Abstract Machines for RASCs and Signal/Image Processing

L. Mullin*, J. Raynolds*, I. Grout, J. Ryan, and Q. Li*

University at Albany, SUNY* & University of Limerick, Ireland

HPEC 2006

Levels of Processor/Memory Hierarchy

continued

y= conv (x)

Map

intricate math

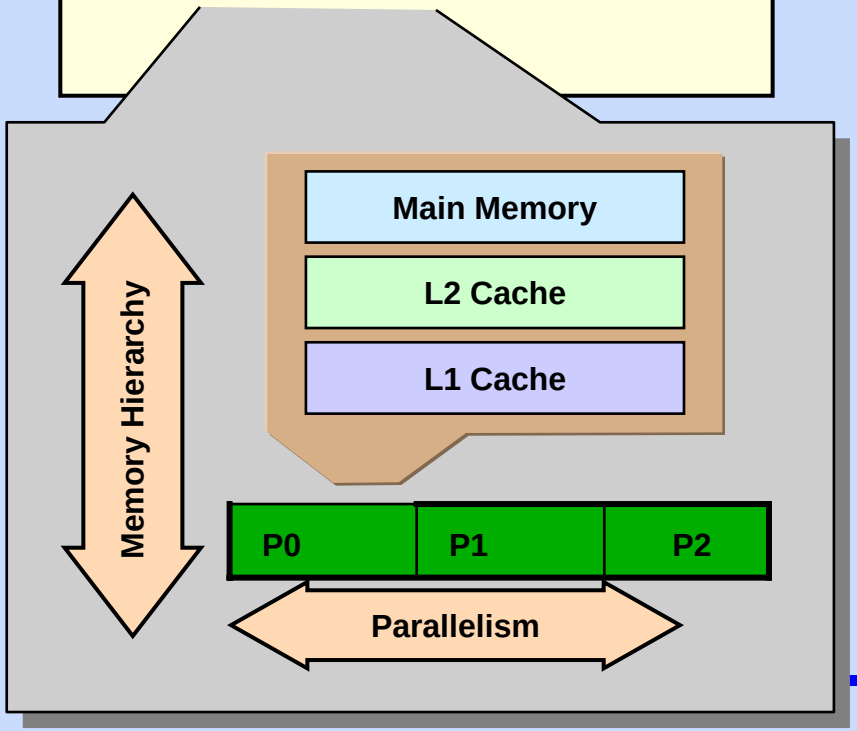
intricate memory accesses (indexing)

Approach

Mathematics of Arrays

- Math and indexing operations in same expression
- Framework for design space search
  - Rigorous and provably correct
  - Extensible to complex architectures

Example: "raising" array dimensionality



x: < 0 1 2 … 35 >

Map:

P0: < 0 1 2 > < 3 4 5 > < 6 7 8 > < 9 10 11 >

P1: < 12 13 14 > < 15 16 17 > < 18 19 20 > < 21 22 23 >

P2: < 24 25 26 > < 27 28 29 > < 30 31 32 > < 33 34 35 >

Abstract Machine Concept

- Abstract Machine is a one-to-one mapping between the various levels of hardware and software cache, memory, disk, network, FPGA etc. onto multi-dimensional arrays.
- The Abstract Machine is an algebraic specification of the algorithm using Mullin's formalism of A Mathematics of Arrays (MoA) and its corresponding indexing scheme the psi-calculus.
- The Abstract Machine is an exact Engineering Specification derived mathematically that can be given to a software engineer for direct, step by step, implementation into any convenient hardware and/or software language.
- We use an Abstract Machine approach to the efficient implementation of our cache-optimized FFT algorithm on a RASC architecture in which various levels of the FFT are efficiently distributed over hardware (FPGA) and software.

Arbitrary RASC/FPGA

Fast-calculations directly in hardware

Slower calc's in software and controlled by OS

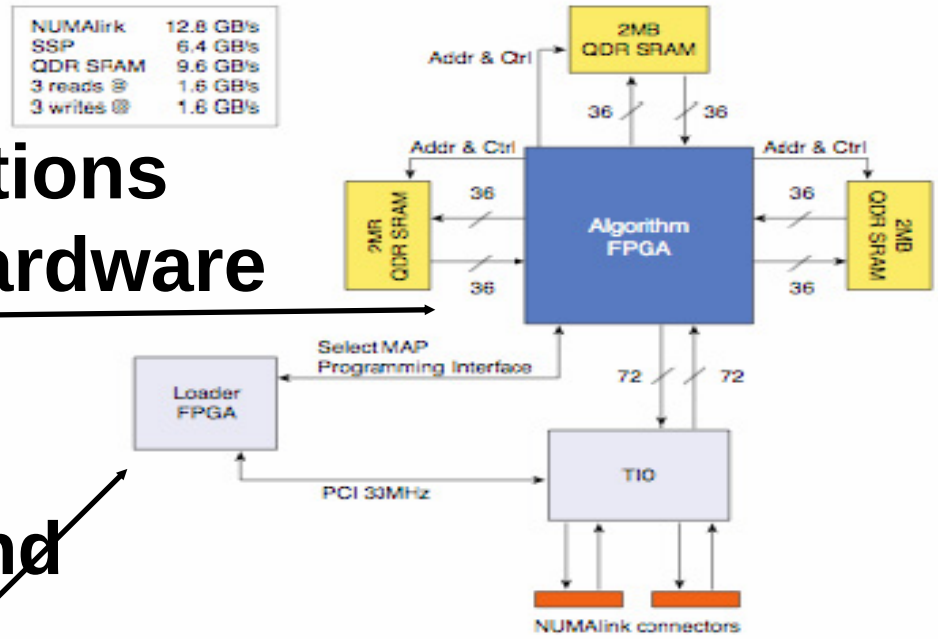
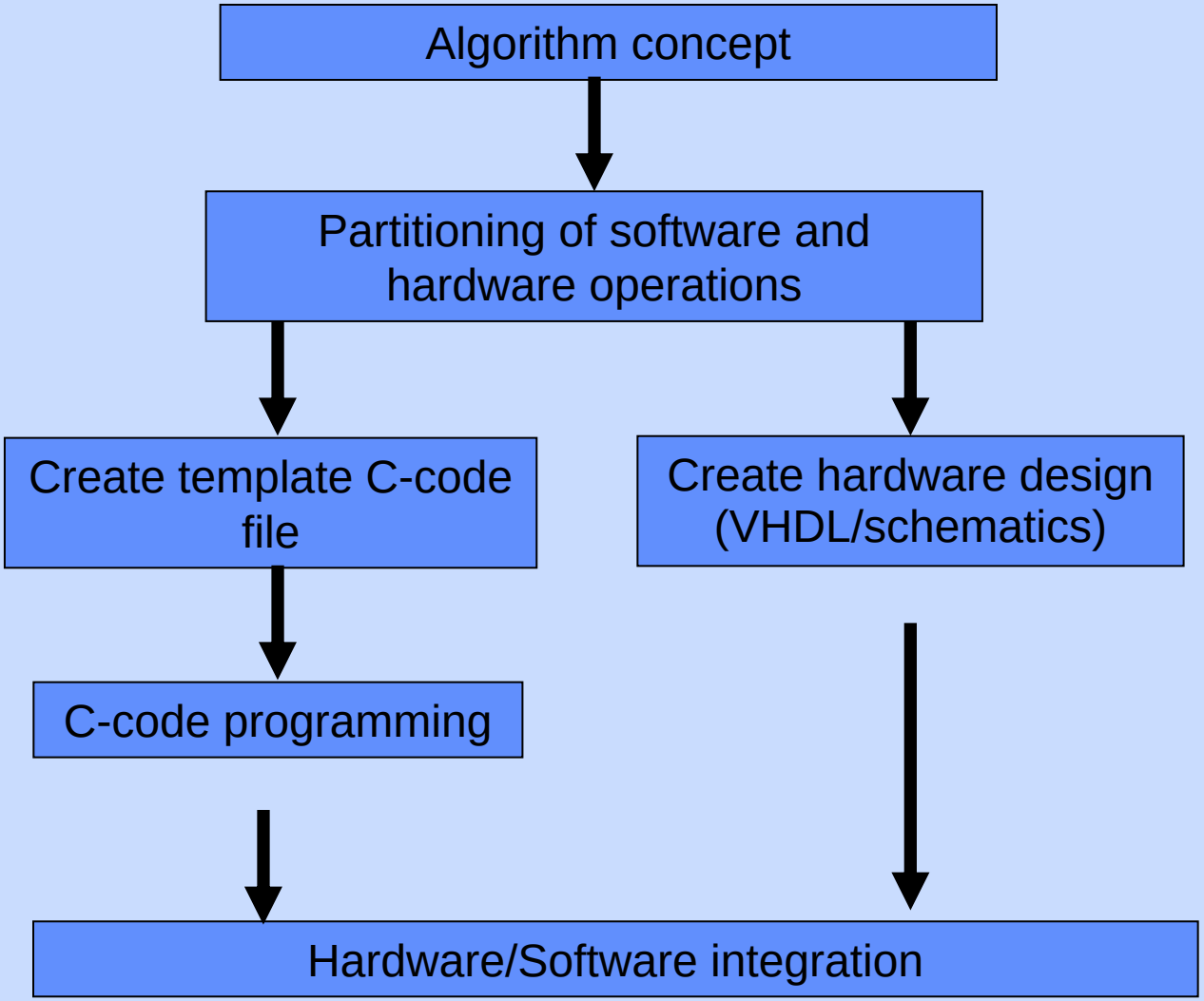


Figure 1-2 RASC FPGA Functional Block Diagram

Figure 1-3 shows a block diagram of RASC Scalable System Port (SSP) Field Programmable Gate Array (FPGA) with Core Services and Algorithm Block. The FPGA is connected to an SGI Altix system via the SSP port on the TIO ASIC. It is loaded with a bitstream that contains two major functional blocks:

- The reprogrammable algorithm
- The Core Services that facilitate running the algorithm

Design Flow


Algorithm concept
Partitioning of software and hardware operations
Create template C-code file
Create hardware design (VHDL/schematics)
C-code programming
Hardware/Software integration

Field Programmable Gate Arrays

For use in RASC systems:

- Large number of available logic gates and interconnect resources to implement complex DSP functions.

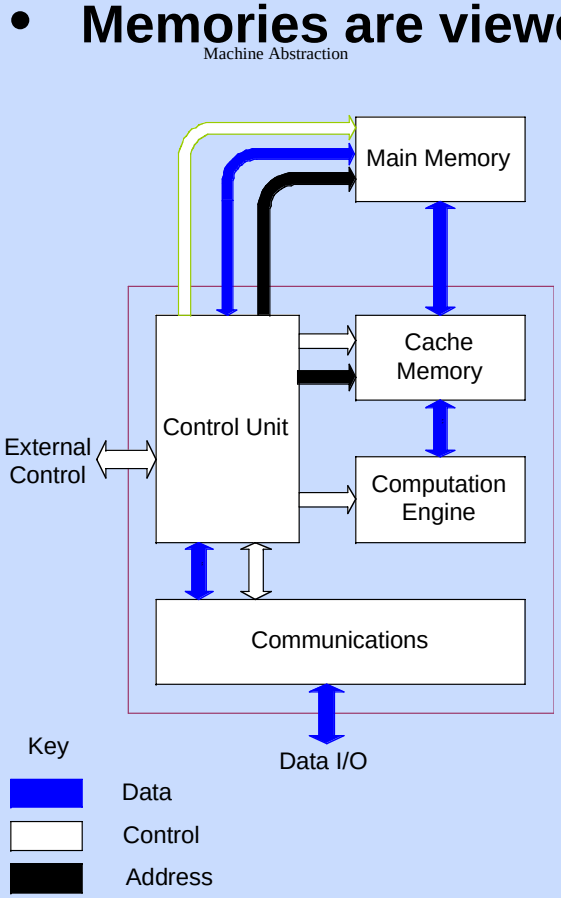
- Specific FPGAs come with dedicated hardware blocks such as embedded 2's complement multipliers (combinational logic designs that do not require multiple clock cycles to complete the operation) allowing for the support of high-speed operations.

- Specific FPGAs come with embedded SRAM blocks to allow for temporary storage of data.

Machine Abstraction

FFT with Cache Algorithm

- **Memories are viewed as arrays**



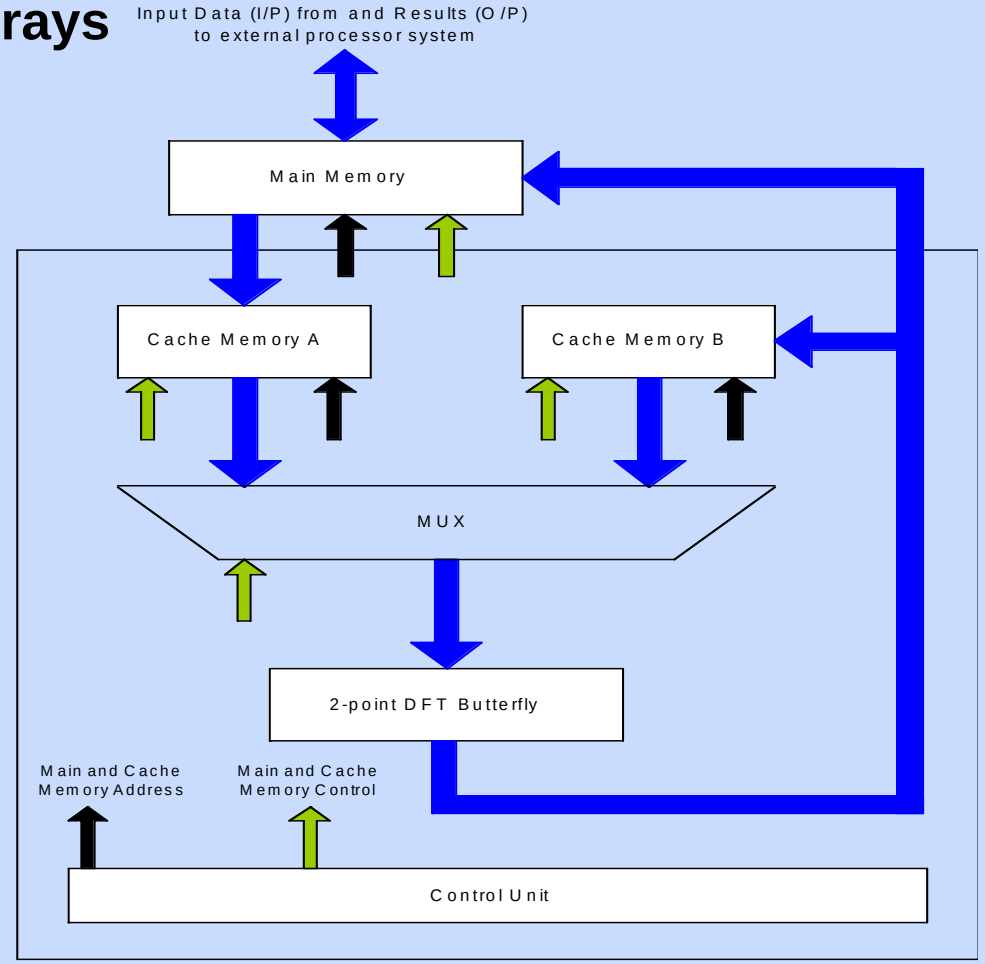


Figure 2. FFT architecture

Summary: FFT Cache Algorithm to FFT Machine with Cache

- **Maximize in-cache operations through use of repeated transpose-reshape operations**
  - **Re-materialize the array to achieve locality.**
  - **Do as many operations in cache as possible and repeat process**
  - **Operations carried out at the price of rearranging**
  - **Cache loop is an iteration space**
  - **Could also be over processors, etc.**
- **Data blocks, *abstract cache sizes*, can be of any size (powers of the radix):**
- **Optimum performance: tradeoff between reduction of cache misses and cost of transpose-reshape operations, i. e. rearranging.**
- **Costs are determined by combining iteration space from *normal from* to *abstract machine* and actual costs of access and control.**
- **Uniform view and analysis using an *array algebra and index calculus: Conformal Computing.***
- ***Cost-intensive calculations carried out in hardware (FPGA) for extreme speed seemlessly integrated with faster software calculations by design.***