Algorithm Acceleration with Reconfigurable Hardware

Maya Gokhale, maya@lanl.gov

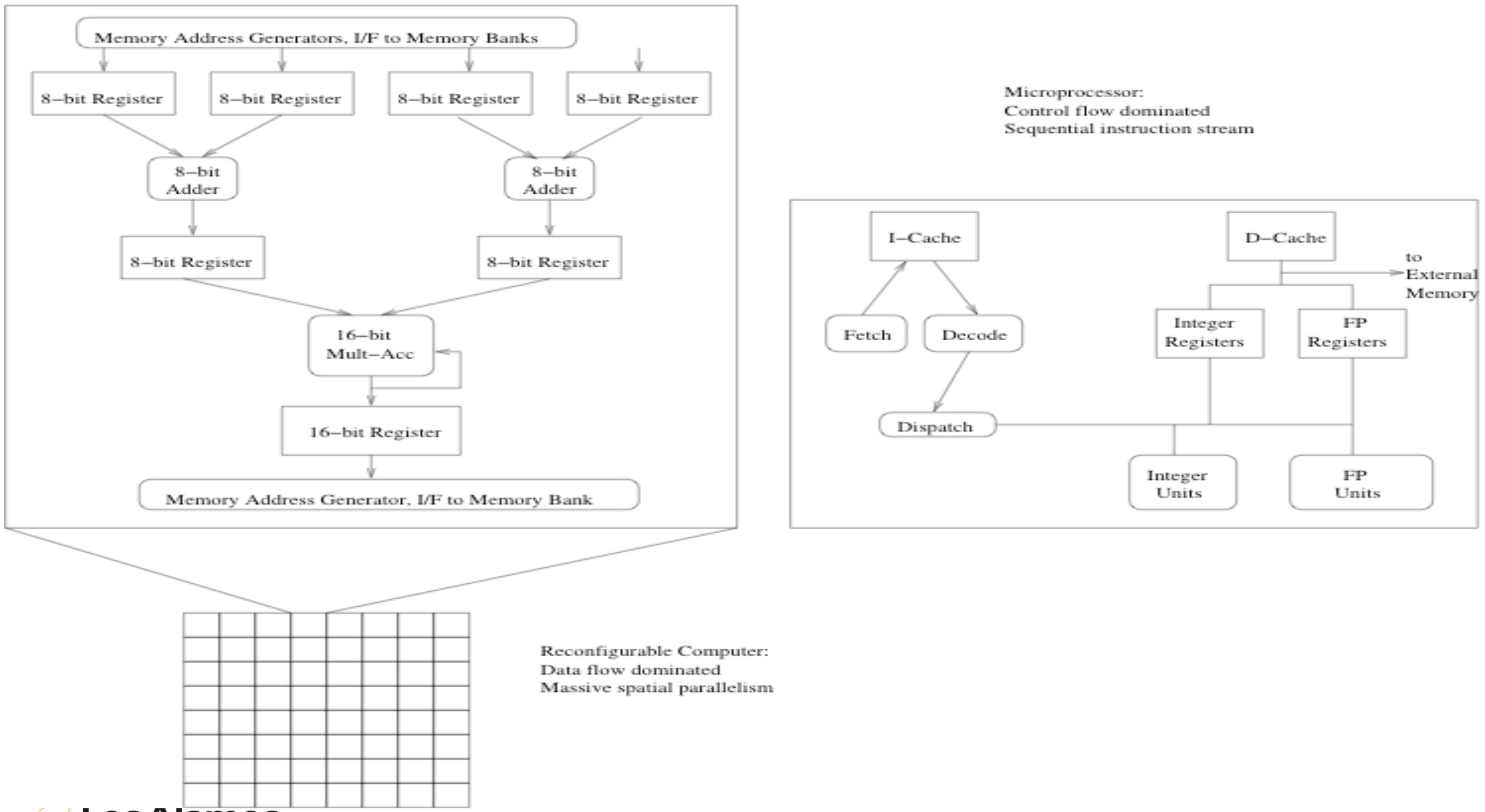
Questions we will discuss

- What is reconfigurable hardware?
- How is reconfigurable hardware incorporated into high performance systems?
- What applications or application classes can benefit from reconfigurable hardware?
- How do you write code for reconfigurable hardware?

Much of this material is taken from the book:

Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays, by **Gokhale**, Maya B., **Graham**, Paul S., ISBN: 0-387-26105-2

Reconfigurable Hardware: Example

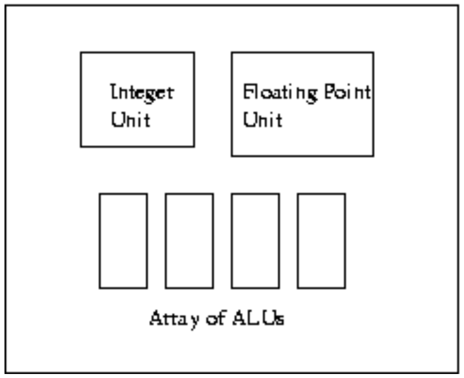


It’s all about parallelism

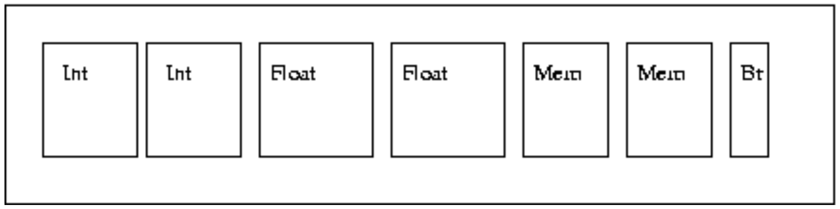
- more activities at the same time => higher performance

Low Level Parallelism

- Co-processor parallelism
  - Within a single instruction stream
  - Customized "instruction" is performed by co-processor, eg. Altivec, MMX, SSE, vector instructions
  - Integrated into the instruction set of the processor
  - Shares state with processor

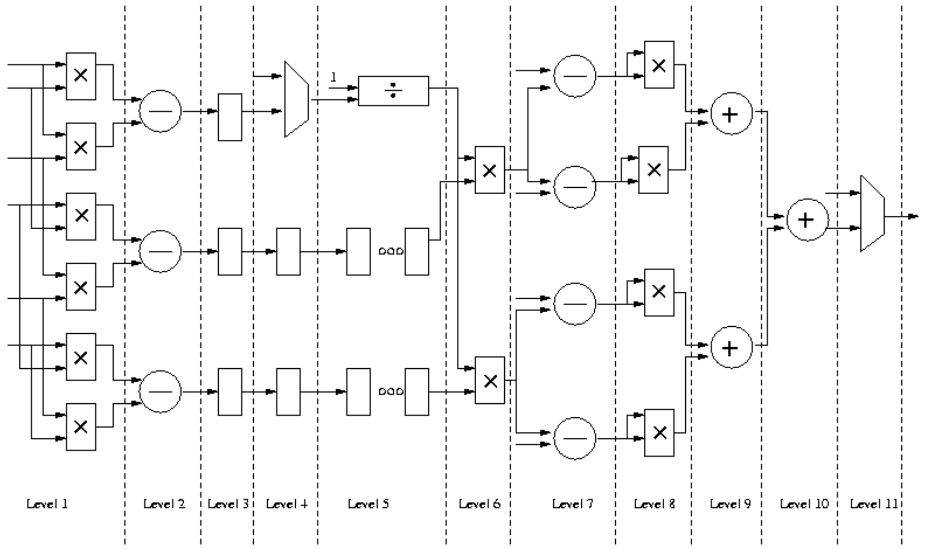


- Instruction level parallelism
  - VLIW: Multiple operations bundled into a single instruction
  - Superscalar: multiple instructions in progress at the same time



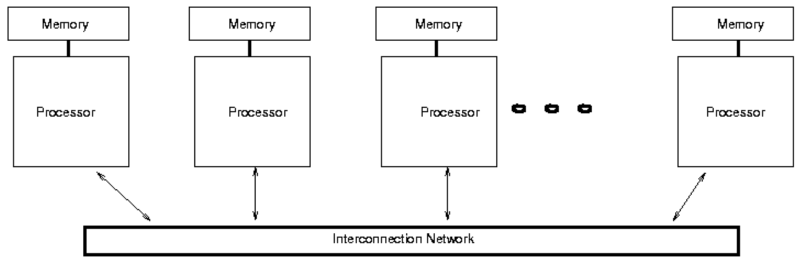
Low Level Parallelism: RCC View

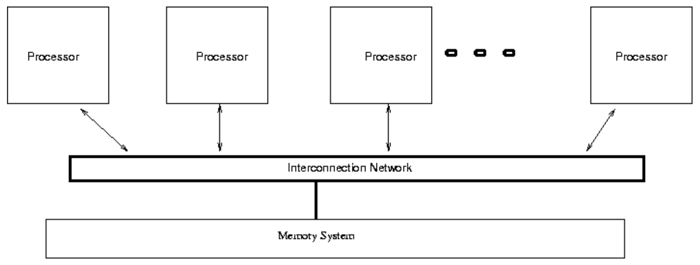
- Build customized instructions in programmable hardware
- Flexibility
  - Arbitrary number, kind of operations
  - Many alternative data sources and sinks
    - Interconnection network
    - I/O bus to microprocessor
    - Serial I/O, A/D
    - Other FPGAs
    - On-board memory
    - On-chip memory
- Performance
  - Customized pipelines
    - Vector chaining
  - Example
    - 11 stage floating point pipeline

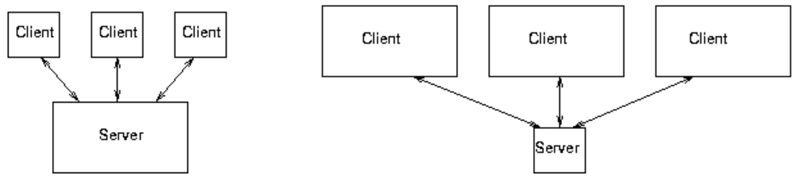


High Level Parallelism

- Peer-to-peer
  - Process level
    - Separate address space for each process
    - Communication among processes to share state
      - High latency messaging, eg. MPI
        Buffered
        Asynchronous
      - Low latency streaming, eg. Streams-C
        FIFO or Valid bit
        Valid bit
        Semi-synchronous
      - Tightly synchronized streaming
        Delays are pre-computed and built into algorithm
  - Thread level
    - Shared address space for the threads
    - Communication through shared memory or messaging
    - Signaling mechanisms
      - Critical sections, mutual exclusion
      - Barriers
- Client/Server
  - Client can request service from server
  - Client can request work from server (taskbag)

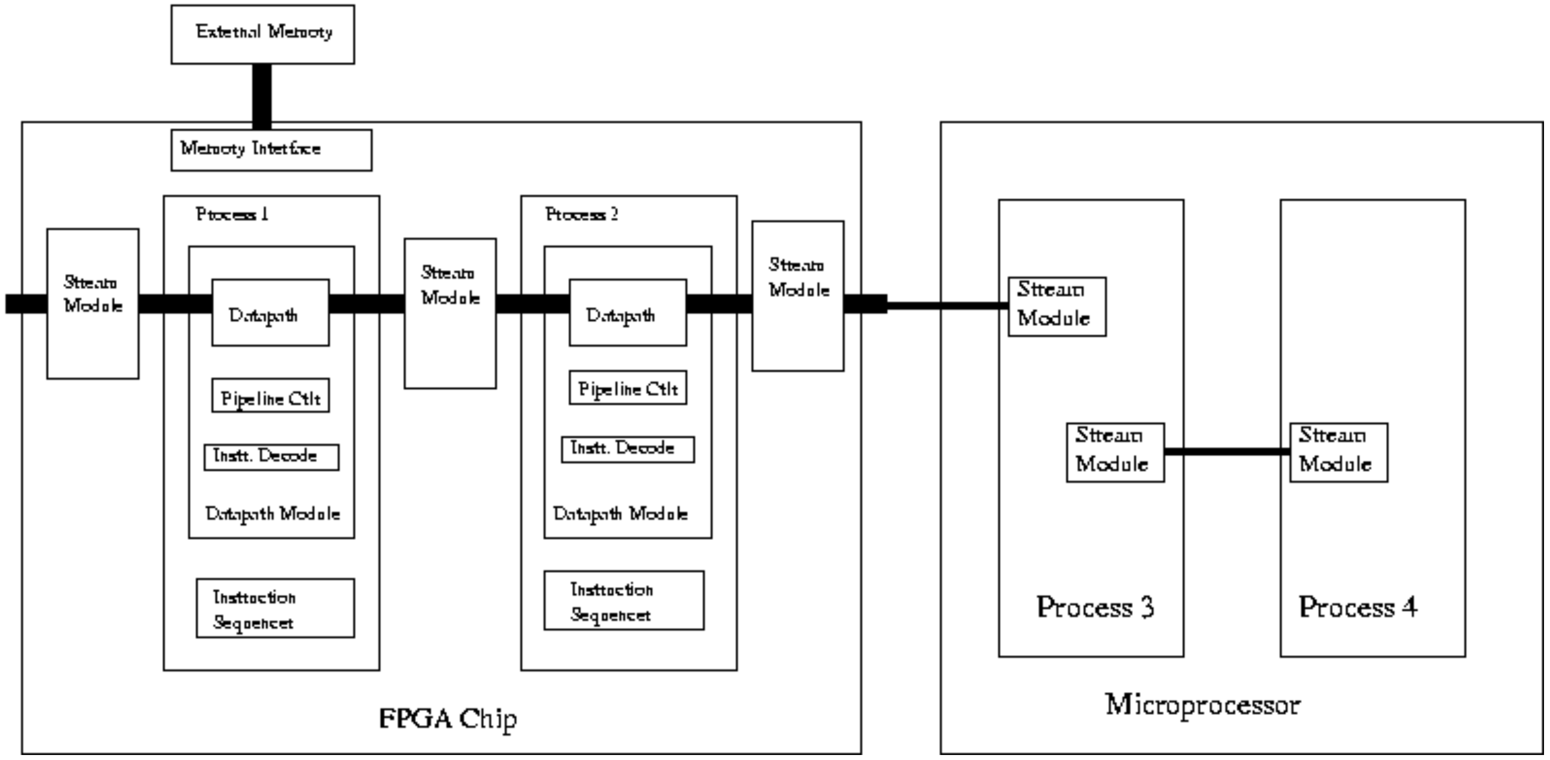




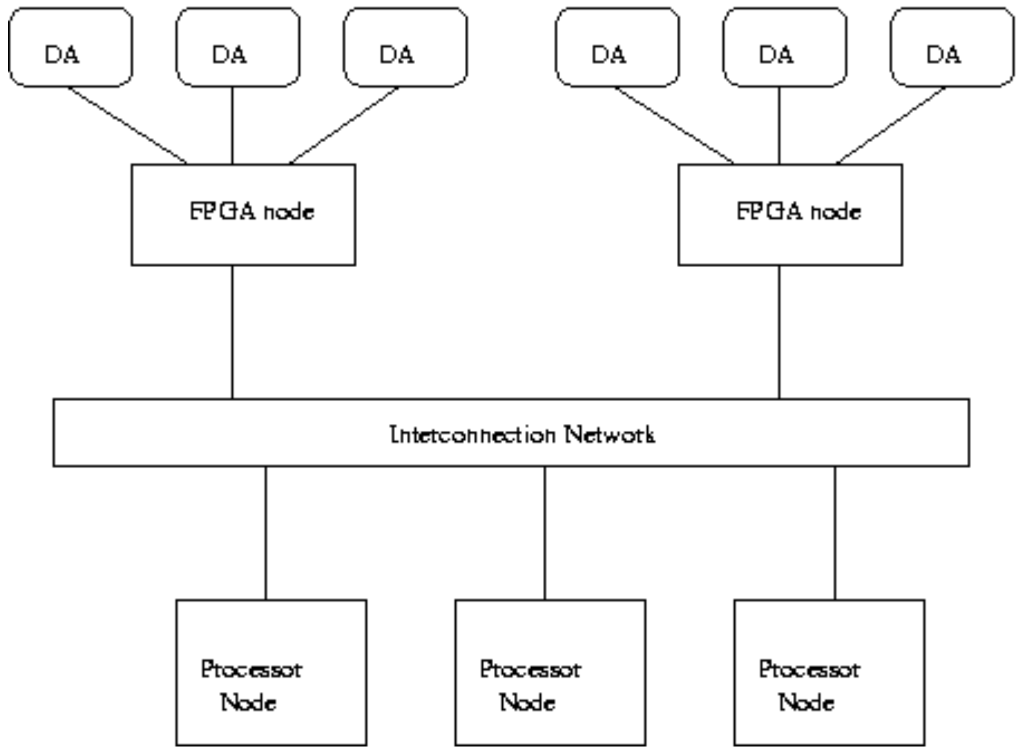


Peer-to-Peer Parallelism: RCC View

- Peer-to-Peer
  - hardware processes communicate with software processes
  - MPI not generally available
  - Most complete implementation is Streams-C/Impulse-C model
    - Low latency, high bandwidth streaming model



Client-Server Parallelism: RCC View

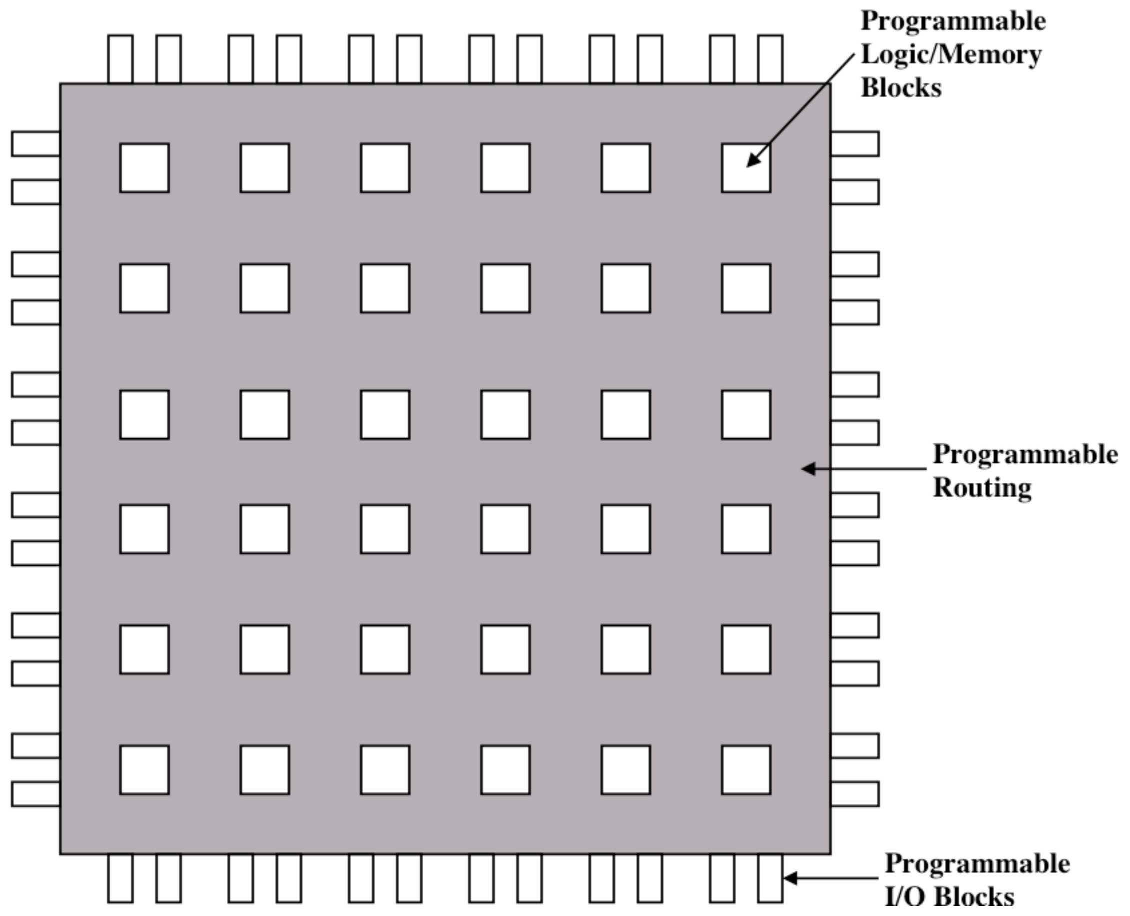


- Client/Server
  - FPGA performs data/compute task as requested by processor
  - Data Acquisition
    - FPGA collects high bandwidth, real-time data
    - FPGA performs first level processing
    - FPGA forwards to shared memory and processors over interconnection network

FPGA Hardware

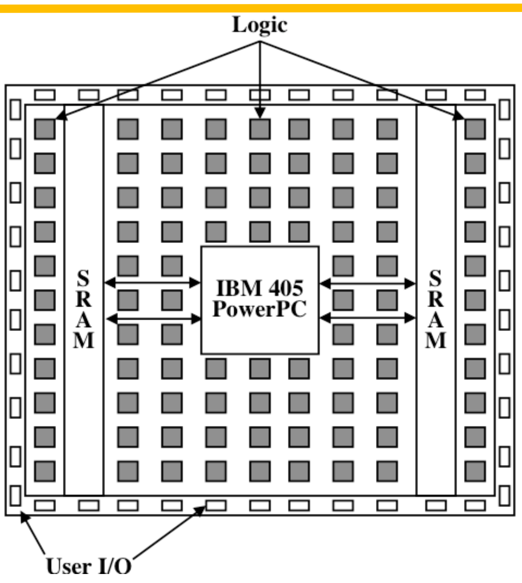
Overview of the architecture and building blocks of Field Programmable Gate Arrays

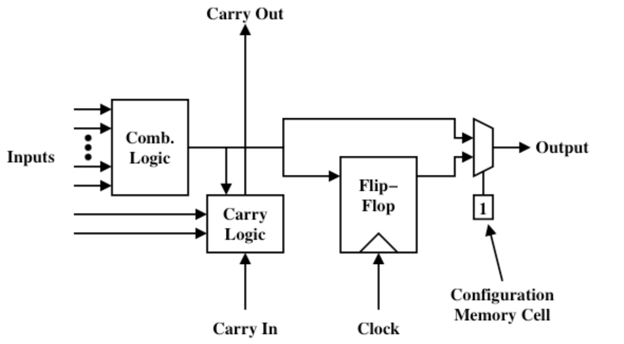
What is an FPGA

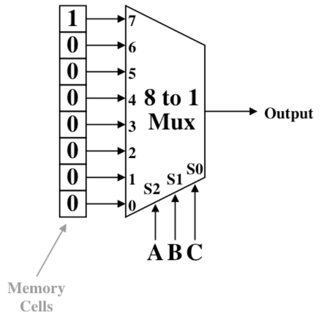


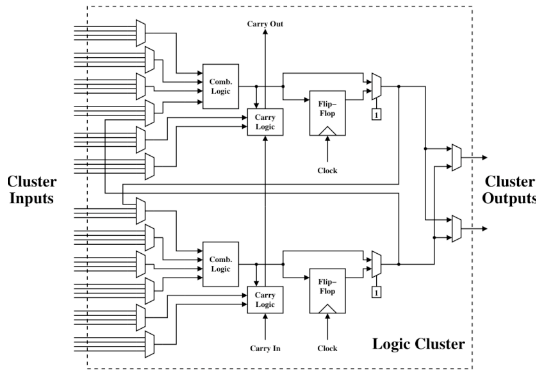
•Function Units are synthesized from Logic/Memory Blocks.
•Function Units are interconnected with programmable routing wires.
•Communication to outside uses I/O blocks

A Closer Look



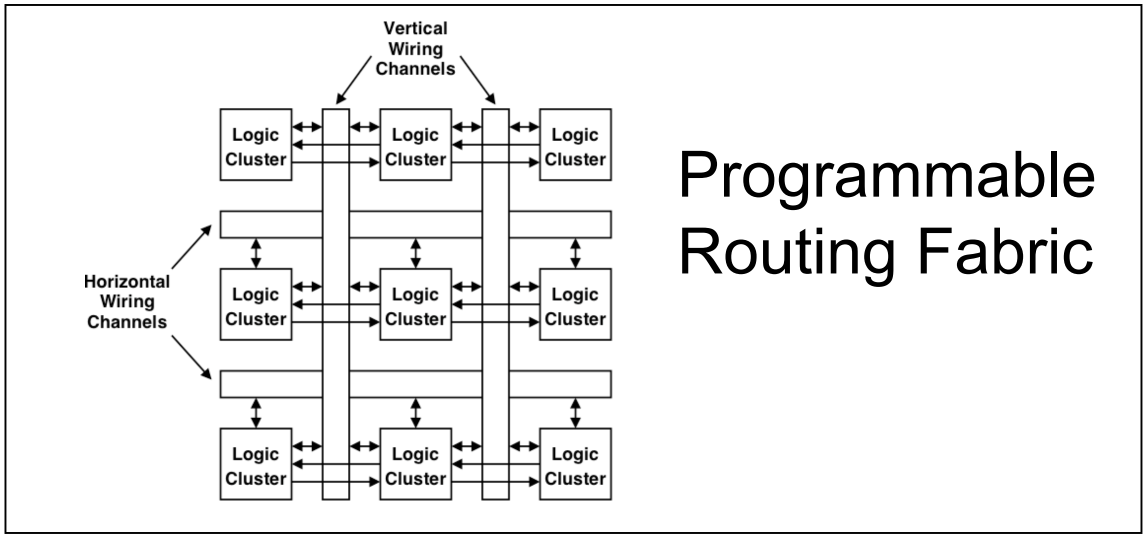




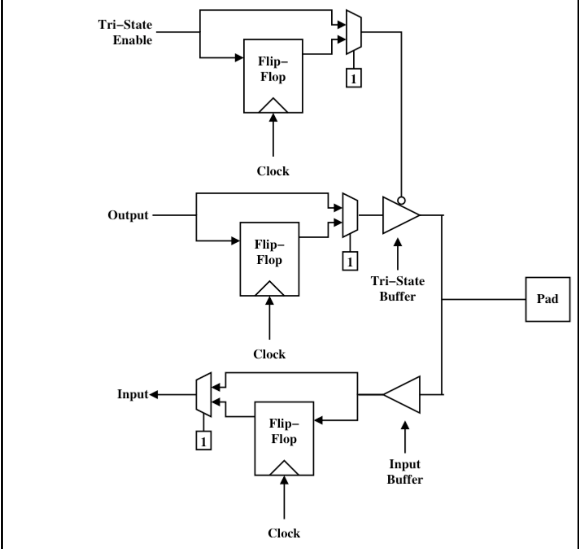


An **FPGA slice** usually consists of two flip-flops, two LUTs and some associated mux, carry, and control logic.

Routing and I/O

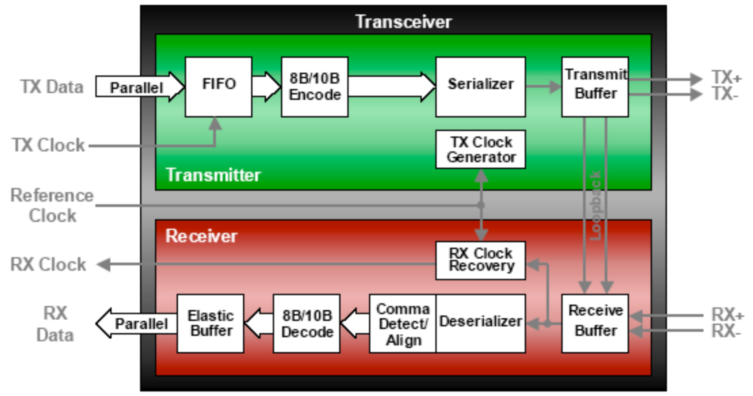


Programmable Routing Fabric



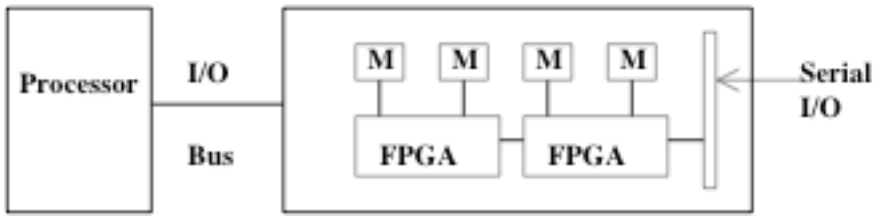
IO Block

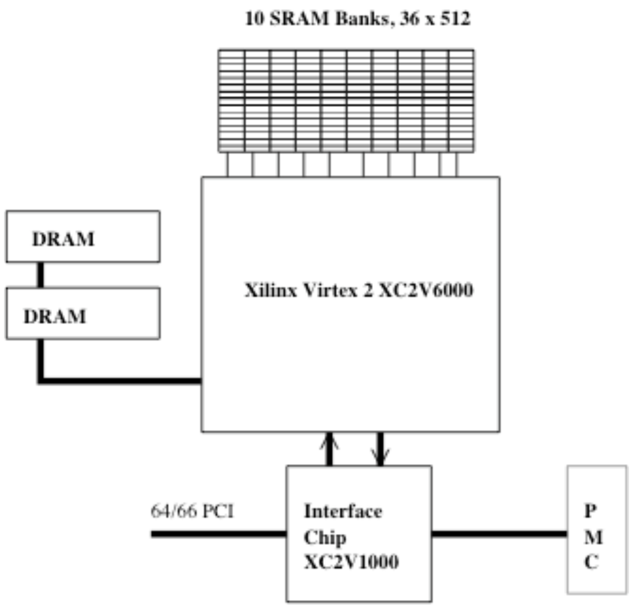
Multi-Gb Serial IO



Putting FPGAs into systems

Generic Reconfigurable Computing System

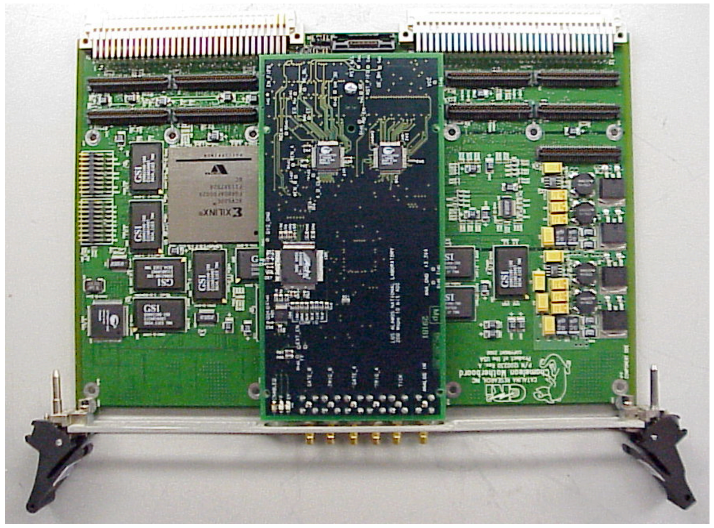




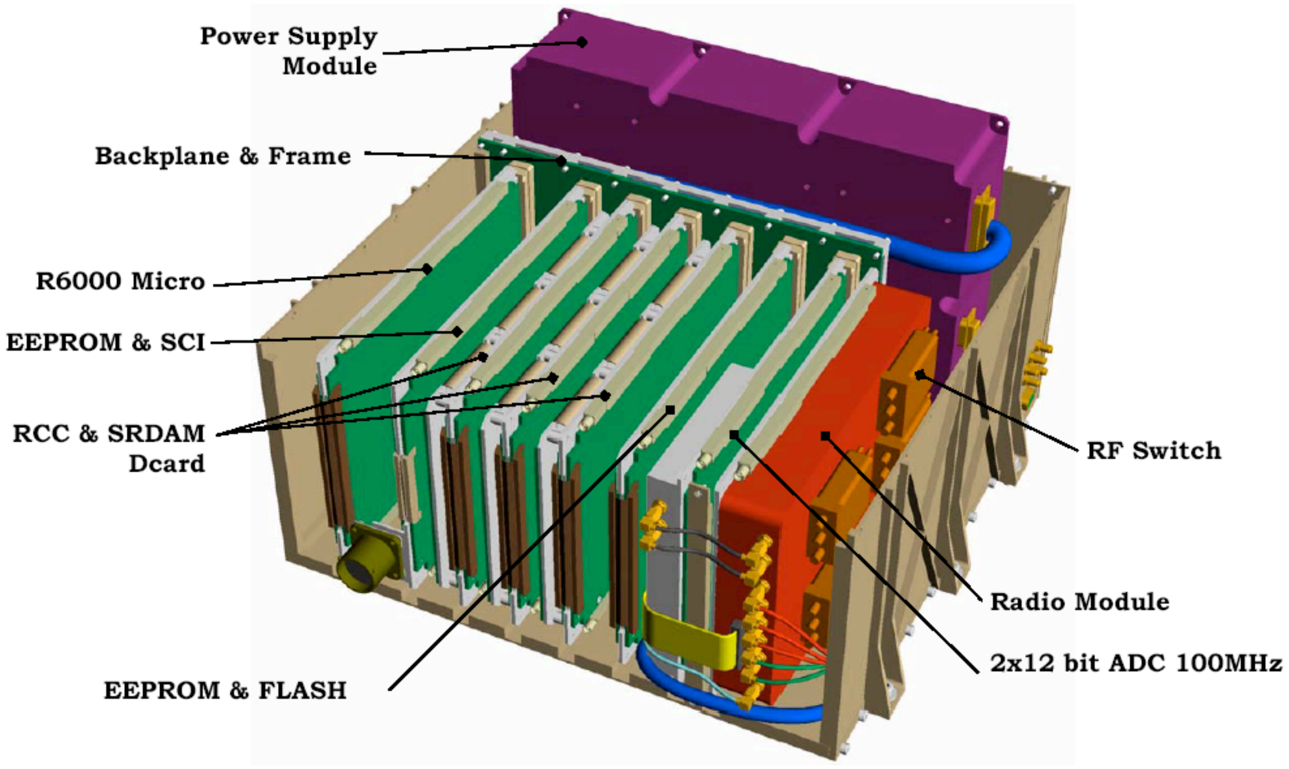
- Field Programmable Gate Array (FPGA) is the processor.
- Each processor has dedicated (or shared) memory
- Collections of FPGA plus external memory form the computing system.
- RC system attaches to "host" workstation via an I/O bus.

Embedded Systems - Chameleon Board

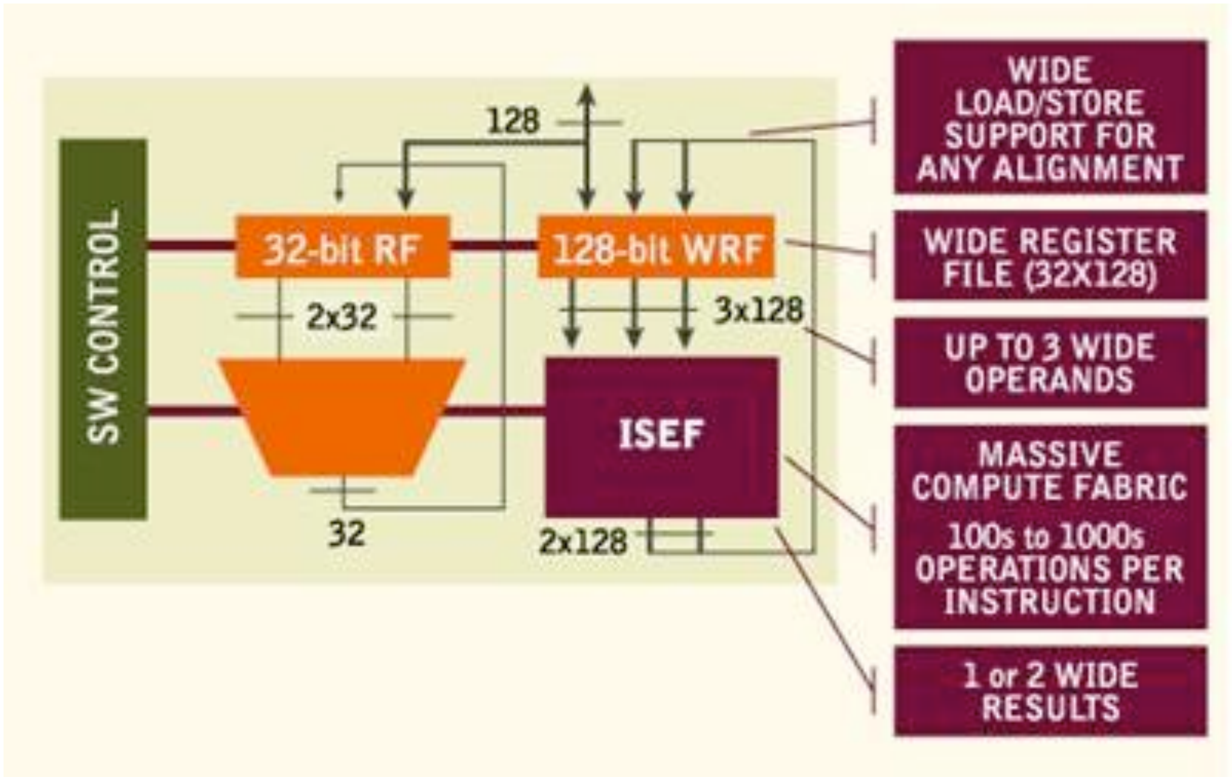
The heart of the Chameleon VME board consists of four Xilinx® Virtex - E 1000 FPGAs, providing over four million reconfigurable system gates that the user can apply to the application. Connected to FPGAs A, B, and C are independent synchronous SRAM blocks. Each bank is independent and can be used to provide large circular buffers. The FPGAs on the Chameleon VME link together in a ring with dedicated paths: A to B, B to C, and A to C. All three FPGAs also link back to D. These FPGA busses can be clocked at rates up to 133MHz.The Chameleon VME has three high-speed I/O daughtercard sites compatible with the industry's PCI Mezzanine Card (PMC) standard. By enhancing PMC, the effort to tailor the Chameleon VME to specific I/O requirements reduce to driver integration instead of a full custom I/O daughtercard development task.On the Chameleon VME board, FPGAs A, B, and C each have a dedicated connection to an I/O daughtercard. For PMC-based I/O applications, these links are separate 64-bit 66 MHz PCI busses. CRI provides a 'ready-to-go' PCI core. The daughtercard tied to FPGA C also contains additional connectivity to the VME64x P2 user defined pins, allowing connections to such back plane interfaces as RACE or Dual RACE++.



FPGA Board in Software Radio

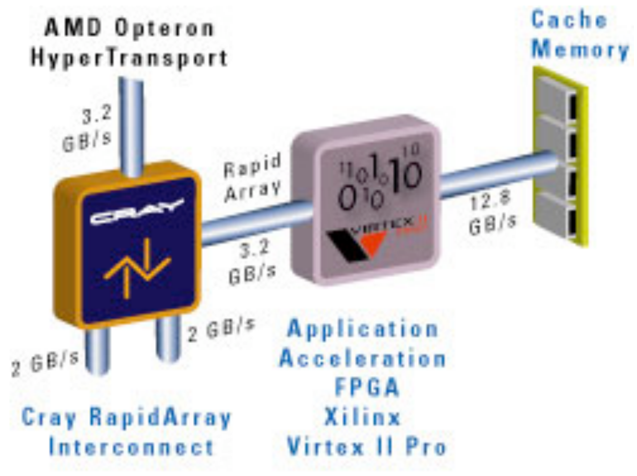


Stretch: Tensilica Processor + FGPA

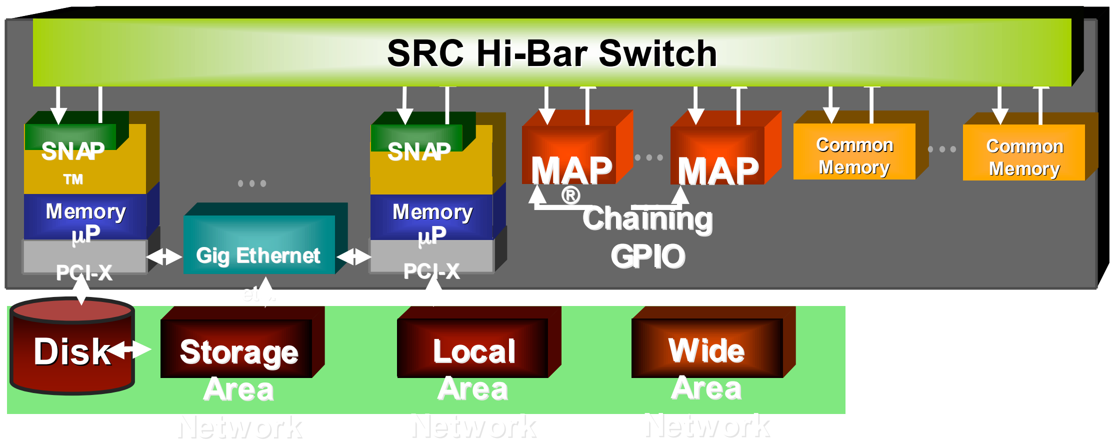


- Configurable processor
- Closely integrated fabric interface

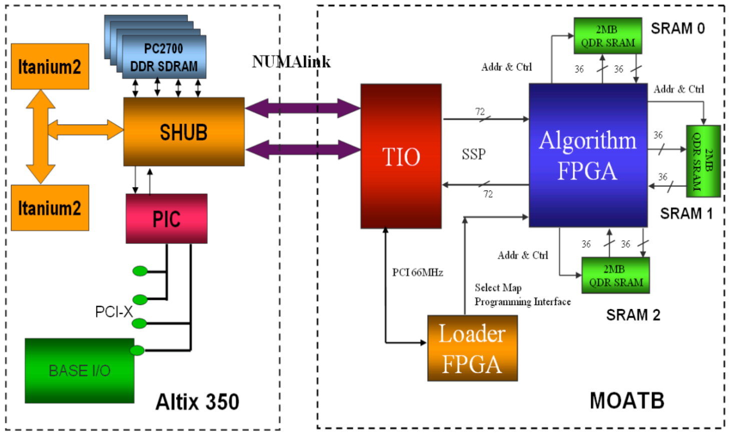
Reconfigurable Supercomputers: microprocessor + FPGA on high BW network


AMD Opteron HyperTransport
3.2 GB/s
Rapid Array
3.2 GB/s
12.8 GB/s
Cache Memory
2 GB/s
2 GB/s
Application Acceleration FPGA Xilinx Virtex II Pro
Cray RapidArray Interconnect

Cray XD1



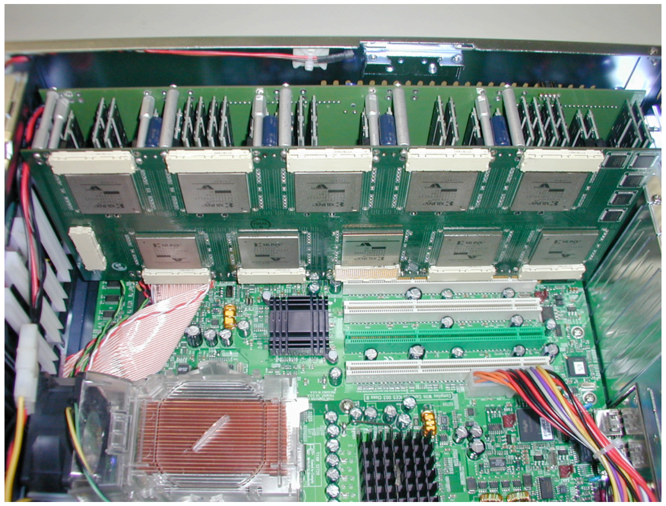
SRC MAP



SGI RASC

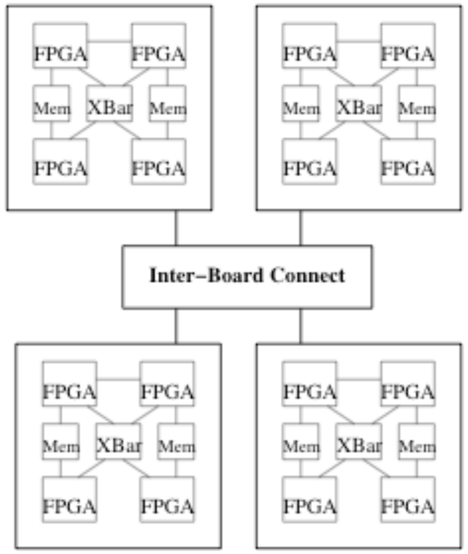
(El-Raby MAPLD2005)

FPGA Arrays



Starbridge Systems
(Storaasli, 2003)

DINI Group
Logic Emulator



BEE, BEE2

- Large collection of FPGAs
- High bandwidth connectivity among FPGAs
- Varying amounts of external memory
- Often used for emulation of large circuits

Architecture and Systems

- Chen Chang, John Wawrzynek, Robert W. Brodersen, The Design And Application of BEE2, A High-End Reconfigurable Computing System
- Heather Quinn, et. al., Terrestrial-based Radiation Upsets: A Cautionary Tale
- Prasanna Sundararajan, Xilinx, Future Architectures and Tools

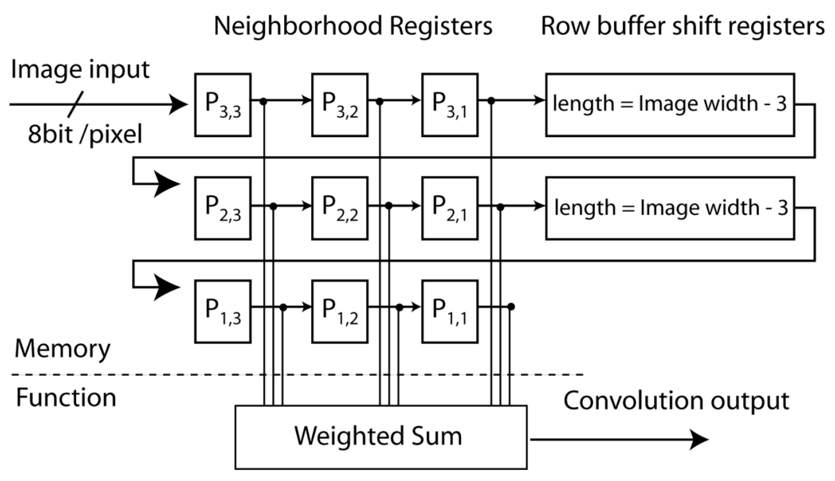
Applications

- Data Stream processing
  - hardware block on FPGA acquires sample from A/D or memory buffer
  - data packet flows directly through processing pipeline
  - processed data streamed to microprocessor or storage device
- Kernels of compute-intensive pipelines
  - large floating point expression evaluated on FPGA
  - input from microprocessor
  - result back to microprocessor for further processing
- Library functions
  - FFT
  - BLAS

Application Design Principles

- Maximize parallelism
  - multiple customized function units
  - pipelining within function unit
  - customized interconnect among function units
    - systolic flow
- Maximize memory bandwidth
  - application-controlled memory hierarchy
  - on-chip
    - flipflops in CLB configured to be part of a register
    - Embedded on-chip RAM blocks, configurable in width/depth
  - off-chip
    - multiple SRAM banks
    - DRAM

Data Stream Processing



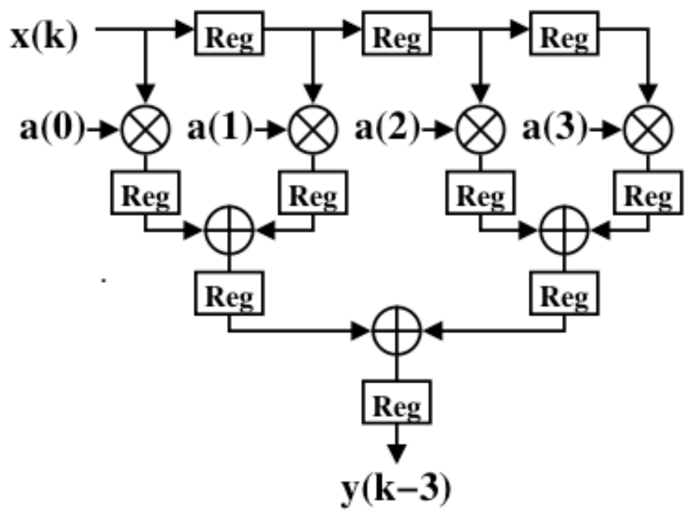
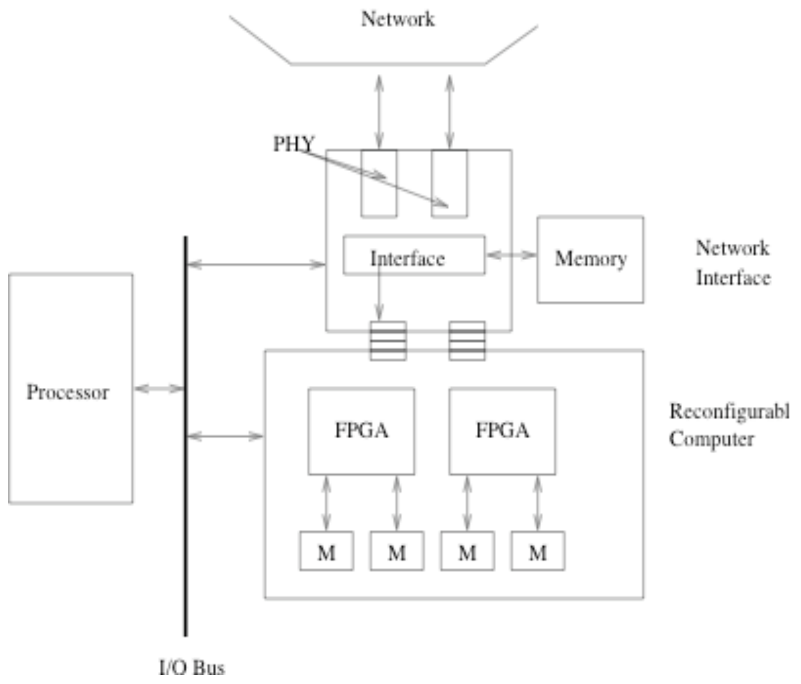


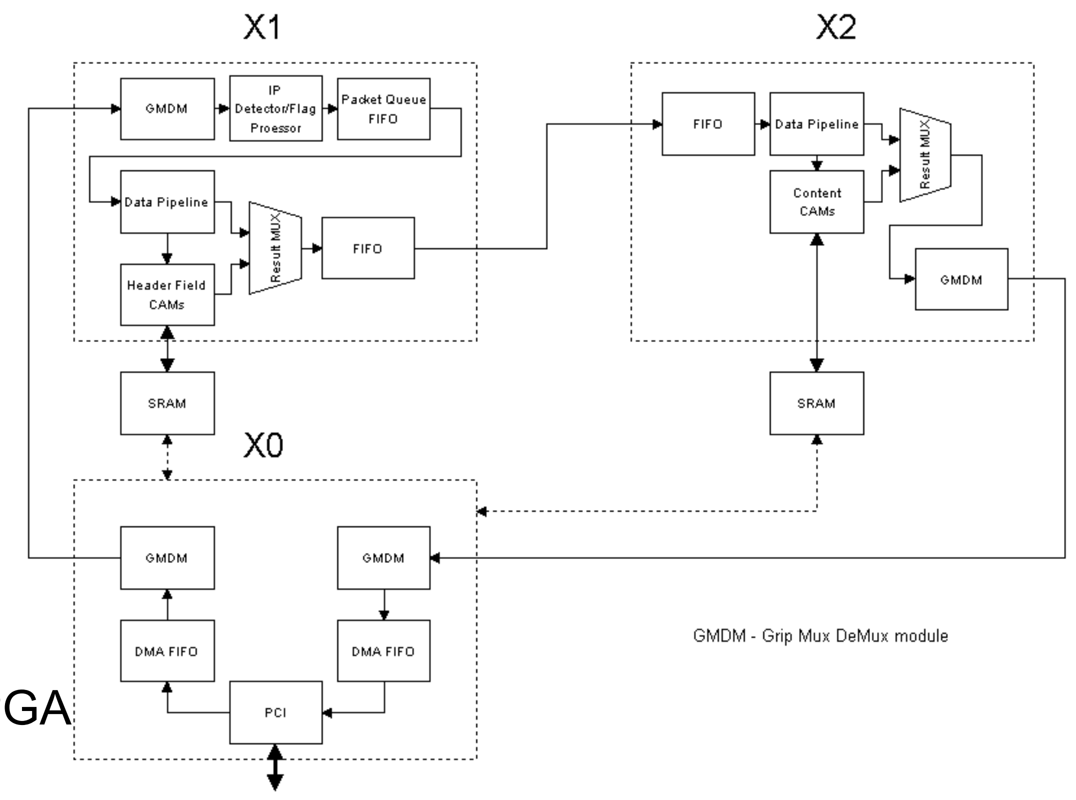
Image Processing
Neighborhood Operations

Signal Processing
FIR Filter

•use on-chip memory (BRAM, registers)
•exploit maximum spatial parallelism
•exploit pipelining

Network Packet Processing



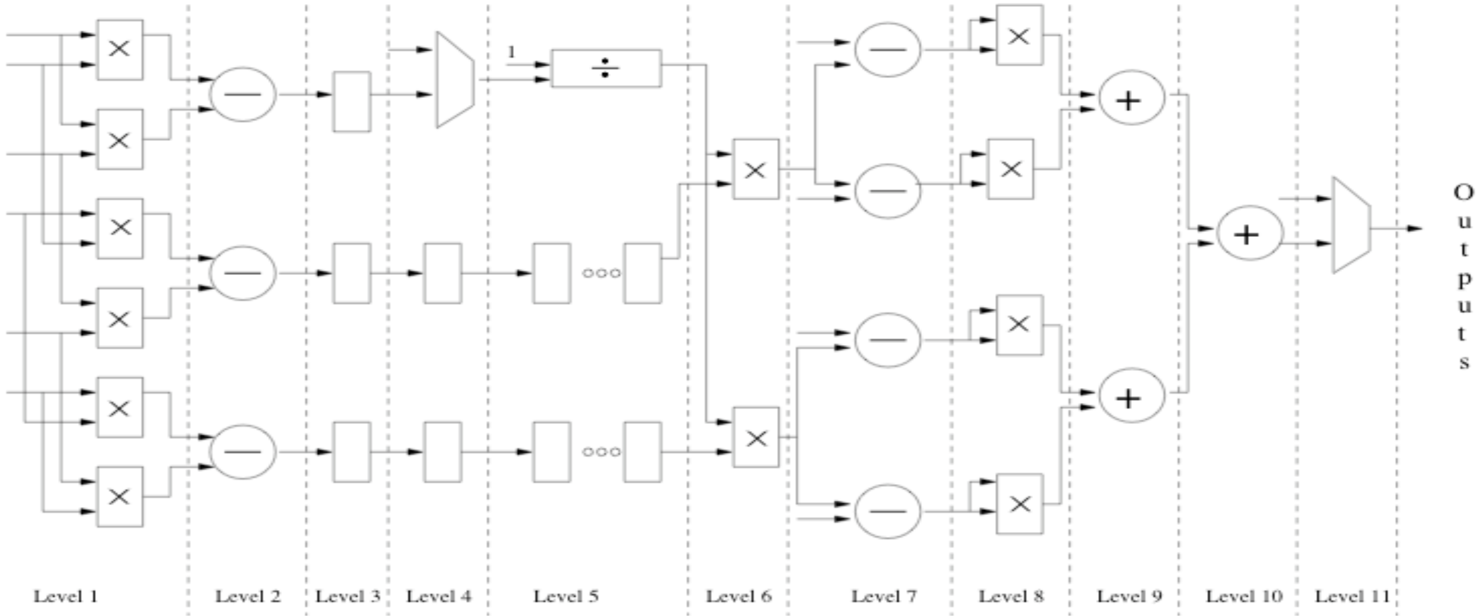


- •Network packets flow from NI to FPGA
- •FPGA circuits can match headers
- •FPGA circuits can match content
- •TCP-level packet assembly is difficult

Kernel Processing

```
for each surface in the m–surface polygon                                   (a)
  for each of N photons emitted from this surface                           (b)
    Emit a photon with random characteristics from this surface
    while the photon is not absorbed, transmitted or lost                   (c)
      for each side in the polygon                                          (d)
        if the current side is not the emitted side
          Check if the photon intersected with this side
        end if
      end for
      Randomly determine if the photon is absorbed, transmitted, reflected or lost
    end while
  end for
end for
```

- FPGA circuit for innermost loop
- FP operators are pipelined
- Expression eval is pipelined



Libraries

- FFT “IP” modules have been implemented that outperform older ASICs
  - good performance as FFT is in data flow of signal processing pipeline
- Star-P project - FPGA backend to accelerate Matlab library routines
  - performance depends on data communications costs, data re-organization costs, Amdahl’s law

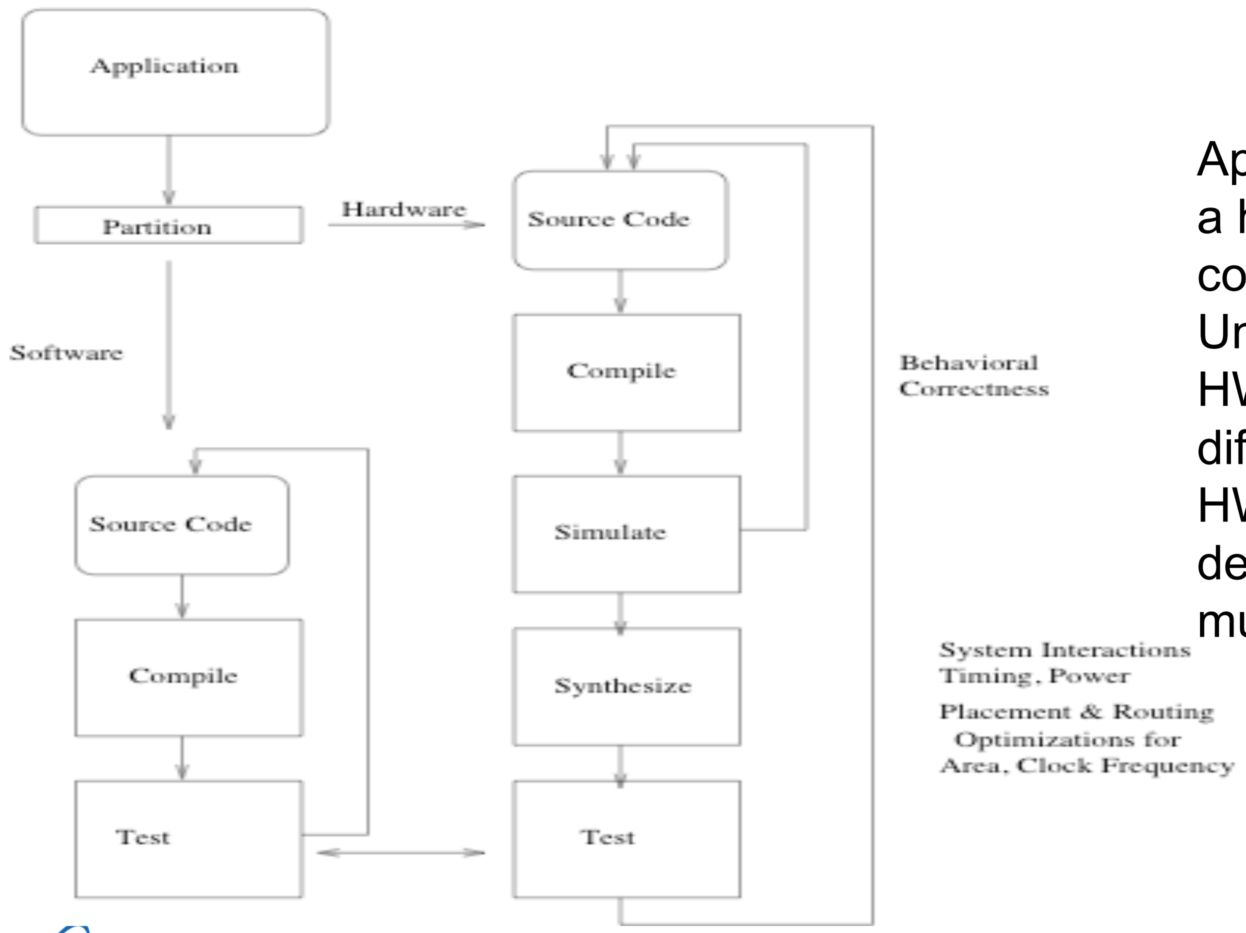
Applications Presentations

- Reid Porter, et. al.. A Reconfigurable Computing Framework for Multi-scale Cellular Image Processing
- Daniel G. Chavarra, Miranda and David Chassin. A Hardware-Accelerated Steady-State Power Flow Solver
- Chuan He, Guan Qinand and Wei Zhao, High-order Finite Difference Seismic Modeling on Reconfigurable Computing Platform
- Zachary K. Baker and Viktor K. Prasanna, Hardware Accelerated Apriori Algorithm for Data Mining

Design Tools and Compiling for FPGAs

- Select functions to run on FPGA
- Translate from algorithm to circuit
- Interface software and hardware

Design Cycle



Application has
a hardware and software
component.
Unlike SPMD model,
HW and SW perform
different functions.
HW and SW are compiled,
debugged differently, yet
must work together.

Levels of Description

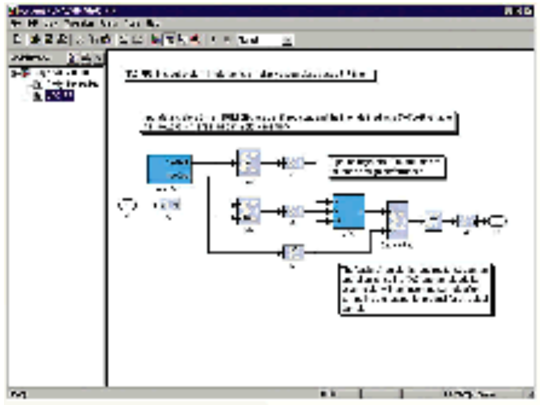
- Algorithmic
  - True software level of description
  - Function calls, pointers, …
- Behavioral
  - Describe behavior of a circuit
  - Timing behavior is synthesized
  - Alternative modules automatically chosen
  - C = (A + A) * (B+C)
- RTL
  - Registers, interconnection between registers is defined by programmer
  - Timing behavior is defined by programmer
  - Control structure is defined by programmer using state machine
  - Synthesizer builds control logic for register transfers
  - Seq(Par(T1 = B+C; T2 = A+A;) C = T1*T2)
- Structural
  - Arithmetic or control modules explicitly instantiated
  - Control behavior explicitly specified
  - M1: adder(B, C, T1, clk, enable); M2: adder (a, a, T2,…); M3: mult(T1, T2, C,…)

Algorithmic Languages

- SRC Computer C/Fortran
  - Tailored to SRC reconfigurable system
  - Common language for hardware and software
  - Manual partitioning between sw/hw
  - Automatic ILP extraction
  - Manual memory hierarchy management
  - Some capabilities for pipelining
  - unrolling?
- Impulse C (Impulse Accelerated Technologies), based on Streams-C
  - Targeted to system on a programmable chip (embedded processor)
    - Multiple board support packages
  - Common language for hardware and software
  - Manual partitioning between sw/hw
  - Automatic ILP extraction
  - Manual memory hierarchy management
  - Automatic pipelining
  - Unrolling?
- Catapult C (Mentor Graphics)
  - Targets FPGA/ASIC chips
  - Allows timing annotations to optimize design
- Forge (Xilinx)
  - Java compiler targets Xilinx FPGAs
  - Manual partitioning
  - Automatic ILP extraction
  - Pipelining
  - Unrolling?
  - Manual memory hierarchy management
- Accelchip
  - Matlab subset target gate level
  - Manual partitioning
  - Automatic ILP extraction
  - No loop level pipelining
  - Unrolling via directives
  - Manual memory hierarchy management
  - Provide hardware DSP libraries

High Level Graphical Languages

- Xilinx System Generator
  - Targets Xilinx chip
  - Matlab/Simulink interface
  - Signal processing library
  - Manual control over timing
  - Manual assembly of operators
  - Manual pipeline synchronization



- Starbridge Viva
  - Targets Starbridge system
  - Ported to other boards
  - Polymorphic modules
  - Manual control over timing through choice of operators
  - Manual assembly of operators
  - Manual pipeline synchronization
  - Manual partitioning

Behavioral Languages - Hardware Description

- Behavioral Language characteristics
  - Designer controls cycle-by-cycle behavior of parallel processes Compiler generates state machine to sequence computation
  - Manual ILP
  - Manual pipelining
  - Manual unrolling
- Celoxica Handel-C
  - Proprietary C variant
  - Compiler generates gate-level description
  - Simulation/debug tools provided by Celoxica
- System C
  - Embedded systems modeling language
  - Synthesis compiler offered by several vendors (Synopsys, Celoxica)
- VHDL/Verilog
  - Industry standard hardware description languages (HDL)
  - Many choices of synthesizers
  - Many choices of simulation/debug tools
  - Same language is used for behavioral, Register Transfer Level (RTL), and structural.

Celoxica Handel-C

- Based on Communicating Sequential Processes model
  - Independent parallel processes
  - Processes communicate over synchronous channels
    - Sender and receiver must arrive at synchronization point at same time
  - “par” construct to specify parallel computation blocks within a process
  - Well-defined timing model
    - Each statement takes a single clock cycle

Embedding HDL in General Purpose Language

- Programmatic generation
  - Parameterized design trivial
  - Built-in language features
    - Looping
    - Complex data structures
    - File I/O
    - Recursion
  - Designed for interactive module generation, design, and test.

```
for (int I=0;I<64;I++)
  if (someFunCall(I))
    q[I] = regp(d[I]);
  else
    q[I] = reg(d[I]);
```

- Debug Environments
  - Open circuit structure API.
  - Extensible using inheritance to tackle concepts like power awareness and fault tolerance.
- Unified debug environment
  - Simulation ⇔Hardware execution.
    - Same environment
    - Same testbenches
  - Application built on top of debug environment.

+ *files of constants*
+ *branch/bound optimal searches*
+ *regular expression parsing*

SystemC

- C++ Class Library
- System-level modeling of
  - Software algorithm
  - Hardware architecture
  - Interfaces between communicating entities
- Allows design space exploration
  - Executable specification
- System is represented as a set of interacting processes
  - Cycle-based approach
- Supported data types include bit, bit vector, fixed point,standard scalar types, four-state logic with resolved logic signals
- Synthesis compilers for SystemC now available

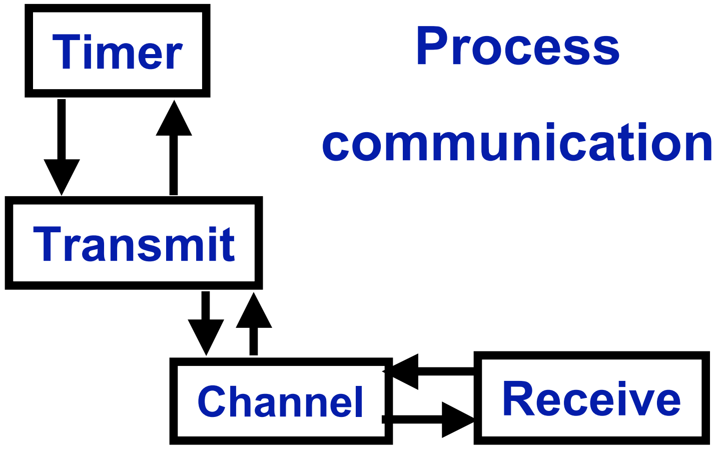
- Module is the basis for encapsulation, packaging
- Module contains processes
- Three types of processes
  - Method process
    - activation is triggered by event on input signal (sensitivity list)
  - Thread
    - Co-routine behavior
    - Can suspend (wait statement) and resume
  - Clocked Thread
    - Useful for more detailed specification, yielding better synthesis results
    - Activated on one edge of one clock
    - Basis for state machine synthesis (with wait statement)

System C Example: Module

```
SC_MODULE(transmit) {
sc_in<bool> clock; // input ports
sc_in<packet_type> tpackin;
sc_in<bool> timeout;
sc_out<packet_type> tpackout; // output ports
sc_inout<bool> start_timer;
int buffer;
void send_data();
 int get_data_from_app();
// Constructor
SC_METHOD(send_data);
sensitive << timeout; sensitive_pos << clock;
… buffer = get_data_from_app();
```

* Examples from SystemC User Manual

```
void transmit::send_data() {
if (timeout) { …
}
… else {
…
}
```



Process communication

VHDL and Verilog

VHDL

- Sponsored by DoD
- Based on Ada, the DoD-mandated software programming language
  - Modularity constructs
  - User-defined data types
  - Entity/architecture
    - Allows different implementation to conform to an interface
- IEEE Std 1076-1987
- IEEE Std 1076-1993
- IEEE Std 1076.1-1999
  - Analog and Mixed Signal extensions

Verilog

- Created in 1983-84 by Gateway Design Automation
- Based on C, modula
  - Concise, compact code
- IEEE Std-1364-1995
- IEEE Std-1364-2001
  - Adds features such as multi-dimensional arrays, generate
- IEEE Std-1364-AMS in progress

Compiling for Reconfigurable Systems

- Algorithmic/Behavioral
  - Control/Dataflow analysis
  - Dependency analysis and loop scheduling
  - High level pipelining
  - Target-independent expression-level optimizations
  - Predication
  - Operation selection
  - Memory access scheduling
  - Low level scheduling and pipelining
  - State machine synthesis
- RTL
  - Resource binding (function units, registers)
- Structural
  - Gate-level code generation
- Mapping
  - Gates are packed into (virtual) logic blocks of target FPGA
- Placement
  - Virtual logic blocks are mapped to physical logic blocks
- Routing
  - Connections between physical logic blocks assigned to routing resource on chip

Steps in Hardware Synthesis

- Predication
  - If (a) then X else Y → [(a) X; (~a) Y]
  - Eliminates control flow
  - Useful transformation for convention microprocessors also
- Operation selection
  - Size, eg. 4/8/12/14/16 … 64 bit
  - Data type, eg. [unsigned] int or float
  - Implementation, eg. Combinational, pipelined, bit serial, …
- Scheduling
  - Transform the control/data flow graph into a timed execution flow
  - Decide on operations that occur at every clock cycle
    - Function units
    - Memory access
    - Registers
  - Specify finite state machine to sequence through the sets of operations at each clock cycle
  - Synthesize pipelines for loops that can be pipelined

Compiler Optimizations

- Universal optimizations
  - Constant propagation
  - Common sub-expression elimination
  - Induction variables and strength reduction
  - Dead code elimination
  - Code motion for loop invariants
  - Resource sharing
- Parallelization optimization
  - Fine grained parallelization
  - Loop unrolling
  - Loop pipelining
- Specialized optimizations
  - Bit width analysis
  - Fixed vs. floating point representations
  - Power reduction

Universal Optimizations

- Constant propagation
  - C = 1; … C+D →
  - C = 1; … 1 + D
- Common subexpression elimination
- A = C+D; B = E*(C+D) →
- A = C+D; B = E*A
- Strength reduction
- A*2 → A+A
- Dead code elimination
- A = 5 … A = 6 →
- A = 6
- Loop invariants
- for (I…) A = 10 →
- A = 10; for (I…)
- Resource sharing
- A = (B + C) * (D + E)
- two adders or one?

- Optimizations are applicable to hardware and software
- Applicable to HDLs and HLLs
- Can be applied at different levels
  - Peephole
  - Basic block
  - subroutine
  - program

Parallelization Optimizations

Fine grained parallelization

$$\begin{array}{l} \text{T1: A = 10} \\ \text{T2: B = 5} \end{array} \longrightarrow \text{T1: A = 10; B = 5}$$

Loop unrolling

For (I=1 to 4) A[I] = B[I]

$\downarrow$

```
T1:A[1]=B[1]
T1:A[2]=B[2]
T1:A[3]=B[3]
T1:A[4]=B[4]
```

Loop pipelining

For (I=1 to N) A[I] = f(B[I])

$\downarrow$

Steady state pipeline:

```
Ti:fetch A[I]
Ti:temp=f(A[I-1])
Ti:B[I-2]=temp
```

Logic Synthesis and Technology Mapping

- Circuit design: express gates as boolean equation
  - Minimize boolean equations
  - Synthesize two-level, multi-level circuits
  - Minimize and synthesize sequential logic (multiple clock cycles)
    - Optimize and synthesize finite state machines

- Technology mapping: convert logic circuit into equivalent one that uses modules from a particular library
- Match subject graph (circuit to be mapped) onto pattern graphs in the technology library
- NP-hard problem

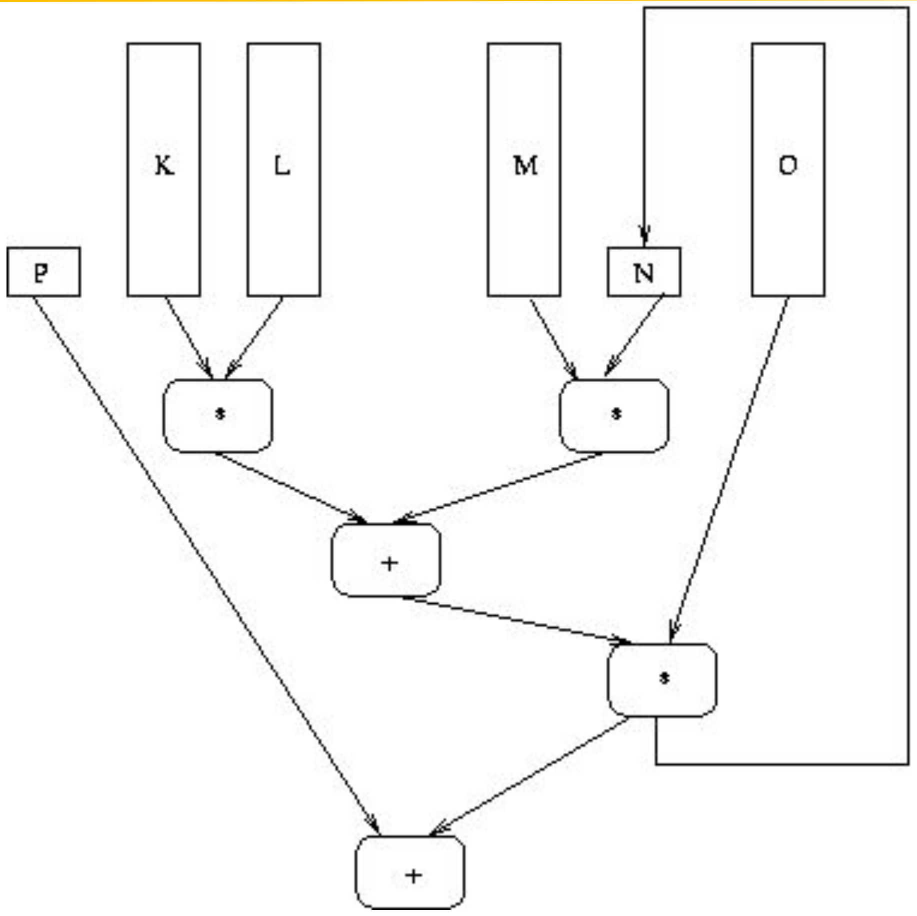
Placement and Routing

- Placement
  - Assign modules selected in mapping phase to physical modules on FPGA
  - Generates a layout of the design onto logic blocks on FPGA
- Routing
  - Assign paths between modules to physical routing resources on the chip
  - Simulated annealing optimization

Example

```
n=0
p=0
for i
    n = (K[i]*L[i]+M[i]*n)*O[i]
    p = n+p
```

- Many choices for instruction level parallelism
  - multiplies/adds in parallel or sequential?
    - area/speed tradeoff
    - affects loop-level pipelining
  - Number of clock cycles to compute n & p
    - affects clock speed



Pipeline Loop Alternatives

- Pipelined: One memory for all arrays
- Initiate a new loop iteration every 4 clock cycles.
- 8 stage pipeline

Stage 0: Update array pointers; issue read of M

Stage 1: Increment i; issue read of K; save M in register

Stage 2: temp1 = M*n; issue read of L; save K in register

Stage 3: issue read of O; save L in register

Stage 4: temp2 = K*L; save O in register

Stage 5: temp3 = temp1 + temp2

Stage 6: n = temp3*O

Stage 7: p = p+n

- Pipelined: Four memories
- Initiate a new loop iteration every 2 clock cycles
- 6 stage pipeline

Stage 0: Issue reads of K, M, L, O

Stage 1: Increment i; save M, L, O in registers

Stage 2: temp1 = K*L; temp2 = M*n

Stage 3:temp3 = temp1 + temp2

Stage 4: n = temp3*O

Stage 5: p = p+n

- Combinational: Four memories
- Initiate a new loop iteration every clock cycle
- 1 stage pipeline

Stage 0: Perform all multiplies and add, store results in registers

For XC2V2000-6

153/10752 slices

5/56 multipliers

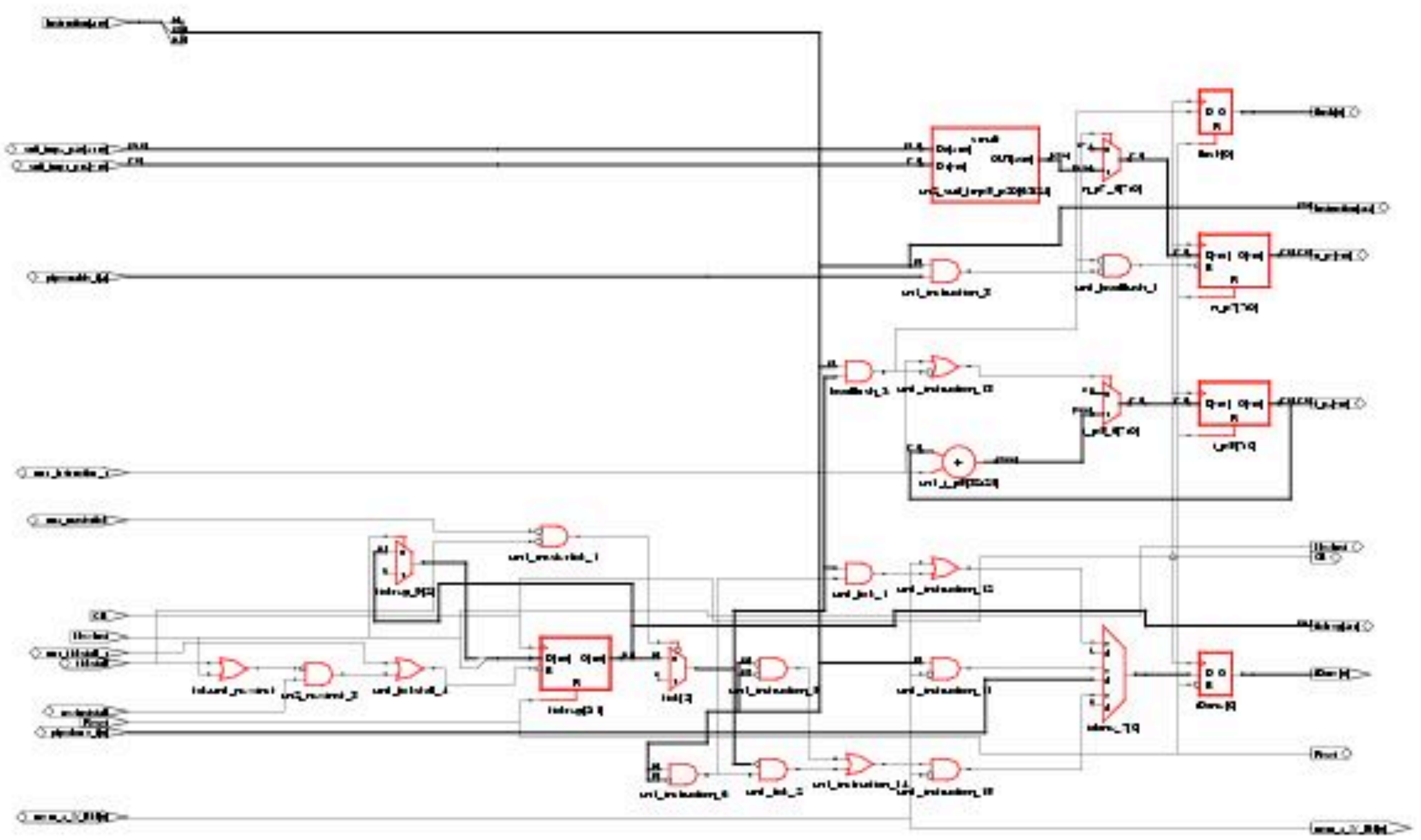
66 MHz

131/10752 slices

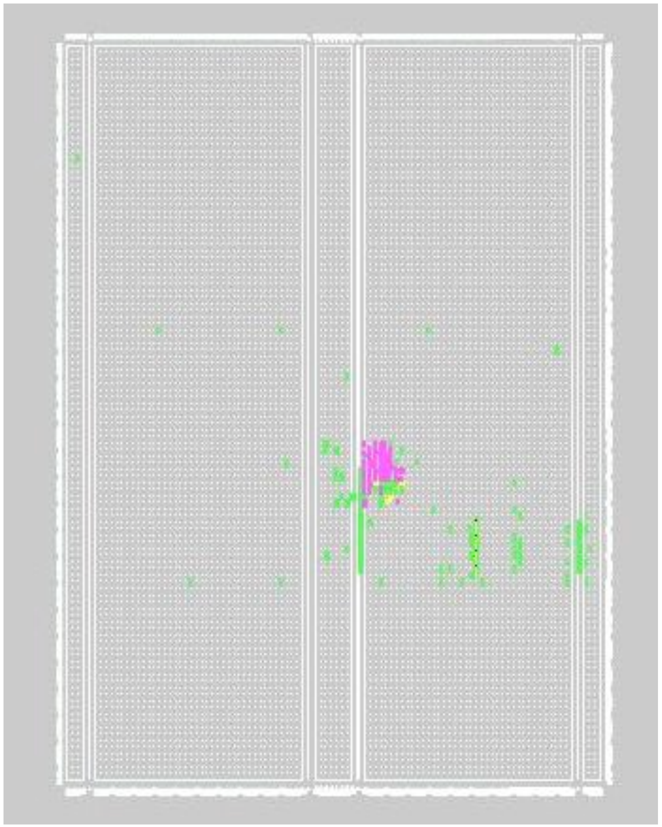
3/56 multipliers

219 MHz

Schematic View from Synthesis



Schematic View from Place&Route



Pipeline controller

Datapath: adders/multipliers/registers

Sequence controller

```
Device speed data version:  PRODUCTION 1.118 2004-03-12.
Device utilization summary:
Number of External IOBs            442 out of 624    70%
Number of LOCed External IOBs      0 out of 442      0%
Number of MULT18X18s               3 out of 56       5%
Number of SLICEs                   131 out of 10752  1%
Number of BUFGMUXs                 1 out of 16       6%
The AVERAGE CONNECTION DELAY for this design is:        1.217
The MAXIMUM PIN DELAY IS:                              4.558
 The AVERAGE CONNECTION DELAY on the 10 WORST NETS is:4.005
```

Language/Compiler Summary

- Mapping algorithms onto RCCs is a parallel processing problem
- Languages for reconfigurable computers range from high level C/Java to schematic to hardware description languages
- Compilers face a daunting task - extract ILP, pipeline loops, unroll, trade-off area/speed
- Tool chain has many components unfamiliar to software developers

Design Tools and Libraries

- Justin L. Tripp et. al. , Trident: An FPGA Compiler Framework for Scientific Computing
- Keith D. Underwood and K. Scott Hemmert, Implications of FPGAs for Floating-Point HPC Systems