Shakti Processor Program

G S Madhusudan (Principal Scientist)

Neel Gala (PhD candidate)

RISE Group, CSE Dept.

IIT Madras

Overview

- Goals of the program – Consolidate various Processor development initiatives and create a viable processor eco-system
- Research in design of next generation Processors, Interconnects and Storage Systems
  - Experimental Processors
  - SHAKTI family of processors
    - Low power, Mobile, Server and HPC variants
    - RapidIO based interconnect for the SHAKTI family
    - RapidIO based SSD storage to complement SHAKTI server class processors
    - OS, Compilers, BSP and related SW sub-systems
    - Reference designs and boards

Program Goals

- Processor Research
  - SHAKTI family, Adaptive System fabric/RapidIO Interconnect , Lightstor Storage
  - Experimental architectures like Tagged ISA, Belt systems and Single Address space systems
  - Low power design
- Bluespec/Chisel based workflow
  - ADD workflow
  - Formal Techniques for system design workflow
  - Low power design techniques and tool development

Program Goals

- OS development for SHAKTI
  - Linux, Android and L4 BSP development
  - Compiler development
- OS research for Experimental processors
  - Single address space OS
  - Secure OS for Tagged CPU arch.
  - Languages for secure CPU/OS

Processor Variants

Overview

- The SHAKTI Processor systems encompasses the following HW/system components
  - A family of cores ranging from uC class cores to Server class cores
  - A collection of interconnect fabrics, from single core AHB based bus fabrics to 100+ core multi-stage fabrics
  - Serial RapidIO based interconnect for external I/O and chip to chip CC interconnects

Overview

- SSD based storage system that uses SRIO to provide multi-million IOP and exabyte scalable storage
- Hybrid Memory Cube based memory controller to provide a scalable fabric based memory architecture
- Security components to provide trusted computing support
- Fault Tolerant SoC components (cores, fabrics) for safety critical applications

Shakti Processor Family

- C-Class
  - 32-bit Micro-controller
  - RV32I
- I-Class variant
  - 64-bit GP controller
  - Single-threaded, Quad Issue, etc.
- S-Class
  - Targeting Server Applications
  - Multi-stage fabric
  - Hybrid Memory Cube.
- M-Class
  - Multi-core version of I Class high performance and embedded applications.
  - Targeting complex SoC systems
- H-Class
  - 64+ Cores
  - SIMD/VPU Support
- T-Class
  - Tagged ISA for security.

Secure variants

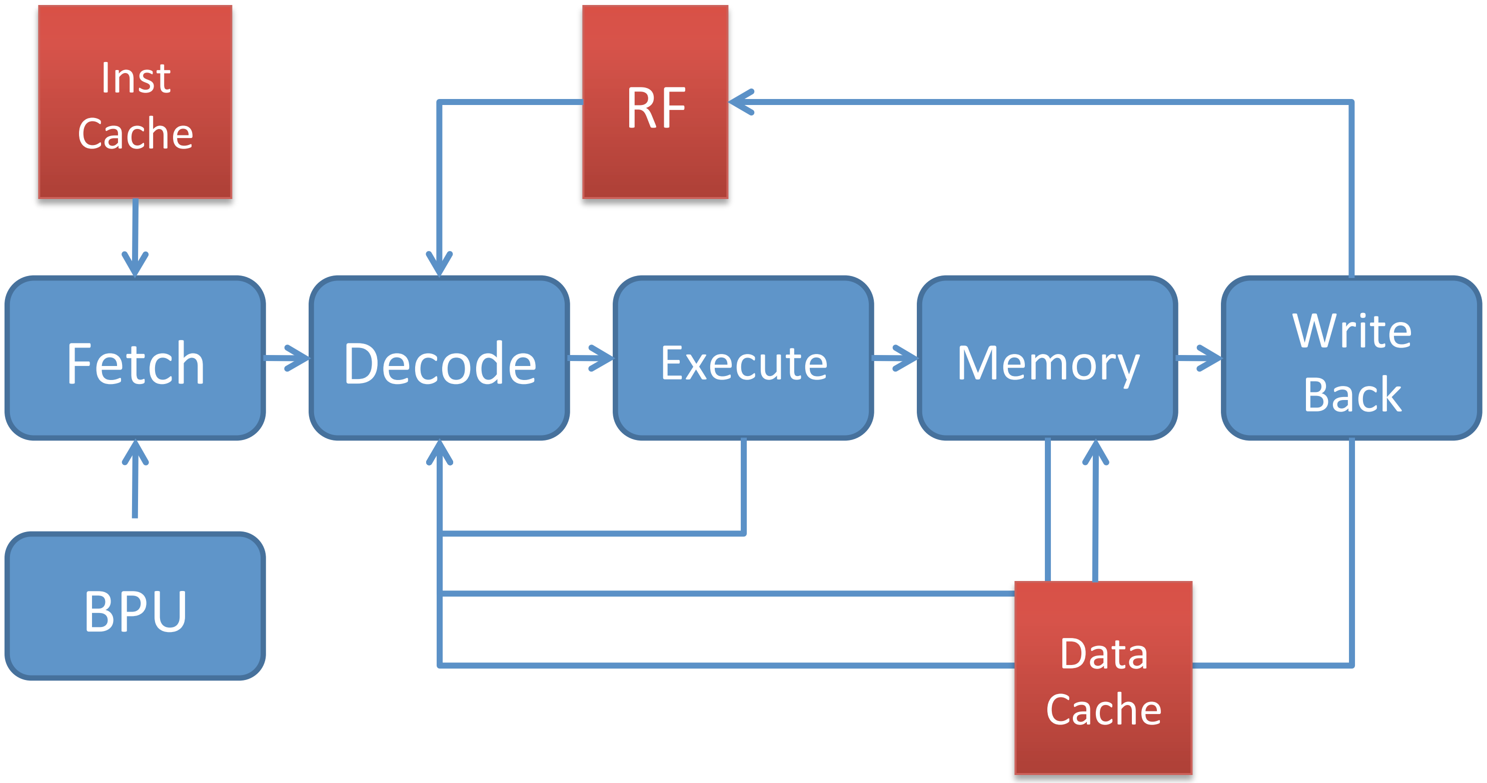
- All SHAKTI variants can have secure variants.
- Standard variant will have a Trustzone like functionality for a formally defined OS TCB.
- Other standard security blocks like HAB and security co-processor will be developed.
- Experimental T-class variants will have tagged ISA architecture.

Tagged ISA

- Various approaches being examined, final version will most likely use a combination of the free space available above the 43rd bit in 64 bit pointers for smaller tags and an indirect scheme for larger tags.

C- CLASS PROCESSOR
AND
FAULT TOLERANCE

C Class Processor



- 32-bit 5 Stage Pipeline with Branch Prediction.
- Supports all Integer Instructions.
- AHB/AXI-Lite bus

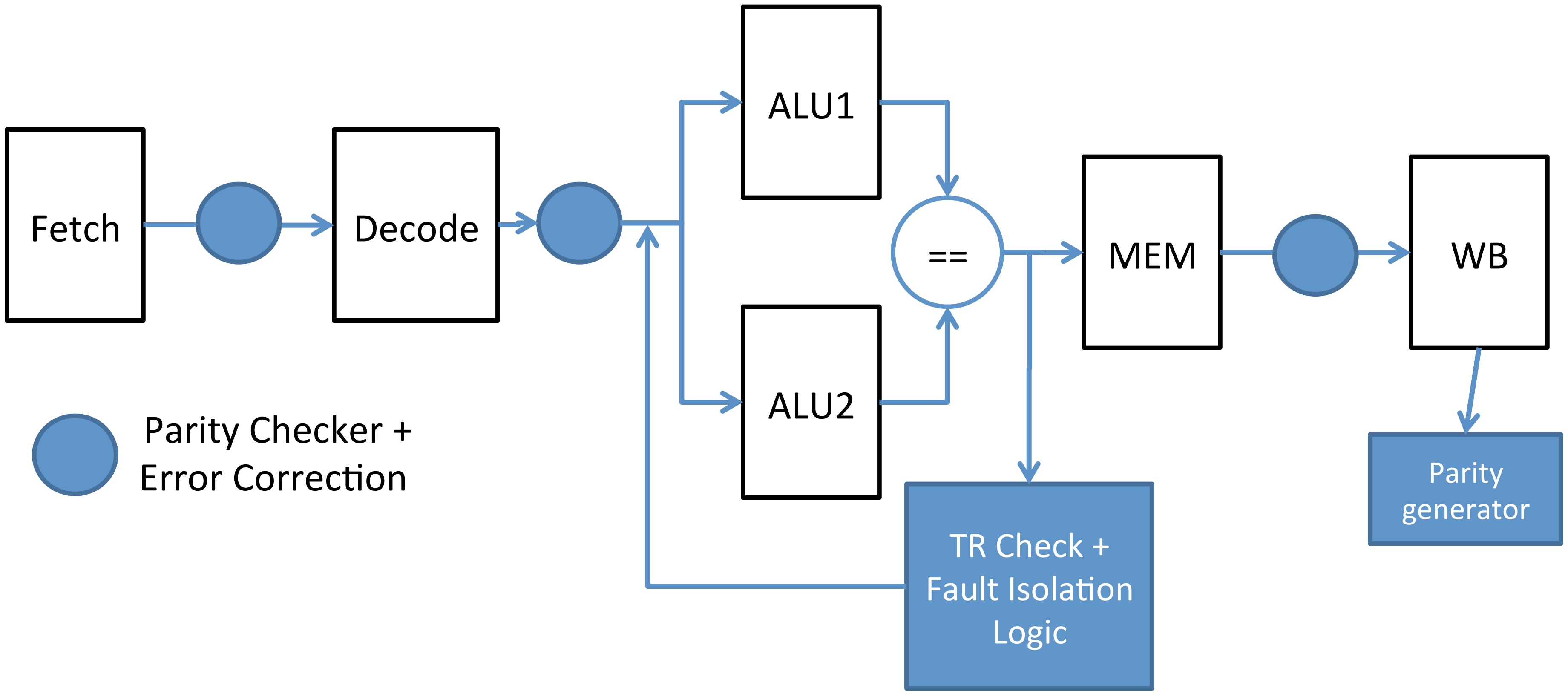
C Class Processor

- More Features
  - An L1 Cache.
  - Optional single and multi-cycle multiplier, DSP instr.
  - Memory protection (8 regions)
  - Tournament branch predictor.
  - Targeting 50-75MHz on FPGA.
  - Modularized RTL
- Target Applications
  - Non-MMU class control applications (Cortex M class)
  - Secure and FT variants

Fault Tolerant Variant

- Fault Tolerant features to meet the ISO26262 type standards (HW+SW eco-system)
- Will have companion formally verified micro-kernel OS
- Information Redundancy
  - Parity check at each pipeline stage
- ALUs
  - DMR – Dual Modular Redundancy (Parameterized).
  - Time redundancy to isolate faulty module.
  - Recomputation with different coding schemes for different Arithmetic operation for fault isolation.
  - Proposed technique bridges the gap between space redundancy and time redundancy.

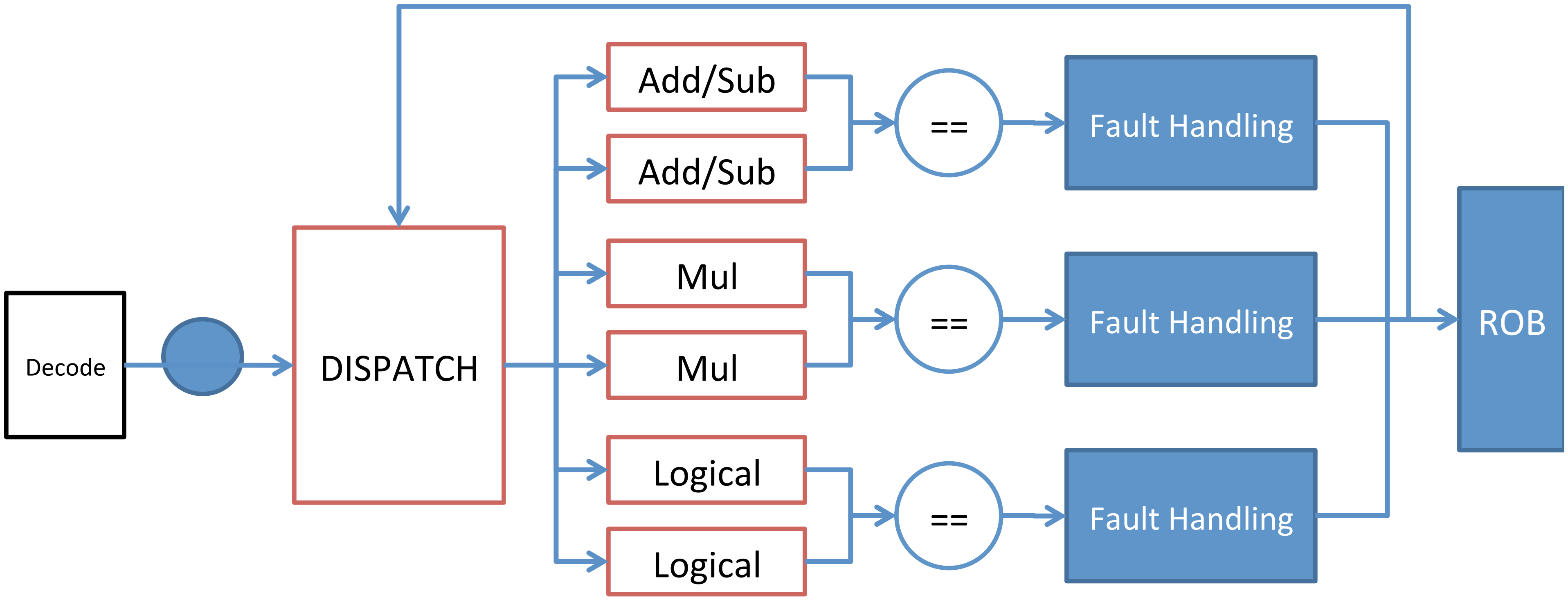
Micro-architecture



- Programmable Tolerance level.
    - The number of cycles for TR can be programmed to register.
- Programmable Confidence driven computing
    - Error resilient initiative.

Micro-architecture

- Performance enhancement through ILP
  - Split ALU into Functional Units.
  - Out of order dispatch.
  - Use ROB structure to maintain in-order commits.
  - Extra Stages Added.



Fault Handling Methodology

Cycle	Compare Result	Error Type	Action Taken
1	Match	No error	Posted from ALU1
2,3,4	Mismatch	Transient	No posting (Redo the operation)
5	As transient error persists for 3 consecutive cycles,Permanent error flag raised.Recomputing with different coding schemes for different operation to identify faulty ALU. Status of health flag of ALU's will be set to decide data selection and posting. Post the data from healthy ALU.		

Cycle	ALU 1 Health Flag	ALU2 Health Flag	Conclusion/Action Taken
6.	0	0	Not able to detect faulty ALU, Post '0' and status of health flag status will be send to ISR to decide
	0	1	ALU2 faulty. Isolate ALU2 and Post from ALU1 here onwards.
	1	0	ALU1 faulty. Isolate ALU1 and Post from ALU2 here onwards.
	1	1	Both ALU1 & ALU2 faulty. Post '0' and status of health flag status will be send to ISR to decide
7.	Post from one available healthy ALU		

Re-computation Methodology

Operation	Normal Operation	Recomputing Operation	Fault detection capability
ADD/ SUB	F = OP1 ± OP2 (Cin=0 implied)	F' = (~op1+1) ± ~op2 (+1 to take care cin=1)	As F & F' are complementary,stuck at failure at any bit position will be known.
Logical	F= OP1 and/ or/ xor OP2	Swapped Operands	As Upper half (31:16) bits will be done by lower half (15:0) circuit and vice versa. Error at any bit position will be detected.

Signed Multiplication	F1= op1* op2	F2=(2's Complement of op1)*op2
		
		F2 = -F1
		
		Comparison of 2's complement of F2 & F1 for error detection
			No error Condition :
			F1= 2^n - F2
			
			Stuck at failure at particular bit position i ,
			Rnormal= F1±2^i , Rrecomp = F2 or F2±2^i .
			2's complement of Rrecomp = 2^n -F2 or 2^n - (F2±2^i) = F1 or F1 -+ 2^i
			
			Idea is, if Rnormal increases then 2's complement of Rrecomp2 will remain same or decreases and vice versa therefore ERROR detected.

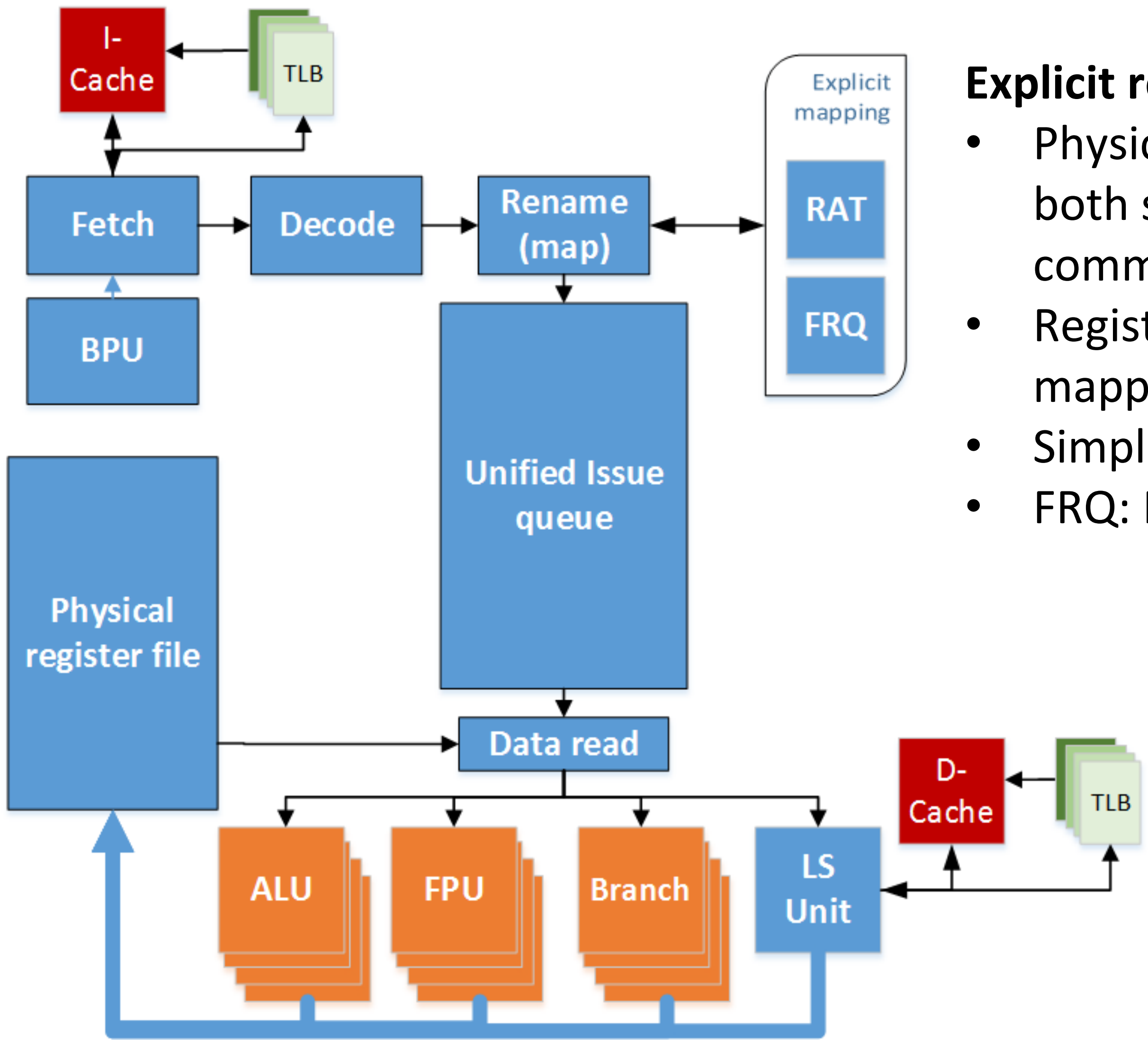
Unsigned Multiplication	Here we have to append the zero at the MSB and use the signed multiplication method for fault handling		

I CLASS PROCESSOR

I Class Processor

- 500Mhz – 1Ghz class multi-core variant
- Supports all 64-bit RISCV Instructions.
- Parameterized Issue width.
- IEEE-754 single and double precision FPU.
- Enable quick design space exploration.
  - Highly parameterized
- Initial target – 45/65 nm

Micro-Architecture



Explicit renaming:

- Physical register file stores both speculative and committed values.
- Register alias table stores mapping.
- Simple commit logic.
- FRQ: Free Register Queue

PRF vs ROB renaming

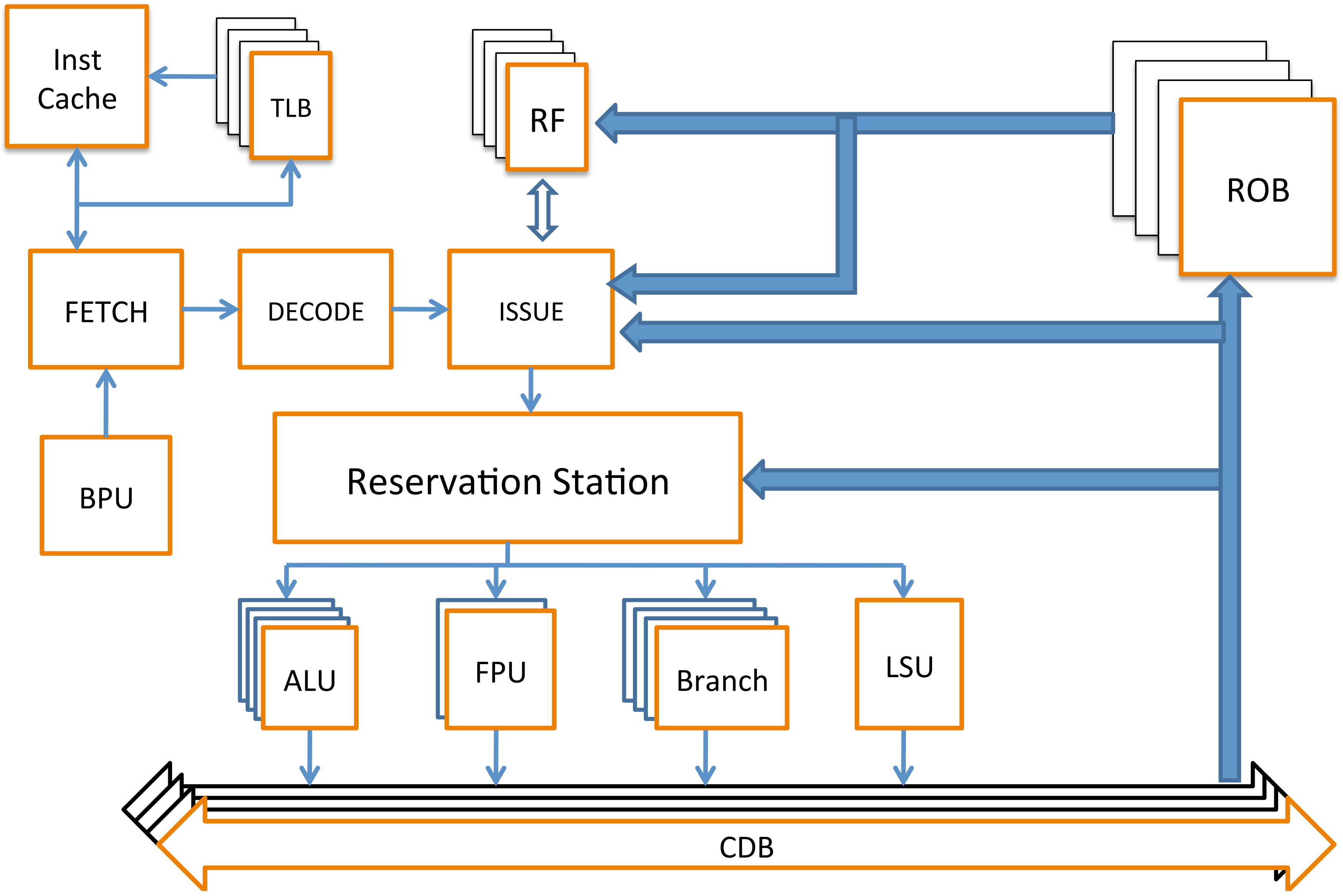
- Physical Register File :
  - Decouples Issuing and operand fetch.
  - One source for all operands.
  - Less Data movement.
  - No WAW and WAR checks required at all
  - Revert Mapping in-case of mis-prediction/ exception
  - Scalable
- ROB :
  - In order commit – precise exception handling.
  - Plenty of data movement for operand forwarding.
  - With RS allows distributed scheduling points.

S-CLASS PROCESSOR AND SERVER FABRICS

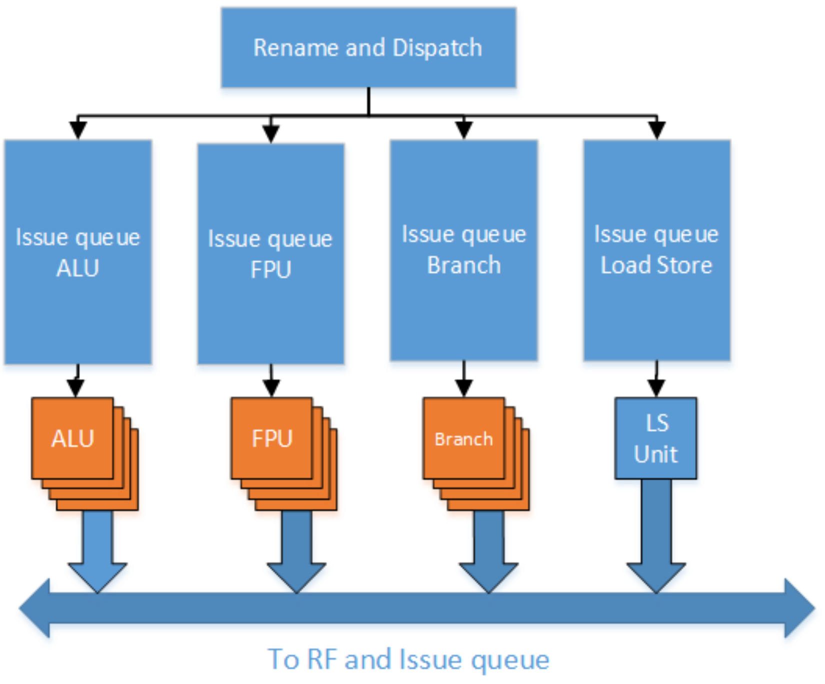
S- Class Processor

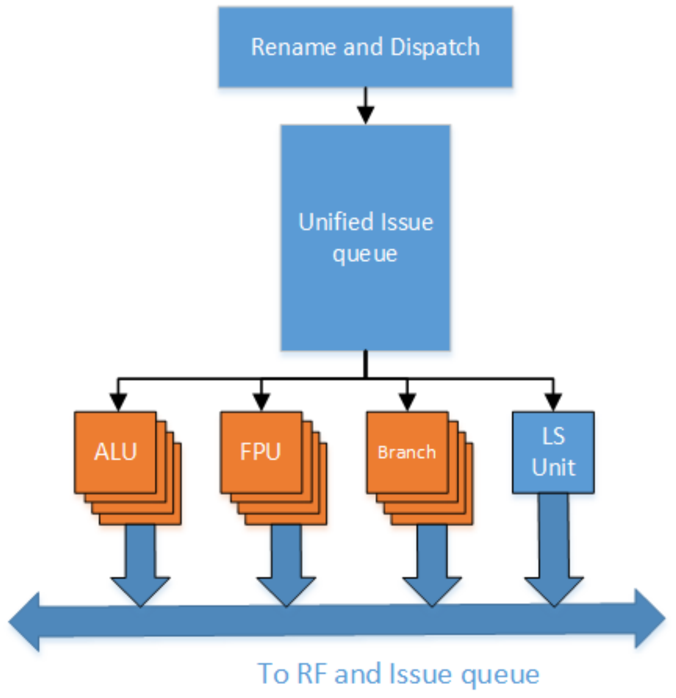
- 64 bit SMT variant for server applications
  - Virtualization supported (CPU and I/O)
  - 1-3 Ghz
- SoC configuration – Clusters of 8 cores/cluster each connected via fabric for 32 clusters (total 256 cores).
  - Intra cluster – Ringbus
  - Inter-cluster – Mesh, Ring, NoC
  - Socket to Socket CC interconnect – SRIO GSM based
- Memory/Cache
  - Private L1, shared/private L2 and shared segmented L3 cache
  - Main Memory – DDR4/Hybrid Memory Cube
- Parameterized for N-threads (8 max) and M-issue(4 max).

Micro-Architecture



Distributed Vs Centralized Issue Queue

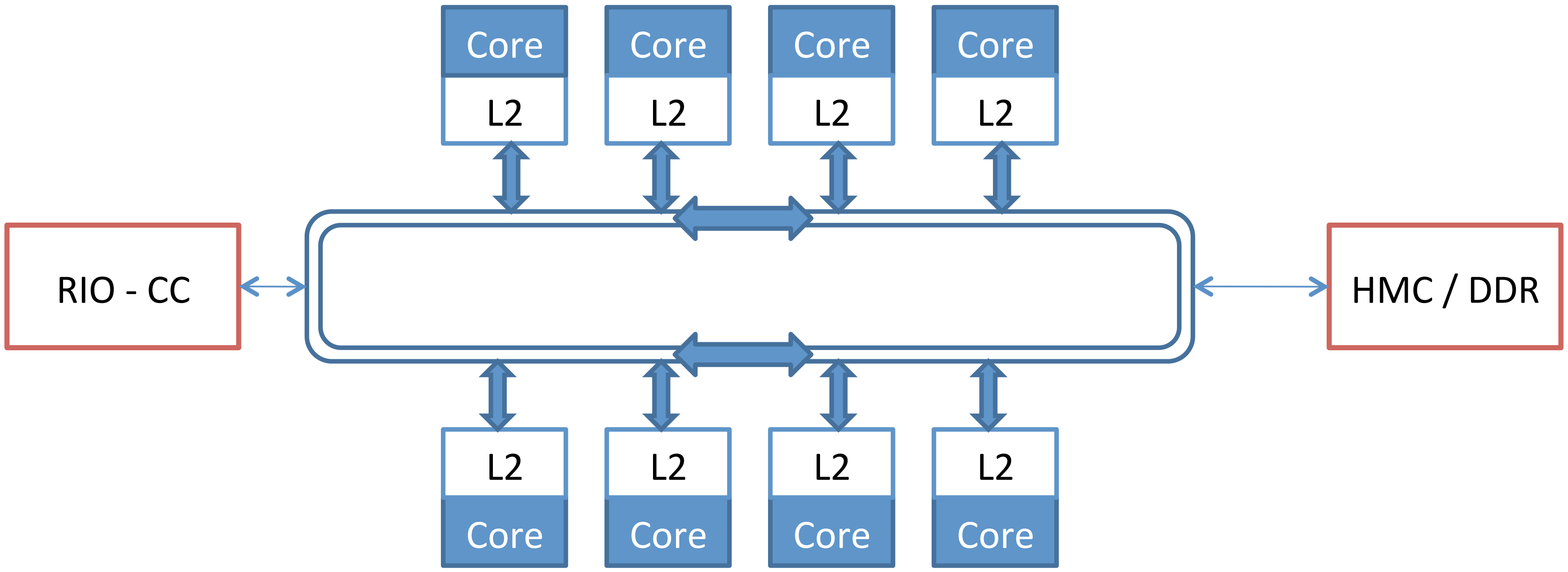




- Distributed IQ:
    - Each smaller sub-IQ holds specific type of instructions.
    - Lesser issue logic delay compared to unified IQ.
    - Easier to power gate unused queue.
- Unified IQ:
    - Holds all types of instructions.
    - Effective usage of IQ space.
    - Has higher issue logic delay.
    - Higher instruction window compared to distributed IQ.

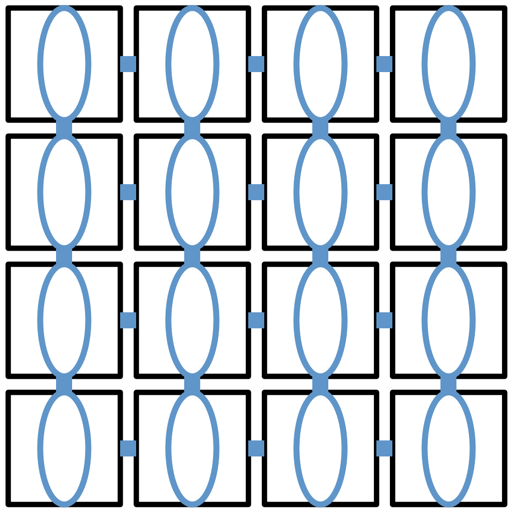
Server Fabrics for the S-Class

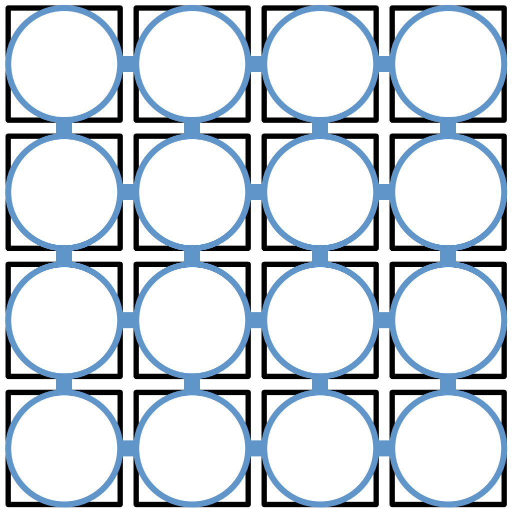
- Hybrid Ring + Mesh fabric
  - Each cluster uses 512 bit Bi-directional Ring for up to 8-cores.
  - Closely coupled MOESI variant with home node
  - Power, Area and verification efficiency over mesh
  - Memory ordering semantics easier to prove

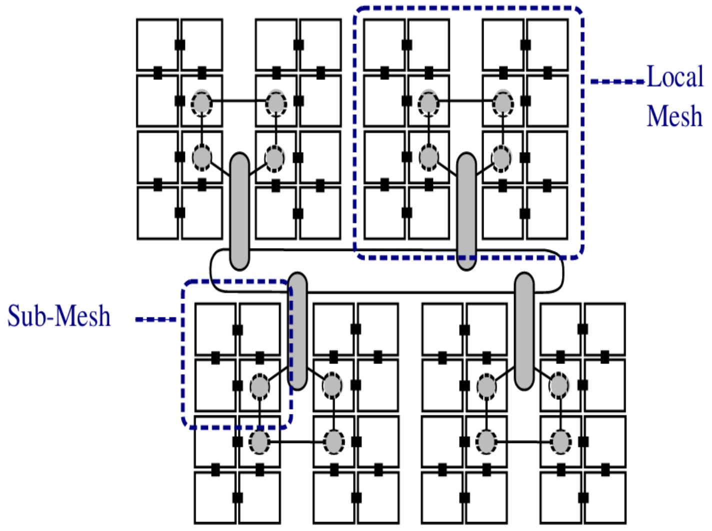


Server Fabrics (contd.)

- Hybrid Interconnect scheme
  - Rings scale well for up to 8 cores.
  - Inter ring connection via mesh or ring depending on server characteristics
  - Cache coherency will be directory based and will depend on CC-NUMA aware SW architecture


Mesh of Rings with 2 bridges

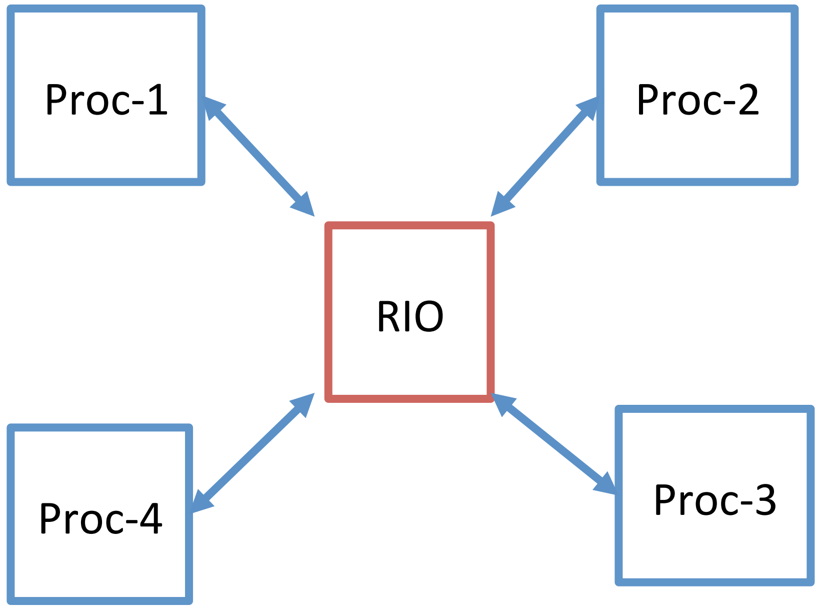

Mesh of Rings with 4 bridges



Hybrid with hierarchical rings

Socket Interconnect

- 10/25G RapidIO for socket-socket connection
  - A 5+ state MOESI/MESIF like directory based protocol will be the CC protocol
  - Scalable up to 128 sockets (16 clusters of 8 sockets)
  - Quasi-parallel SRIO variant being examined to reduce latency since SERDES latency is too high (vis a vis QPI)



Adaptive System Fabric

- Proposed experimental fabric to unify Memory, I/O and Networking for clusters and data-centers.
- Based on persistent memory based systems with NVM storage
- Initial candidate being explored
  - A hybrid memory cube + RapidIO fabric
  - Common PHY + physical connectivity for both fabrics
  - Application specific protocols will dynamically configure an interconnect link for specific purpose
  - The fabric's topology can be changed dynamically to suit the workload in questions. The intelligence will reside in the fabric router

Adaptive System Fabric

- A Microkernel based OS architecture is also being proposed to take advantage of the HW fabric, computation can dynamically move to where the data is located
- Berkeley Spark/Tachyon being used to evaluate viability of architecture for data analytics scenarios

Thank you

- Contact and further info :
  - Madhusudan : mail@gsmadhusudan.net
  - Neel : neelgala@gmail.com
  - SHAKTI : http://rise.cse.iitm.ac.in/shakti