        1, waveform_raw);

   fir_process = co_process_create("filter_process", (co_function)fir,
                                   2, waveform_raw, waveform_filtered);

   consumer_process = co_process_create("consumer_process",(co_function)test_consumer,
                                        1, waveform_filtered);

   // Assign processes to hardware elements
   co_process_config(fir_process, co_loc, "PE0");
}
```

stream declarations

process declarations

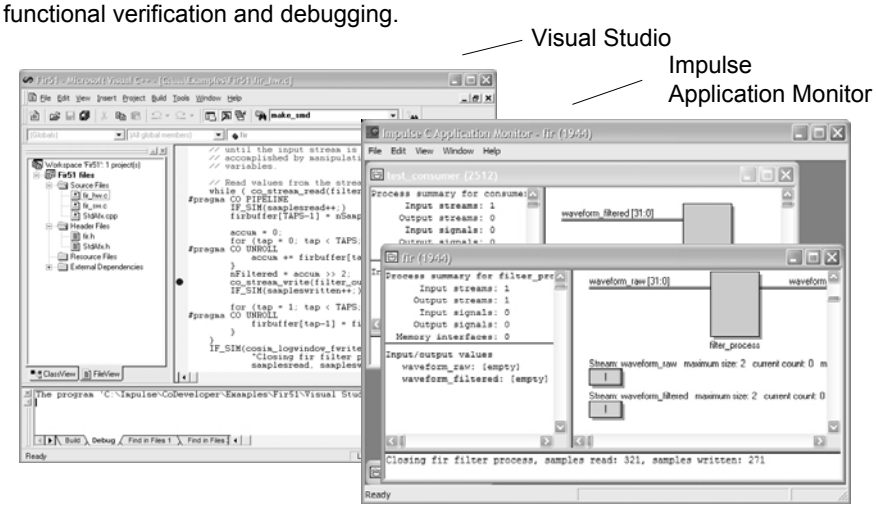
stream creation

process creation and stream connection

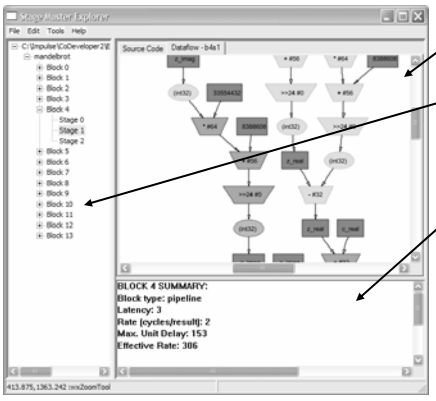
process configuration (hardware)

Desktop Simulation

Impulse C is standard C with the addition of the Impulse C libraries, which means that any standard C development environment can be used for functional verification and debugging.



Optimization Exploration

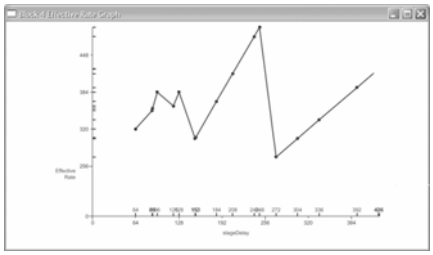


Dataflow graph shows hardware structures generated from C code

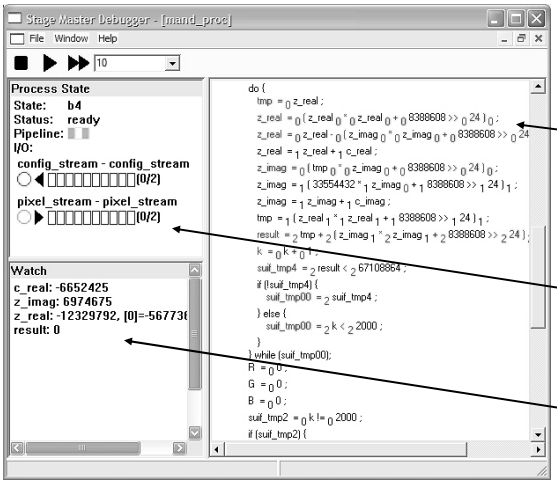
Explorer window shows blocks and stages of C source code

C code block summary shows results of optimization

Pipeline effective rate graph helps analyze performance



Cycle-Accurate Debugging



Expanded source code window shows active pipeline stages for each cycle (in red)

Stream I/O and pipeline status window monitors current status

Variable watch window allows the observation of specific data objects

Introduction to Mitrion-C

Mitrion-C Overview

- Looks like C, very similar syntax
- Behaves differently,
  - Mitrion C is a single assignment language, uses a functional programming model, it is data driven and not program counter driven
  - Data driven computations are inherently parallel
  - C is an imperative language, algorithm is described in terms of sequential statements and execution has states
  - C is inherently sequential and expressing parallelism requires extensions to the language

Mitrion-C and C Are Syntactically Similar

- Mitrion-C is a C-family language but not ISO-C
- Basic syntax is similar to other C-family languages
  - Blocks are surrounded by `{ }`
  - Assignments are made by `=`
  - Statements end with `;`
  - Common Selection and Iteration Statements are available (`if`, `for`, `while`), but the semantics are different
  - Common C operators are supported
  - C-style comments (but nesting is allowed)

Differences Between Mitrion-C and C

- Single assignment for variables
- Order is data driven not dependency driven
- `main()` is the only point of input and output to the program
- No pointers, pointers do not lend themselves to FPGAs
- No dynamic allocation
- Recursion only with statically known depth, need to know the resources ahead of time
- Flexible bit width supported for scalar types

Single Assignment

- A variable may only be assigned once
  - Operations occur concurrently
  - Multiple assignments of variables can cause inconsistency

```
x = 125;
a = x*4;
x = b+6; // Error!
         // Is x b+6 or 125?
```

Data Driven Model

```
(int:33, int:32) sqradd(int:32 s, int:16 a)
{
       sum = s + sqr;
       sqr = a*a;
} (sum, sqr);


uint:22<30> main() //returns a list of 30 22-bit items
{
  uint:22 prev  = 1;
  uint:22 fib   = 1;

  uint:22<30> fibonacci = for(i in <1..30>)
  {
    fib  = fib+prev;
    prev = fib;
  } ><fib;

} fibonacci;
```


Scoping and Blocks

- **Blocks are used as body for** `if`, `while`, `for`, **...**

```
uint:4 q = 0;                                  Parent Scope
int:16 x = if(i > 5)                           Child Scope
{
     q = f(a, i); // Shadows parent q
} q               // Return q and place it in x
else { ... };
// q has the value 0 here
```

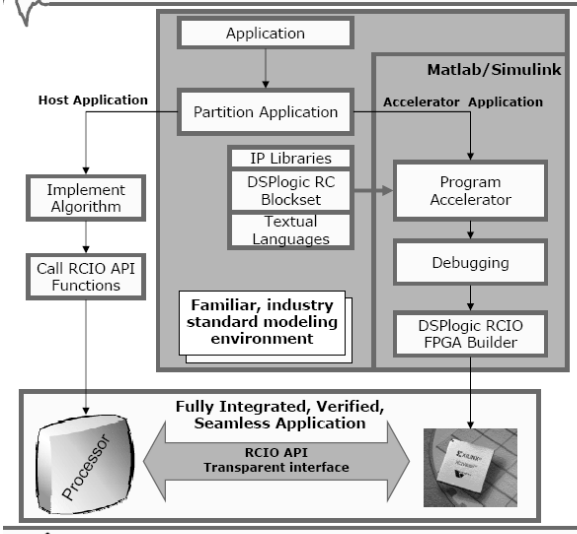
- **Each block defines a scope**
- **Only returned values and *nothing else* can exit the block**
- **Reassignments shadow outer variables (create local variables of the same name)**
  - **Can not modify outer variable**

Loops and Collections

	List	Vector
foreach	Pipelined	Wide parallel
for	Sequential	Unrolled

Introduction to DSPlogic RC Toolbox

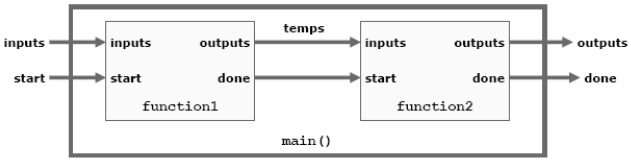
Programming Flow



- **Maximize Acceleration**
- **Minimize Costs**
- **Simple Application messaging**
- **Graphical program flow constructs**
- **Optimized core libraries**
  - **Efficient resource utilization**
- **Text-based debugging, verification, analysis**
- **Portable one-click compiler**
- **Floating-point capability**

Sequential Programming

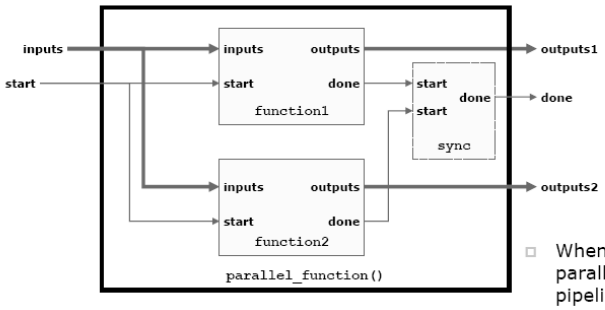
- Example main() program




```
// Equivalent pseudo code

outputs main(inputs){
  temps   = function1(inputs);
  outputs = function2(temps);
}
```

- Note hierarchical variable scope
- Globals are also easily defined

Parallel Programming (with synchronization)



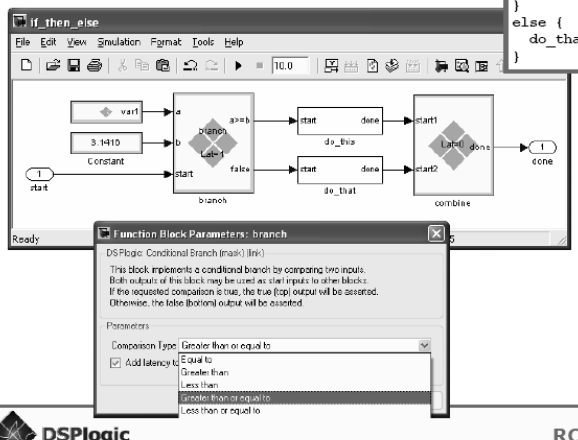

```
// Equivalent pseudo code

parallel_function(inputs, outputs1, outputs2){
  #pragma parallel
  outputs1 = function1(inputs);
  outputs2 = function2(inputs);
}
```

- When possible, utilize parallelism **and** pipelining for maximum speed

Conditional Branch (if-then-else)

- If condition is true, initiate a task
  - otherwise, initiate an alternate task
- Requires 'combine' block to complete statement

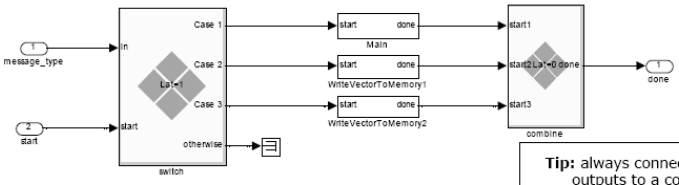
```
// Equivalent pseudo code

if (a xx b)  {
  do_this();
}
else {
  do_that():
}
```



Switch statement

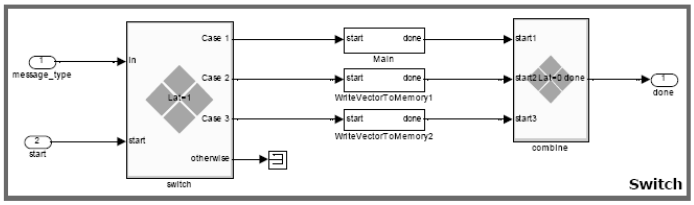
- Performs one of several actions based upon the value of an input variable
- A switch statement requires the use of two RC blocks
  - Switch
    - Used to generate a 'start' for each function
  - Combine
    - Generates required 'done' indicator for the following function
- Quiz: Can you find a potential bug in the function below?

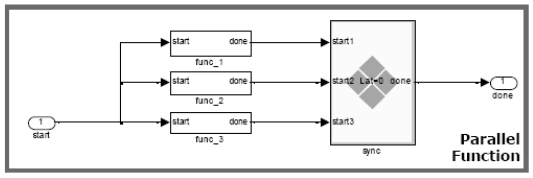
```
// Equivalent pseudo code

switch (message_type)
{
  case 1:
    main();
    break;
  case 2:
    WriteVectorToMemory1();
    break;
  case 3:
    WriteVectorToMemory2();
    break;
  default:
    do nothing;
}
```



Tip: always connect all switch outputs to a combine block

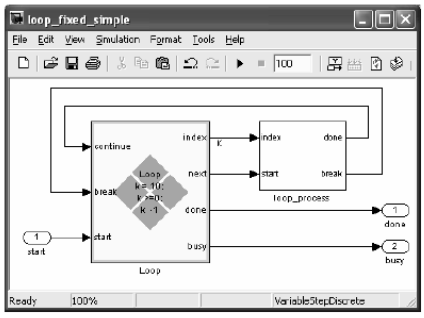
Switch vs. Parallel operations

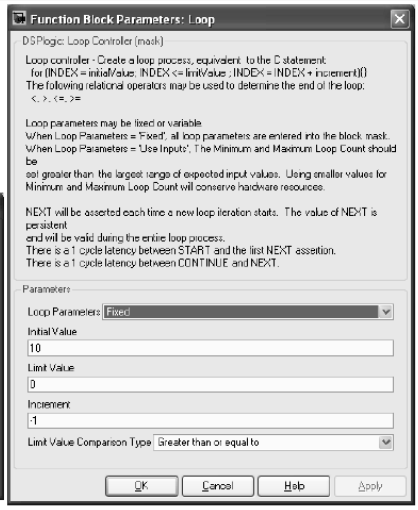




Iterative Constructs (Loops)

```
// Equivalent software operation

for (k=10;k>=0;k=k-1){
  loop_function(index);
}
```

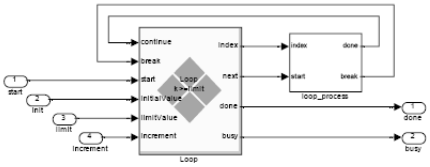


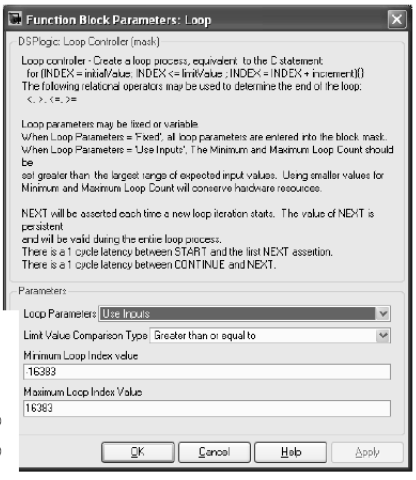


Simple loop – variable index parameters

- The loop may be executed a different number of times, based on an input variable.
- Set Loop Parameters = Use Inputs

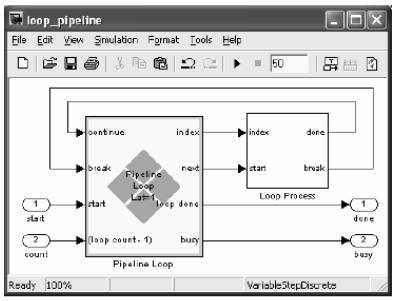
Tip: Set Min and Max loop index to smallest expected range to optimize performance and resources





Pipelined Loop

```
// Equivalent software operation

for (k=0;k<loop_count;k++){
  loop_function(data1);
}
```

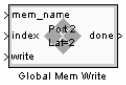


- Simple loop execution
  - 'Loop' initiates execution of 'loop_function' in accordance with loop equation
  - 'K' is updated and valid on each 'loop_function/start'
- Pipelined
  - 'Loop' will start 'loop_function' on consecutive clock cycles until loop is complete or a 'break' event occurs.
- Uses
  - When maximum loop execution speed is required
  - When loop_function() has no data dependencies.
- Automatic pipeline latency handling
  - 'Pipeline Loop/done' is not asserted until 'loop_function' pipeline is completed.
- Break
  - 'loop_function' can force a break from loop any time
- Break with pipelining handled automatically
  - Break may occur while previous computations have already been initiated by a 'start'
  - 'loop_function' should produce one 'done' for each 'start' received
  - 'Pipeline Loop' will be done only after break *and* pipeline clears

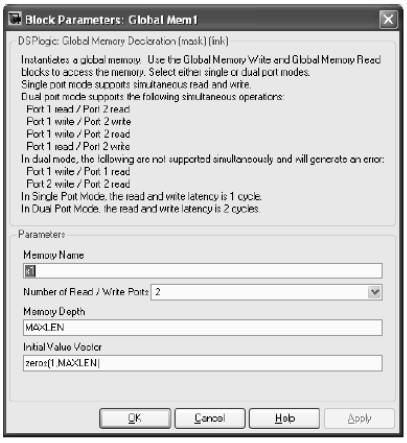
Memory / Arrays

- Declaration

- Write Access
  - Select Port
  - Variable type automatically determined
  - All ports must write same type



- Read Access



HLLs Productivity Study

- **Classic HDLs**
  - VHDL/Verilog
- **Text-based HLLs**
  - Impulse-C
  - Handel-C
  - Carte-C
  - Mitrion-C

→ **Imperative Languages** (Impulse-C, Handel-C, Carte-C)

→ **Functional Languages** (Mitrion-C)

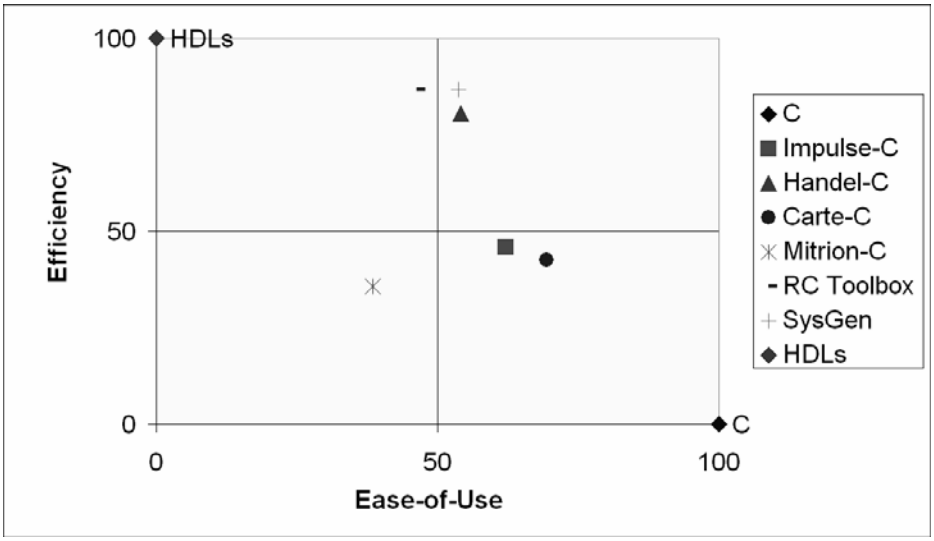
- **Graphical tools**
  - SysGen
  - DSPLogic

→ **Dataflow Languages** (SysGen, DSPLogic)

Evaluation Metrics and Parameters

- Ease-of-Use
  - Acquisition time
  - Development time
- Efficiency (Quality of performance)
  - Synthesized frequency
  - Area utilization

Using the Evaluation Framework



CHREC NSF Center for High-Performance Reconfigurable Computing

UF UNIVERSITY of FLORIDA Virginia Tech BYU BRIGHAM YOUNG

Outline

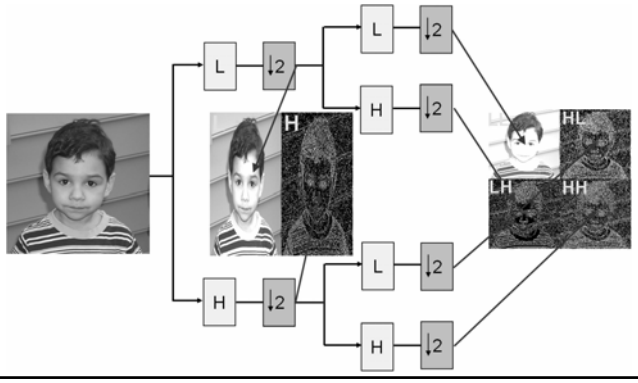
- Architectures and Systems
- Tools and Programming
- Applications
  - Remote Sensing
    - Discrete Wavelet Transform (DWT)
    - Wavelet-Based Hyperspectral Dimension Reduction
    - Image Registration
    - Cloud Detection
  - Bioinformatics
- Performance
- Wrap-up

Outline

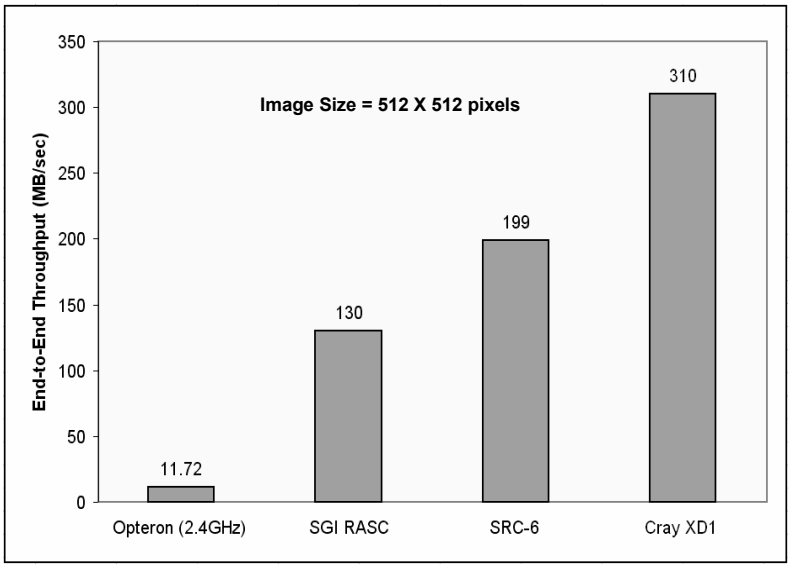
- On-Board Processing for Remote Sensing
  - Discrete Wavelet Transform (DWT)
  - Wavelet-Based Hyperspectral Dimension Reduction
  - Cloud Detection
  - Image Registration
- Bioinformatics

Multi-Resolution DWT Decomposition
(Mallat Algorithm)

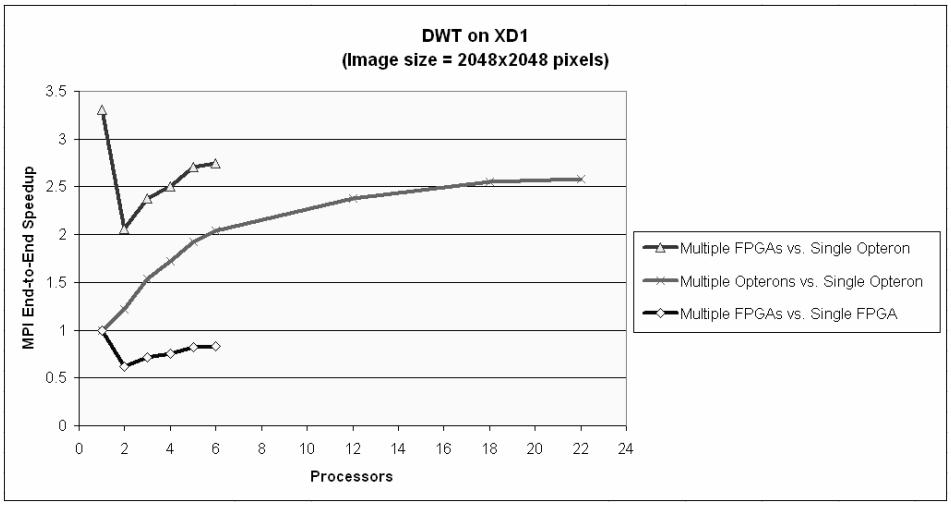
- The input image is first convolved along the rows by the two filters L and H and decimated along the columns by two resulting in two "column-decimated" images L and H
- Each of the two images, L and H, is then convolved along the columns by the two filters L and H and decimated along the rows by two
- This decomposition results into four images, LL, LH, HL and HH
- The LL image is taken as the new input to perform the next level of decomposition



DWT Decomposition
(One Engine → One FPGA)



DWT Decomposition (cnt'd)
(Cray-XD1)

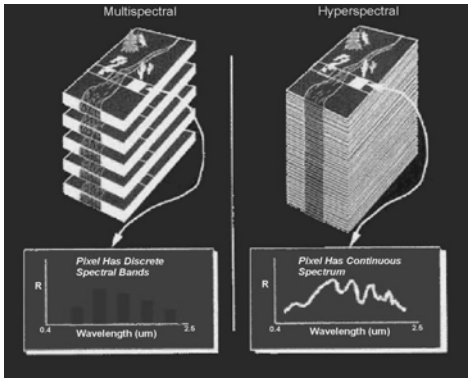


Outline

- **On-Board Processing for Remote Sensing**
  - **Discrete Wavelet Transform (DWT)**
  - **Wavelet-Based Hyperspectral Dimension Reduction**
  - **Cloud Detection**
  - **Image Registration**
- **Bioinformatics**

Hyperspectral Dimension Reduction

- Multi-Spectral Imagery → 10's of bands (MODIS ≡ 36 bands, SeaWiFS ≡ 8 bands, IKONOS ≡ 5 bands)
- Hyperspectral Imagery → 100's-1000's of bands (AVIRIS ≡ 224 bands, AIRS ≡ 2378 bands)
  - Challenges (Curse of Dimensionality)
  - Solution
    - **On-Board Dimension Reduction**
  - Needs
    - **Higher performance**
    - **Lower form / wrap factors** ➡ **HPRCs**
    - **Higher flexibility**

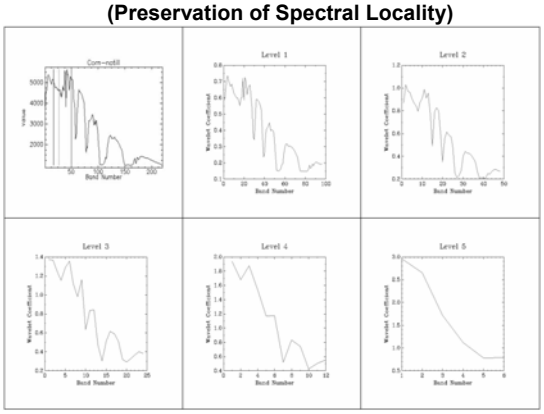


Multispectral / Hyperspectral Imagery Comparison

Hyperspectral Dimension Reduction
(Techniques)

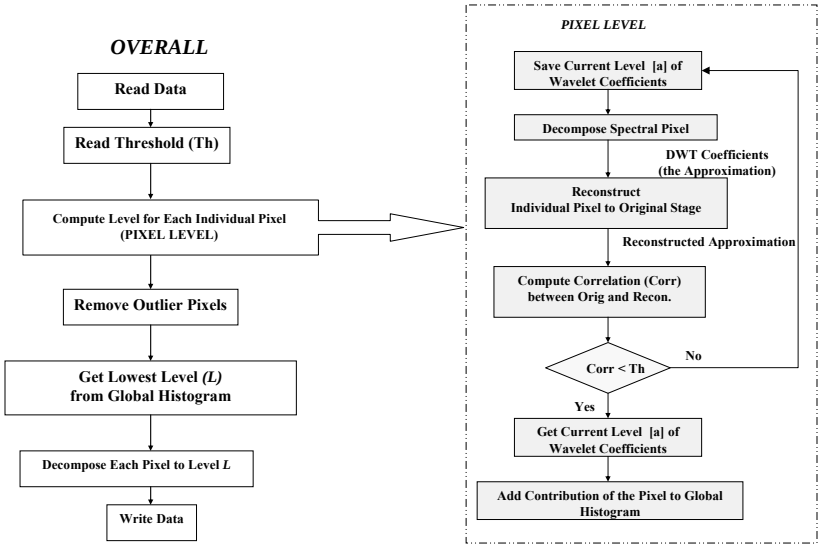
- **Principal Component Analysis (PCA):**
  - Most Common Method Dimension Reduction
  - Complex and Global computations: difficult for parallel processing and hardware implementations
  - Does Not Preserve Spectral Signatures
- **Wavelet-Based Dimension Reduction*:**
  - Simple and Local Operations
  - High-Performance Implementation
  - Preserves Spectral Signatures

Multi-Resolution Wavelet Decomposition of Each Pixel 1-D Spectral Signature (Preservation of Spectral Locality)

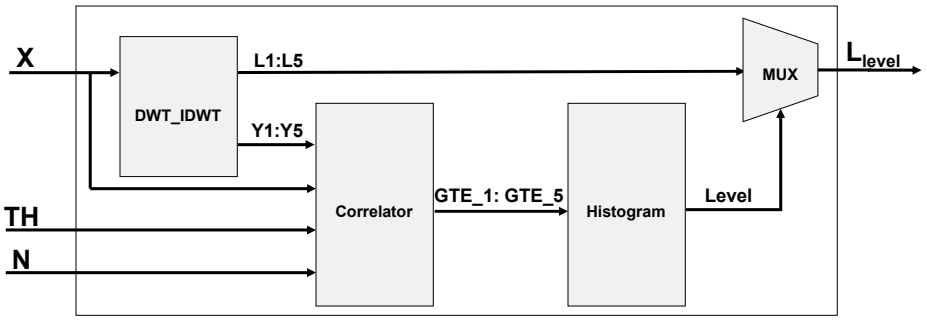


*** S. Kaewpijit, J. Le Moigne, T. El-Ghazawi, "Automatic Reduction of Hyperspectral Imagery Using Wavelet Spectral Analysis", *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 41, No. 4, April, 2003, pp. 863-871.**

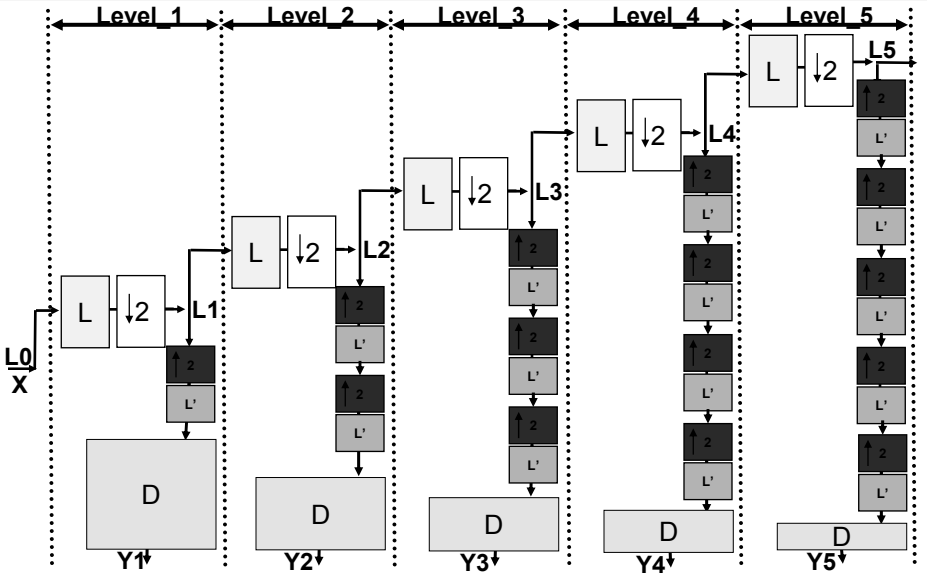
The Algorithm



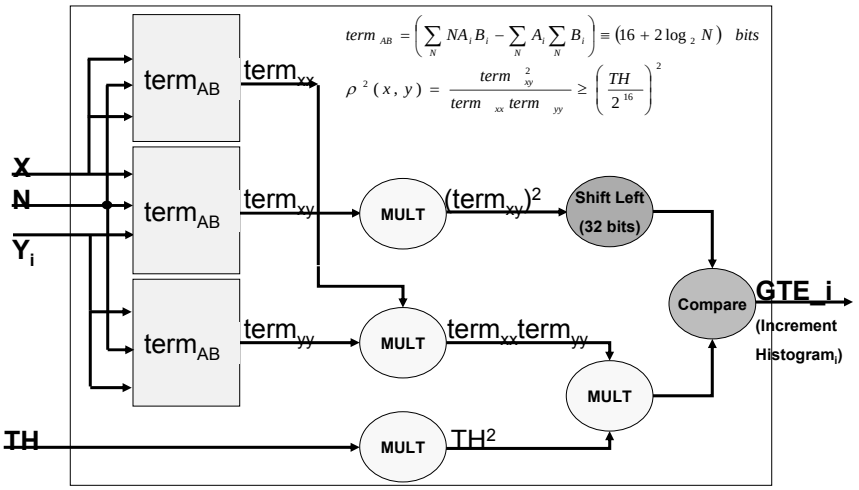
Top Hierarchy Module



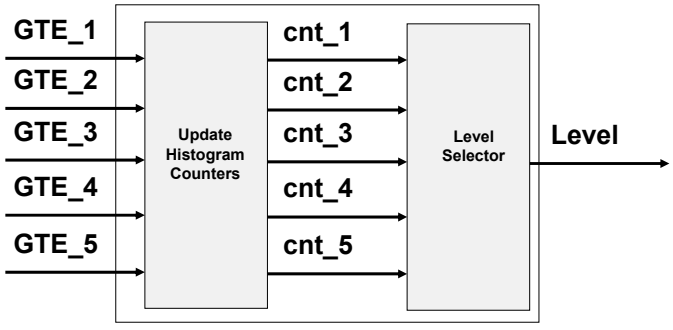
Decomposition and Reconstruction Levels of Dimension Reduction (DWT_IDWT)



Correlator Module

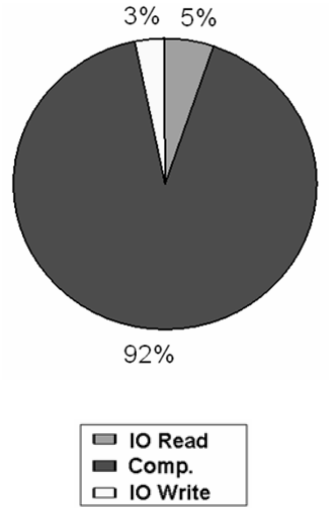


Histogram Module



Wavelet-Based Dimension Reduction
(Execution Profiles on SRC)

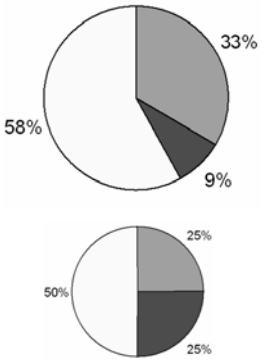
Total Execution Time = 20.21 sec
(Pentium4, 1.8GHz)



Total Execution Time = 1.67 sec (SRC-6E, P3)

Speedup = 12.08 x (without-streaming)

Speedup = 13.21 x (with-streaming)



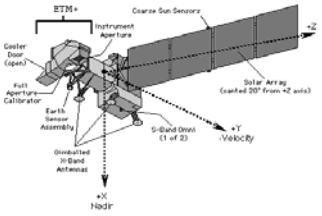
Total Execution Time = 0.84 sec (SRC-6)
Speedup = 24.06 x (without-streaming)
Speedup = 32.04 x (with-streaming)

Outline

- On-Board Processing for Remote Sensing
  - Discrete Wavelet Transform (DWT)
  - Wavelet-Based Hyperspectral Dimension Reduction
  - Cloud Detection
  - Image Registration
- Bioinformatics

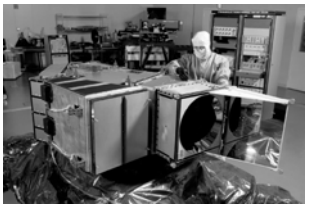
Motivations and Theory of Cloud Detection

- **Why Cloud Detection?**
  - **Can render data useless in land-use/land cover studies**
  - **Critical in weather and climate studies**



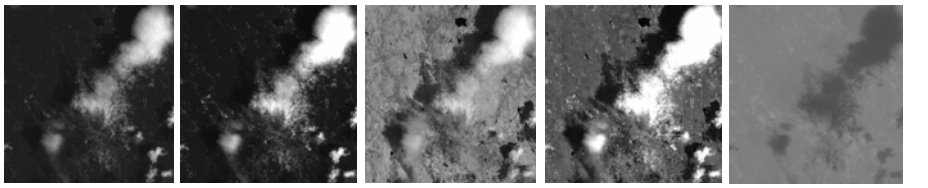
Landsat 7

- **Theory is based on the observation that clouds are:**
  - **Highly reflective (in the visible, near- and mid- IR bands)**
    - **Visible Bands (Green, Red bands)**
      - » Vegetation and land surface discrimination
    - **Near-IR band**
      - » Determines soil moisture level and distinguishes vegetation types
    - **Mid-IR band**
      - » Differentiation of snow from clouds
  - **Cold**
    - **Thermal IR band**
      - » Thermal mapping to Brightness Temperatures

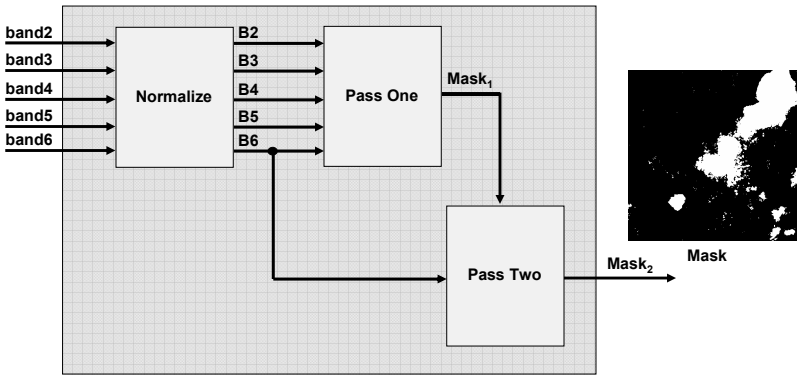


ETM+

Top Hierarchy Module

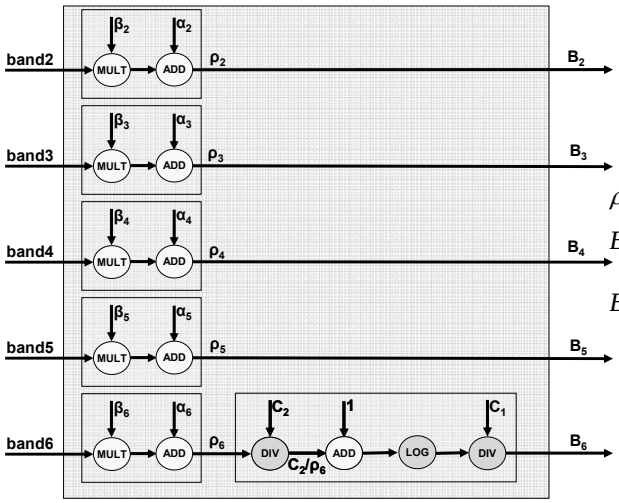


Band 2 (Green Band) **Band 3 (Red Band)** **Band 4 (Near-IR Band)** **Band 5 (Mid-IR Band)** **Band 6 (Thermal IR Band)**



Mask

Normalization Module

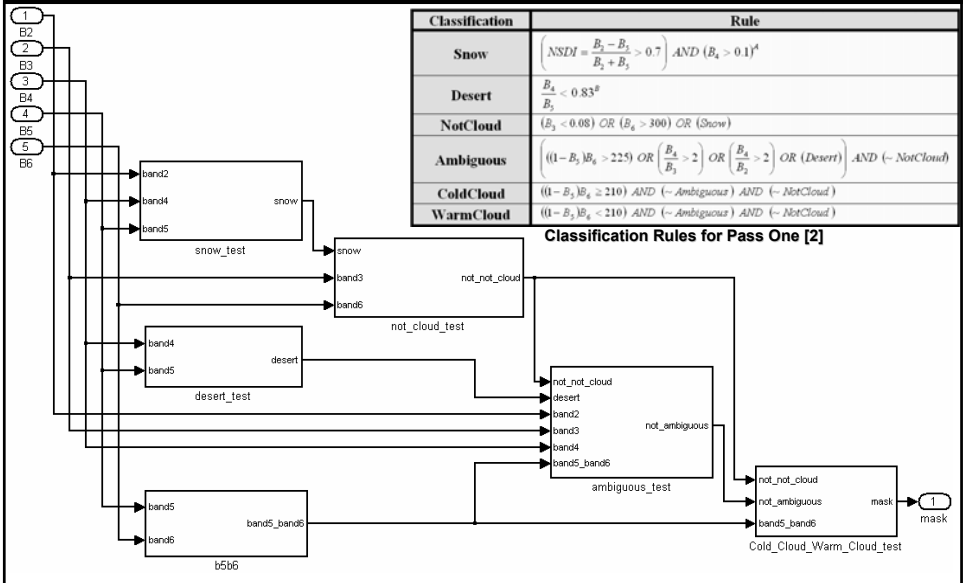


$$\rho_i = \beta_i \times band_i + \alpha_i \ , \ i = 2,3,4,5,6$$

$$B_i = \rho_i \ , \ i = 2,3,4,5$$

$$B_6 = \frac{K_2}{\ln\left(\frac{K_1}{\rho_6} + 1\right)}$$

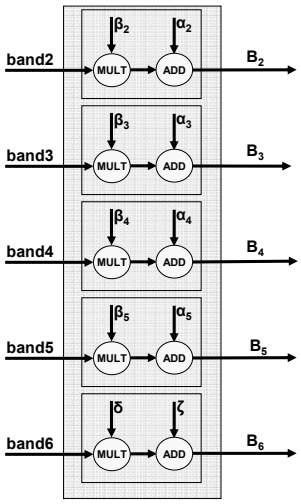
Pass-One Module



Classification	Rule
Snow	$\left(NSDI = \frac{B_2 - B_5}{B_2 + B_5} > 0.7\right) AND\ (B_4 > 0.1)^A$
Desert	$\frac{B_4}{B_5} < 0.83^B$
NotCloud	$(B_3 < 0.08)\ OR\ (B_6 > 300)\ OR\ (Snow)$
Ambiguous	$\left((1 - B_3)B_6 > 225\right) OR \left(\frac{B_4}{B_3} > 2\right) OR \left(\frac{B_4}{B_2} > 2\right) OR\ (Desert)\ AND\ (\sim NotCloud)$
ColdCloud	$((1 - B_3)B_6 \geq 210)\ AND\ (\sim Ambiguous)\ AND\ (\sim NotCloud)$
WarmCloud	$((1 - B_3)B_6 < 210)\ AND\ (\sim Ambiguous)\ AND\ (\sim NotCloud)$

Classification Rules for Pass One [2]

Optimizing Hardware Resources Usage
(Linearization of the Normalization Function)



$$\rho_i = \beta_i \times band_i + \alpha_i \ , \ i = 2,3,4,5,6$$

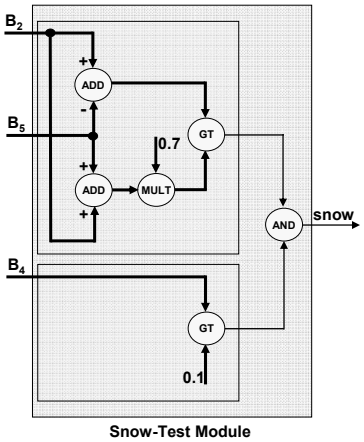
$$B_i = \rho_i \ , \ i = 2,3,4,5$$

$$B_6 = \frac{K_2}{\ln\left(\frac{K_1}{\rho_6}+1\right)} \cong \frac{K_2}{1+\ln(K_1)} + \frac{K_2\left(1-\frac{1}{K_1}\right)}{(1+\ln(K_1))^2} \times \rho_6$$

$$B_6 \cong \left(\frac{K_2}{1+\ln(K_1)} + \frac{K_2\left(1-\frac{1}{K_1}\right)\cdot\alpha_6}{(1+\ln(K_1))^2} \right) + \left(\frac{K_2\left(1-\frac{1}{K_1}\right)\cdot\beta_6}{(1+\ln(K_1))^2} \right) \times band_6$$

$$B_6 \cong \zeta + \delta \times band_6$$

Optimizing Hardware Resources Usage (cnt'd)
(Algebraic Re-Formulation of Pass-One Filters)



Snow-Test Module

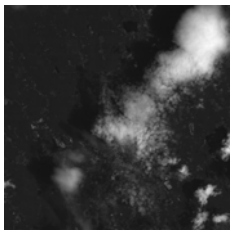
Classification	Rule
Snow	$\left(NSDI = \frac{B_2 - B_5}{B_2 + B_5} > 0.7\right) AND\ (B_4 > 0.1)^A$
Desert	$\frac{B_4}{B_5} < 0.83^B$
NotCloud	$(B_3 < 0.08)\ OR\ (B_6 > 300)\ OR\ (Snow)$
Ambiguous	$\left((1-B_3)B_6 > 225\right)\ OR\left(\frac{B_4}{B_3} > 2\right)\ OR\left(\frac{B_4}{B_2} > 2\right)\ OR\ (Desert)\ AND\ (\sim NotCloud)$
ColdCloud	$((1-B_3)B_6 \geq 210)\ AND\ (\sim Ambiguous)\ AND\ (\sim NotCloud)$
WarmCloud	$((1-B_3)B_6 < 210)\ AND\ (\sim Ambiguous)\ AND\ (\sim NotCloud)$

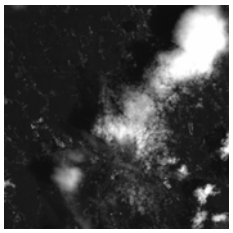
Classification Rules for Pass One [2]

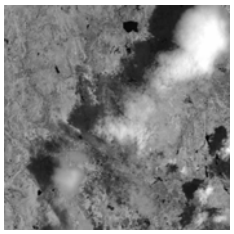
Division Eliminated

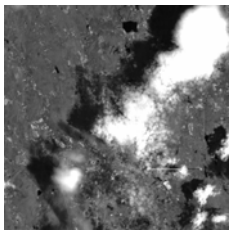
$$\left\{\left(NSDI = \frac{B_2 - B_5}{B_2 + B_5} > 0.7\right) AND\left(B_4 > 0.1\right)\right\} \Leftrightarrow$$
$$\left\{\left(\left(B_2 - B_5\right) > 0.7 \times \left(B_2 + B_5\right)\right) AND\left(B_4 > 0.1\right)\right\}$$

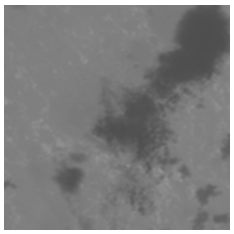
Detection Accuracy
(Software/Reference Mask, Hardware Masks)

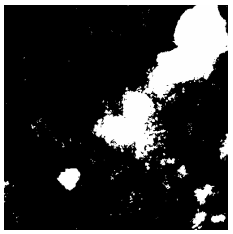

Band 2 (Green Band)

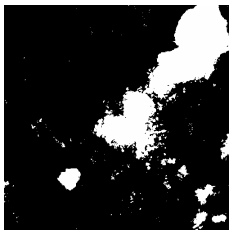

Band 3 (Red Band)

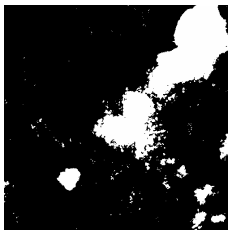

Band 4 (Near-IR Band)


Band 5 (Mid-IR Band)


Band 6 (Thermal IR Band)


Software/Reference Mask

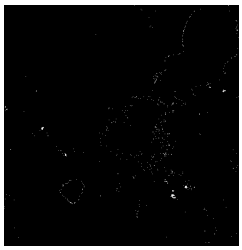

Hardware Floating-Point Mask (Approximate Normalization)


Hardware Fixed-Point Mask (Approximate Normalization)

Detection Accuracy (cnt'd)
(Approximate Normalization and Quantization Errors)


Approximation Error (0.1028 %)


Hardware Floating-Point Error (0.1028 %)


Hardware Fixed-Point (12-bit) Error (0.2676 %)


Hardware Fixed-Point (23-bit) Error (0.1028 %)

$$error = \frac{\sum_{i=0}^{(rows-1)} \sum_{j=0}^{(columns-1)} \left(\left| x_{ij} - y_{ij} \right| \right)}{rows \times columns} ,$$

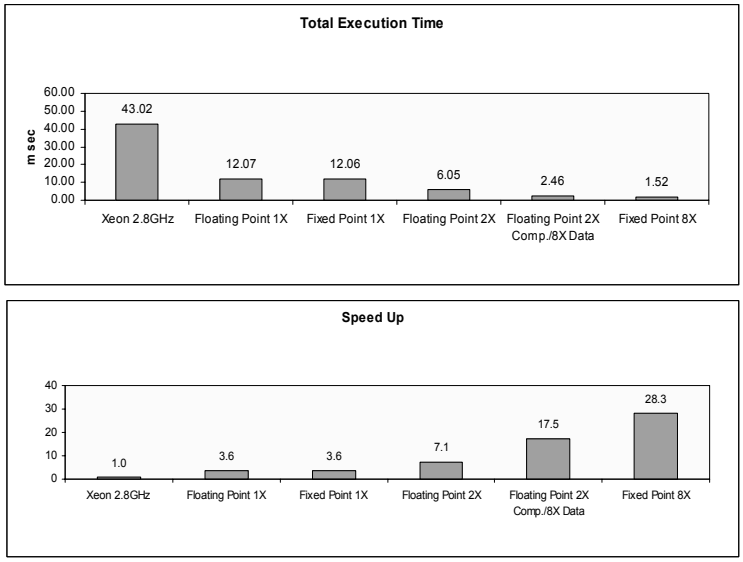
where

$x \equiv output\ image$,

$y \equiv reference\ image$

Reported Error (1.02 %) by Williams et al. [2]

SRC-6 vs. Intel Xeon 2.8 GHz
(Hardware-to-Software Performance)

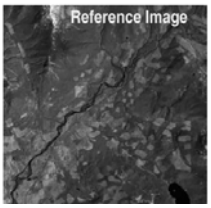


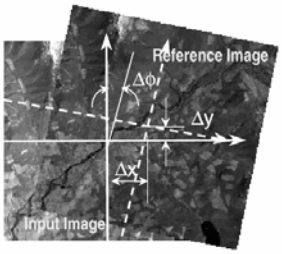
Outline

- On-Board Processing for Remote Sensing
  - Discrete Wavelet Transform (DWT)
  - Wavelet-Based Hyperspectral Dimension Reduction
  - Cloud Detection
  - Image Registration
- Bioinformatics

Introduction and Background

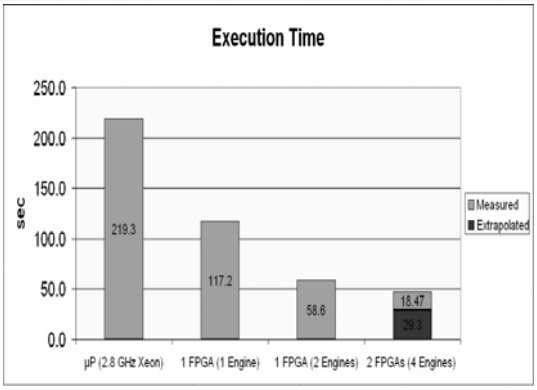
- **Remote sensed data usually contain two types of distortion:**
- **Radiometric distortion**
  - **Sources of radiometric distortion are from**
    - » Effects of atmosphere on radiation,
    - » Effects of atmosphere on remote sensing imagery, and instrumentation errors
  - **These errors can be corrected using the knowledge of the sensor model**
- **Geometric distortion**
  - **Sources of geometric distortion are**
    - » Earth rotation,
    - » Panoramic effects,
    - » Earth curvature,
    - » Scan time skew,
    - » Variation in platform altitude, velocity, attitude, and aspect ratio
  - **To correct the various types of geometric distortion, without the knowledge of error sources, an image can be registered to a map coordinate system:**
    - » The pixels are addressable in terms of map coordinates (latitudes and longitudes or eastings and northings)
    - » The resulting output of image registration is a set of transforms or a mapping function that tells us how the input image is different from the reference image
    - » Using these parameters the input image can be transformed to match the reference image





Implementation Approach and Experimental Results
(SRC-6)

- **Two Techniques**
  - **Exhaustive search**
  - **Iterative refinement**
- **Similarity Measures**
  - **correlation**
  - **Normalized cross-correlation**
    - **Expensive**
    - **One of the best similarity measures**
  - **Statistical correlation**
  - **Match filters**
  - **Phase-correlation**
  - **Sum of absolute differences**
  - **Root mean square**
  - **Masked correlation**
- **Two engines**
  - **79% usage of the chip resources (slices)**
- **High Accuracy**
  - **Floating-point arithmetic (SRC single-precision FP macros)**
- **Extrapolated Higher Performance**
  - **Larger data sizes**
  - **Many optimization techniques such as data streaming**



Platform	Speedup
μP (2.8 GHz Xeon)	1
1 FPGA (1 Engine)	1.87
1 FPGA (2 Engines)	3.74
2 FPGAs (4 Engines)	4.59 (7.48 extrapolated)

DNA Sequencing with Smith-Waterman

- Amino acids
  - The building blocks (monomers) of proteins. 20 different amino acids are used to synthesize proteins. The shape and other properties of each protein is dictated by the precise sequence of amino acids in it.
- Deoxyribonucleic acid (DNA) is written using a code of only 4 letters (bases)
  - Two purines, called adenine (A) and guanine (G)
  - Two pyrimidines, called thymine (T) and cytosine (C)
- DNA sequencing
  - The determination of the precise sequence of nucleotides in a sample of DNA
  - Why – determine origin, …

DNA Matching Basics

- Example:
  - Find the best pairwise alignment of GAATC and CATAC

```
GAATC   GAAT-C   -GAAT-C
CATAC   C-ATAC   C-A-TAC

GAATC-  GAAT-C    GA-ATC
CA-TAC  CA-TAC    CATA-C
```

- We need a way to measure the quality of a candidate alignment
- Alignment scores are driven from :
  - substitution matrix
  - gap penalty

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

A hypothetical substitution matrix

```
GAAT-C
CA-TAC
```

-5 + 10 + ? + 10 + ? + 10 = ?

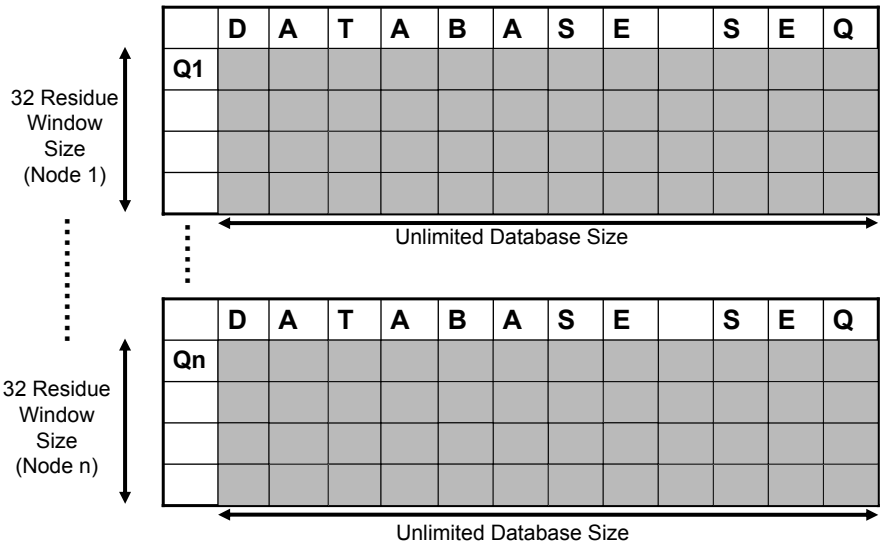
Computing the Scoring Matrix

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + gap\_penalty \\ F(i, j-1) + gap\_penalty \\ 0 \end{cases}$$

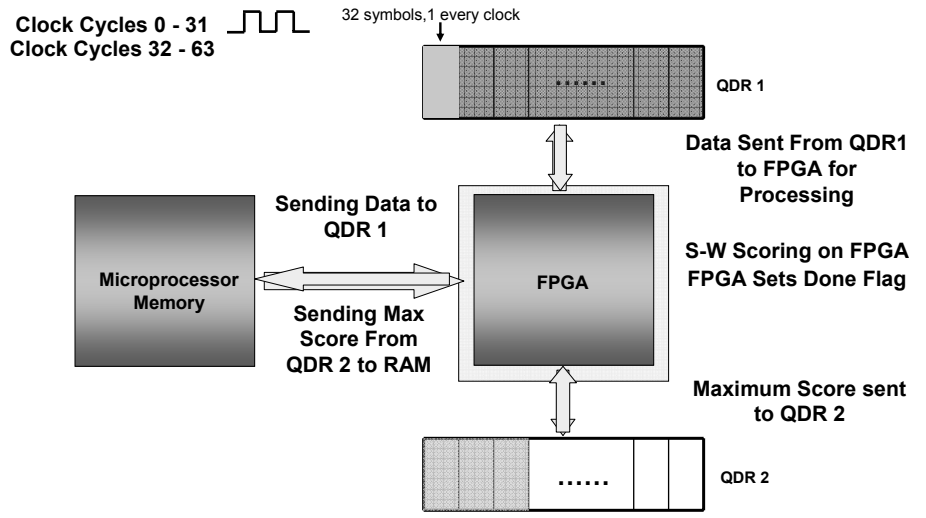
	-	T	T	T	C	A	T	A	G
-	0	2	4	6	8	10	12	14	16
G	2	-1	1	3	5	7	9	11	13
G	4	1	-2	0	2	4	6	8	10
T	6	3	0	-2	-1	1	3	5	7
T	8	5	2	0	-3	-2	0	2	4
A	10	7	4	1	-1	-3	-1	1	1
T	12	9	6	3	0	-2	-3	-4	-2
G	14	11	8	5	2	-1	-3	-4	-4
C	16	13	10	7	4	1	-2	-4	-5

Hardware Implementation
(32x1 Sliding Window)

Multiple Databases and Multiple Queries

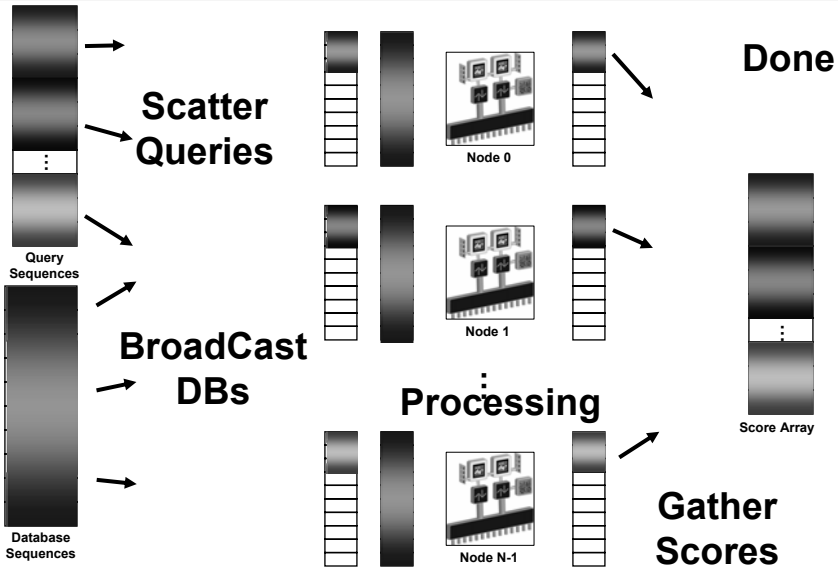


Data Transfer Scenarios

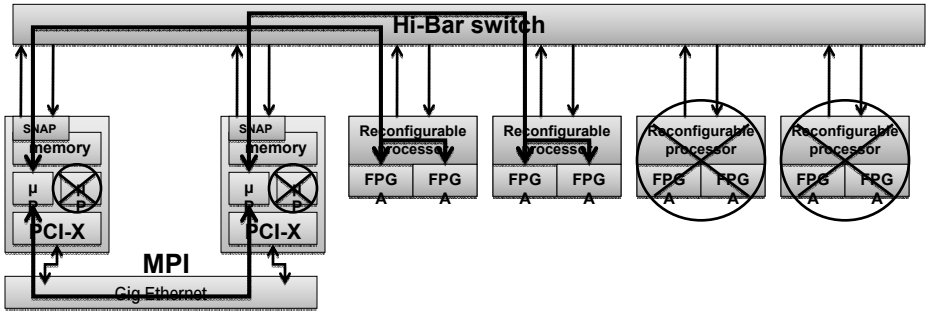


Implementation for Hardware (cnt'd)

(MPI Implementation)

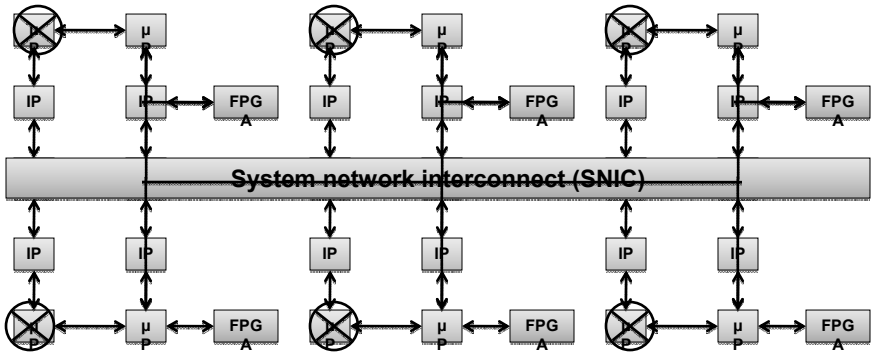


MPI Utilization on SRC-6



- Network Interface Cards cannot be efficiently shared
  - Only two MPI processes were implemented

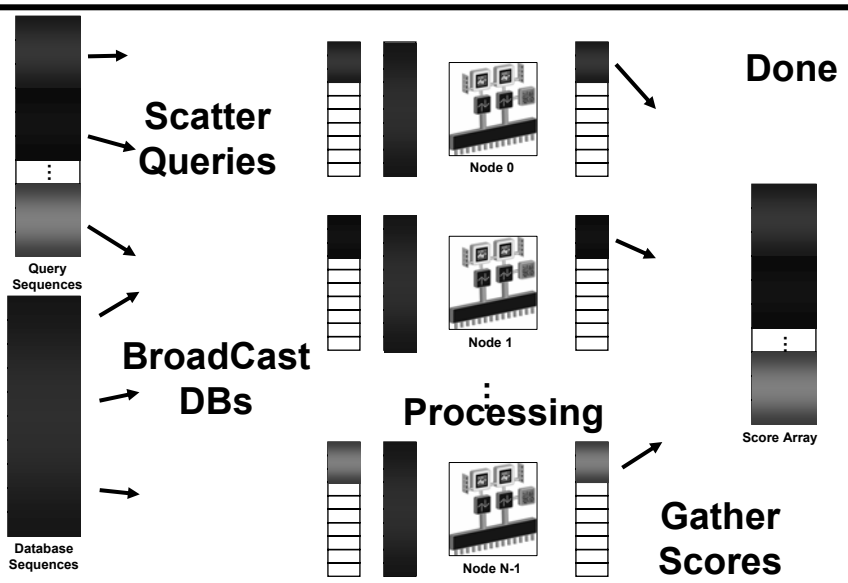
MPI Utilization on Cray-XD1



- All Nodes were exploited using MPI
  - However, only one of the two microprocessors on each node sufficed

Implementation for Hardware (cnt'd)

(MPI Implementation)



Performance Results

				Expected		Measured	
				Throughput (GCUPS)	Speedup	Throughput (GCUPS)	Speedup
FASTA SSEARCH34	Opteron 2.4GHz	DNA		NA	NA	0.065	1
		Protein		NA	NA	0.130	1
GWU	SRC 100 MHz (32x1)	DNA	1 Engine/Chip	3.2	49.2	3.19 → 12.2 1→4 Chips	49 → 188 1→4 Chips
			4 Engines/Chip	12.8	197	12.4 → 42.7 1→4 Chips	191 → 656 1→4 Chips
			8 Engines/Chip	25.6	394	24.1 → 74 1→4 Chips	371 → 1138 1→4 Chips
		Protein		3.2	24.6	3.12 → 11.7 1→4 Chips	24 → 90 1→4 Chips
	XD1 200 MHz (32x1)	DNA	1 Engine/Chip	6.4	98	5.9 → 32 MPI 1→6 nodes	91 → 492 MPI 1→6 nodes
			4 Engines/Chip	25.6	394	23.3 → 120.7 MPI 1→6 nodes	359 → 1857 MPI 1→6 nodes
			8 Engines/Chip	51.2	788	45.2 → 181.6 MPI 1→6 nodes	695 → 2794 MPI 1→6 nodes
		Protein		6.4	49	5.9 → 34 MPI 1→6 nodes	45 → 262 MPI 1→6 nodes

Outline

- Architectures and Systems
- Tools and Programming
- Applications
- Performance
- Wrap-up

Performance

Potential: Comparison with a cluster of microprocessor boards

Assumptions:

- **100% cluster efficiency, i.e., each application can be perfectly parallelized across N microprocessor boards**
- **Each reconfigurable processor is used together with a single dual-µP boards**

System Cost

Cluster of N µPs

cost = N/2 * cost(dual µP board)
+ cost (switch network)

Reconfigurable computer

cost = cost(dual µP board)
+ cost(reconfigurable processor)

System Cost Current Cost Ratio

$$\frac{\textbf{Reconfigurable computer cost}}{\textbf{Dual µP board cost}} \approx 50\text{-}100$$

Power Consumption –SRC-6E

Reconfigurable processor: **200 W**

µP board (with two µPs): **170 W**

Cluster of N single µPs: **N*170/2 W**

Power Consumption Ratio

N - Cluster size (in the number of microprocessors) necessary to obtain equivalent performance

	N	Power consumption advantage Typical reconfigurable computer vs. a cluster of dual µP boards containing N µPs
I/O intensive applications	10	4.25
	100	42.50
Computationally intensive applications	1000	425.00

Power Consumption Cost

Assumptions:

Both systems used non-stop over a 5 year period

Average commercial cost of power in LA, NYC, SF, and DC: $0.12 per kW-hour

Total cost of power over a five year period without cooling

N	Cluster with N µPs	Typical reconfigurable computer	Savings
10	$4,468	$1,051	$3,417
100	$44,680	$1,051	$43,629
1000	$446,800	$1,051	$445,749

Total cost of power over a five year period including cooling

N	Cluster of N µPs	Typical reconfigurable computer	Savings
10	$11,170	$2,628	$8,542
100	$111,700	$2,628	$109,072
1000	$1,117,000	$2,628	$1,114,372

System Size

Cluster of 100 µPs = four 19-inch racks

footprint = 6 square feet

Reconfigurable computer (SRC MAPstation™)

footprint = 1 square foot

Space savings 6 times assuming rack-mounted clusters, and many times more for standard PC-based clusters

Platform Configuration

Platform	Number of FPGA	FPGA Type	Maximum Frequency	Saving Factor (µP : RP)		
				Cost	Power	Size
SRC-6	8	XC2V6000	100MHz	1:200	1:3.64	1:33.3
Cray XD1	6	XC2VP50	200MHz	1:100	1:20	1:95.8
SGI RC-100	6	XC4LX200	200MHz	1:400	1:11.2	1:34.5

Savings of HPRC
(Based on SRC-6)

Application	Speedup	SAVINGS		
		Cost Savings	Power Savings	Size Reduction
Smith-Waterman (DNA Sequencing)	1138	6x	313x	34x
DES Breaker	6757	34x	1856x	203x
IDEA Breaker	641	3x	176x	19x
RC5(32/12/16) Breaker	1140	6x	313x	34x

- **Assumptions**
  - **100% cluster efficiency**
  - **Cost Factor µP : RP → 1 : 200**
  - **Power Factor µP : RP → 1 : 3.64**
    - Reconfigurable processor (based on SRC-6): 200 W
    - µP board (with two µPs): 220 W
  - **Size Factor µP : RP → 1 : 33.3**
    - Cluster of 100 µPs = four 19-inch racks
      - » footprint = 6 square feet
    - Reconfigurable computer (SRC MAPstation™)
      - » footprint = 1 square feet

Savings of HPRC
(Based on one Cray-XD1 chassis)

Application	Speedup	SAVINGS		
		Cost Savings	Power Savings	Size Reduction
Smith-Waterman (DNA Sequencing)	2794	28x	140x	29x
DES Breaker	12162	122x	608x	127x
IDEA Breaker	2402	24x	120x	25x
RC5(32/8/8) Breaker	2321	23x	116x	24x

- **Assumptions**
  - **100% cluster efficiency**
  - **Cost Factor µP : RP → 1 : 100**
  - **Power Factor µP : RP → 1 : 20**
    - Reconfigurable processor (based on one XD1 Chassis): 2200 W
    - µP board (with two µPs): 220 W
  - **Size Factor µP : RP → 1 : 95.8**
    - Cluster of 100 µPs = four 19-inch racks
      - » footprint = 6 square feet
    - Reconfigurable computer (one XD1 Chassis)
      - » footprint = 5.75 square feet

Savings of HPRC
(Based on one Altix 4700 10U rack)

Application	Speedup	SAVINGS		
		Cost Savings	Power Savings	Size Reduction
Smith-Waterman (DNA Sequencing)	8723	22x	779x	253x
DES Breaker	38514	96x	3439x	1116x
IDEA Breaker	961	2x	86x	28x
RC5(32/12/16) Breaker	6838	17x	610x	198x

- **Assumptions**
  - **100% cluster efficiency**
  - **Cost Factor µP : RP → 1 : 400**
  - **Power Factor µP : RP → 1 : 11.2**
    - 1 10U Rack: 1230 W
    - µP board (with two µPs): 220 W
  - **Size Factor µP : RP → 1 : 34.5**
    - Cluster of 100 µPs = four 19-inch racks
      - » footprint = 6 square feet
    - Reconfigurable computer (10U)
      - » footprint = 2.07 square feet

Outline

- **Architectures and Systems**
- **Tools and Programming**
- **Applications**
- **Performance**
- **Wrap-up**

Lessons Learned

- Porting an existing code to an RC platform is difficult
  - Requires an in-depth understanding of the code structure and data flow
  - Code optimization techniques used in the microprocessor-based implementation are not applicable for RC implementation
  - Data flow schemes used in the microprocessor-based implementation in most cases are not suitable for RC implementation
- Only few scientific codes can be ported to an RC platform with relatively minor modifications
  - 90% of time is spent while executing 10% of the code
- Vast majority of the codes require significant restructuring in order to be 'portable', general problems are:
  - No well-defined compute kernel
  - Compute kernel is too large to fit on an FPGA
  - Compute kernel operates on a large dataset or is not called too many times
    - function call overhead becomes an issue

Lessons Learned

- Effective use of high-level programming languages/tools, such as MAP C/Carte (SRC-6) and Mitrion-SDK/Mitrion-C (RC100), to develop code for RC platform requires some limited hardware knowledge
  - Memory organization and limitations
    - Explicit data transfer and efficient data access
  - On-chip resources and limitations
  - RC architecture-specific programming techniques
    - Pipelining, streams, …
- Most significant code acceleration can be achieved when developing the code from scratch; the code developer then has the freedom to
  - structure the algorithm to take advantage of the RC platform organization and resources,
  - select most effective SW/HW code partitioning scheme, and
  - setup data formats and data flow graph that maps well into RC platform resources

Conclusions

- **Making HPRCs relatively easy for scientists is challenging**
  - **More work on Programming Models needed**
  - **More work on OS needed**
  - **More work on Tools**
- **The proven (*demonstrated*) promise (*only in some cases for now*) is too great to give up**
  - **> 38000+ X of speed up**
  - **> 3000+ X saving in power**
  - **> 90+ X saving in $$**
  - **> 1000+ X saving in size**

Publications

- El-Araby, Taher, El-Ghazawi, and LeMoigne. Remote Sensing and High-Performance Reconfigurable Computing (HPRC) Systems, Chapter 18 in High Performance Computing in Remote Sensing, CRC
- El-Ghazawi, El-Araby, Huang, Gaj, Kindratenko and Buell, "The Performance Promise of High-Performance Reconfigurable Computing", IEEE Computer (in press)
- Mohamed Abouellail, Esam El-Araby, Mohamed Taher, Tarek El-Ghazawi and Gregory B. Newby, "DNA and Protein Sequence Alignment with High Performance Recofigurable Systems", NASA/ESA Conference on Adaptive Hardware and Systems 2007(AHS2007), August 5-8, 2007, Scotland, UK
- Proshanta Saha, Tarek El-Ghazawi, "Automatic Software Hardware Co-Design for Reconfigurable Computing Systems", 17th International Conference on Field Programmable Logic and Applications (FPL 2007), 27-29 August 2007, Amsterdam, Netherlands
- E. El-Araby, I. Gonzalez, and T. El-Ghazawi, "Bringing High-Performance Reconfigurable Computing to Exact Computations", to appear in the proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL 2007), Amsterdam, Netherlands, 27-29 August 2007.
- Proshanta Saha and Tarek El-Ghazawi, A Methodology for Automating Co-Scheduling for Reconfigurable Computing Systems. Fifth ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'2007), Nice, May 2007.
- Proshanta Saha, Tarek El-Ghazawi, "Software/Hardware Co-Scheduling for Reconfigurable Computing Systems"; International Symposium on Field-Programmable Custom Computing Machines 2007 (FCCM 2007); 23-25 April 2007, Napa, CA
- Proshanta Saha, Tarek El-Ghazawi, "Applications of Heterogeneous Computing in Hardware/Software Co-scheduling ", International Conference on Computer Systems and Applications (AICCSA 2007), Amman, May 2007.
- Proshanta Saha, Tarek El-Ghazawi, "Software/Hardware Co-Scheduling for Reconfigurable Computing Systems", Proceeding of III Southern Conference on Programmable Logic (SPL 2007), February 26-28, 2007 - Mar del Plata, Argentina
- Miaoqing Huang, Tarek El-Ghazawi, Brian Larson, Kris Gaj : "Development of Block-cipher Library for Reconfigurable Computers", Proceeding of III Southern Conference on Programmable Logic (SPL 2007), February 26-28, 2007 - Mar del Plata, Argentina
- Esam El-Araby, Mohamed Taher, Mohamed Abouellail, Tarek El-Ghazawi, and Gregory B. Newby, "Comparative Analysis of High Level Programming for Reconfigurable Computers: Methodology and Empirical Study", Proceeding of III Southern Conference on Programmable Logic (SPL 2007), February 26-28, 2007 - Mar del Plata, Argentina

Publications

- **M. Taher and T. El-Ghazawi, " A Segmentation Model for Partial Run-Time Reconfiguration", IEEE International Conference on Field Programmable Logic and Applications (FPL06), Madrid, Spain, August 2006.**
- **M. Taher and T. El-Ghazawi, "Exploiting Processing Locality Through Paging Configurations in Multitasked Reconfigurable Systems", IEEE Reconfigurable Workshop (RAW2006), Proceedings of International Parallel and Distributed Processing Symposium, Rhodes Island, Greece, April 2006.**
- **E. El-Araby, M. Taher, T. El-Ghazawi, and J. Le Moigne, "Automatic Image Registration for Remote Sensing on Reconfigurable Computers", 2006 MAPLD International Conference, Washington, DC, September, 2006**
- **Tarek El-Ghazawi, Kris Gaj, Duncan Buell, Proshanta Saha, Esam El-Araby, Chang Shu, Miaoqing Huang, Mohamed Taher, and Alan Michalski, "Libraries of Hardware Macros for Reconfigurable Computers", 2006 MAPLD International Conference, Washington, DC, September, 2006**
- **Kris Gaj, Tarek El-Ghazawi, Dan Poznanovic, Hoang Le, Proshanta Saha, Steve Heistand, Chang Shu, Esam El-Araby, Miaoqing Huang, Deapesh Misra, and Paul Gage,"Design of parameterizable hardware macros for reconfigurable computers", 2006 MAPLD International Conference, Washington, DC, September, 2006**
- **E. El-Araby, M. Taher, T. El-Ghazawi, A. Youssif, R. Irish, and J. Le Moigne, "Performance Scalability of a Remote Sensing Application on High Performance Reconfigurable Platforms", NASA Earth-Sun System Technology Conference (ESTC 2006), Maryland, USA, June, 2006.**
- **E. El-Araby, T. El-Ghazawi and K. Gaj , A System-Level Design Methodology for Reconfigurable Computing Applications, IEEE Conference on Field Programmable Computing Technology (FPT 2005), Singapore, Dec 2005.**
- **E. El-Araby, M. Taher, T. El-Ghazawi and J. Le Moigne Prototyping Automatic Cloud Cover Assessment (ACCA) Algorithm for Remote Sensing On-Board Processing on a Reconfigurable Computer, IEEE Conference on Field Programmable Computing Technology (FPT 2005), Singapore, Dec 2005.**
- **J. Harkins, T. El-Ghazawi, E. El-Araby and M. Huang, Performance of Sorting Algorithms on a Reconfigurable Computer, IEEE Conference on Field Programmable Computing Technology (FPT 2005), Singapore, Dec 2005.**
- **E. El-Araby, M. Taher, K. Gaj, T. El-Ghazawi, D. Caliga, N. Alexandridis, "System-Level Parallelism and Concurrency Maximization in Reconfigurable Computing Applications", International Journal for Embedded Systems (IJES), vol. 2, no. 1/2, 2006, pp. 62-72.**
- **S. Kaewpijit, J. Le Moigne, and T. El-Ghazawi, "Automatic Reduction of Hyperspectral Imagery Using Wavelet Spectral Analysis," IEEE Transactions on Geosciences and Remote Sensing (TGARS), Vol. 41 No. 4, April 2003, pp 863-871.**
- **T. El-Ghazawi, K. Gaj, N. Alexandridis, F. Vroman, N. Nguyen, J. Radzikowski, P. Samipagdi, and S. Suboh, "A Performance Study of Job Management Systems," Concurrency and Computation: Practice and Experience, John Wiley & Sons, Ltd.**

Publications

- E. El-Araby, M. Taher, K. Gaj, T. El-Ghazawi, D. Caliga, N. Alexandridis, "System-Level Parallelism and Concurrency Maximization in Reconfigurable Computing Applications", International Journal for Embedded Systems (IJES), vol. 2, no. 1/2, 2006, pp. 62-72.
- S. Kaewpijit, J. Le Moigne, and T. El-Ghazawi, "Automatic Reduction of Hyperspectral Imagery Using Wavelet Spectral Analysis," IEEE Transactions on Geosciences and Remote Sensing (TGARS), Vol. 41 No. 4, April 2003, pp 863-871.
- T. El-Ghazawi, K. Gaj, N. Alexandridis, F. Vroman, N. Nguyen, J. Radzikowski, P. Samipagdi, and S. Suboh, "A Performance Study of Job Management Systems," Concurrency and Computation: Practice and Experience, John Wiley & Sons, Ltd.
- E. El-Araby, M. Taher, T. El-Ghazawi, A. Youssif, R. Irish, and J. Le Moigne, "Performance Scalability of a Remote Sensing Application on High Performance Reconfigurable Platforms", NASA Earth-Sun System Technology Conference (ESTC 2006), Maryland, USA, June, 2006.
- E. El-Araby, M. Taher, T. El-Ghazawi, J.Le Moigne, "Prototyping Automatic Cloud Cover Assessment (ACCA) Algorithm for Remote Sensing On-Board Processing on a Reconfigurable Computer,"Proc. IEEE 2005 Conference on Field Programmable Technology, FPT'05, Singapore, Dec. 11-14, 2005.
- J. Harkins, E. El-Araby, M. Huang, T. El-Ghazawi, "Performance of Sorting Algorithms on a Reconfigurable Computer," Proc. IEEE 2005 Conference on Field Programmable Technology, FPT'05, Singapore, Dec. 11-14, 2005.
- E. El-Araby, K. Gaj, T. El-Ghazawi, "A System Level Design Methodology for Reconfigurable Computing Applications," Proc. IEEE 2005 Conference on Field Programmable Technology, FPT'05, Singapore, Dec. 11-14, 2005.
- C. Shu, K. Gaj, T. El-Ghazawi , "Low Latency Elliptic Curve Cryptography Accelerators for NIST Curves on Binary Fields," Proc. IEEE 2005 Conference on Field Programmable Technology, FPT'05, Singapore, Dec. 11-14, 2005.
- **S. Bajracharya, D. Misra, K. Gaj, T. El-Ghazawi , "Reconfigurable Hardware Implementation of Mesh Routing in Number Field Sieve Factorization," Extended Abstract, Talk Special Purpose Hardware for Attacking Cryptographic Systems, SHARCS 2005, Paris, France, Feb. 24-25, 2005, pp. 71-81.**
- **E. El-Araby, T. El-Ghazawi, J.Le Moigne, and K. Gaj, "Wavelet Spectral Dimension Reduction of Hyperspectral Imagery on a Reconfigurable Computer," Proc. IEEE 2004 Conference on Field Programmable Technology, FPT 2004, Brisbane, Australia, Dec. 6-8, 2004, pp. 399-402.**

Publications

- E. Chitalwala, T. El-Ghazawi, K. Gaj, N. Alexandridis, D. Poznanovic, "Effective System and Performance Benchmarking for Reconfigurable Computers," Proc. IEEE 2004 Conference on Field Programmable Technology, FPT 2004, Brisbane, Australia, Dec. 6-8, 2004, pp. 453-456.
- S. Bajracharya, C. Shu, K. Gaj, T. El-Ghazawi, "Implementation of Elliptic Curve Cryptosystems over GF(2^n) in Optimal Normal Basis on a Reconfigurable Computer," 14th International Conference on Field Programmable Logic and Applications, FPL 2004, Antwerp, Belgium, Aug 30 - Sept 1, 2004, pp. 1001-1005..
- E. El-Araby, M. Taher, K. Gaj, T. El-Ghazawi, D. Caliga, N. Alexandridis "System-Level Parallelism and Throughput Optimizations in Designing Reconfigurable Computing Applications," Reconfigurable Architecture Workshop, RAW 2004, Santa Fe, USA, Apr 26-27, 2004.
- N. Nguyen, K. Gaj, D. Caliga, T. El-Ghazawi, "Implementation of Elliptic Curve Cryptosystems on a Reconfigurable Computer," Proc. IEEE International Conference on Field-Programmable Technology, FPT 2003, Tokyo, Japan, Dec. 2003, pp. 60-67.
- E. El-Araby, M. Taher, K. Gaj, D. Caliga, T. El-Ghazawi, N. Alexandridis, "Exploiting System-level Parallelism in the Application Development on a Reconfigurable Computer," Proc. IEEE International Conference on Field-Programmable Technology, FPT 2003, Tokyo, Japan, Dec. 2003, pp. 443-446.
- A. Michalski, K. Gaj, T. El-Ghazawi, "An Implementation Comparison of an IDEA Encryption Cryptosystem on Two General-Purpose Reconfigurable Computers," LNCS 2778, 13th International Conference on Field Programmable Logic and Applications, FPL 2003, Lisbon, Portugal, Sep. 2003, pp. 204-219.
- O. D. Fidanci, D. Poznanovic, K. Gaj, T. El-Ghazawi, and N. Alexandridis, "Performance and Overhead in a Hybrid Reconfigurable Computer," Reconfigurable Architecture Workshop, RAW 2003, Nice, France, Apr. 2003.
- T. El-Ghazawi and F. Cantonnet, "UPC Performance and Potential: A NPB Experimental Study," Supercomputing'02, IEEE CS, Baltimore, Nov. 16-22, 2002.
- K. Gaj, T. El-Ghazawi, F. Vroman, N. Nguyen, J. R. Radzikowski, P. Samipagdi, and S. A. Suboh, "Performance Evaluation of Selected Job Management Systems," Proceedings of IEEE International Parallel and Distributed Processing Symposium (PMEO-PDS'02), Fort Lauderdale, Florida, Apr. 15-19, 2002.