Custom Memory Cube (CMC)

CHREC
NSF Center for High-Performance Reconfigurable Computing

RC Class Presentation

Gongyu Wang
Ph.D candidate

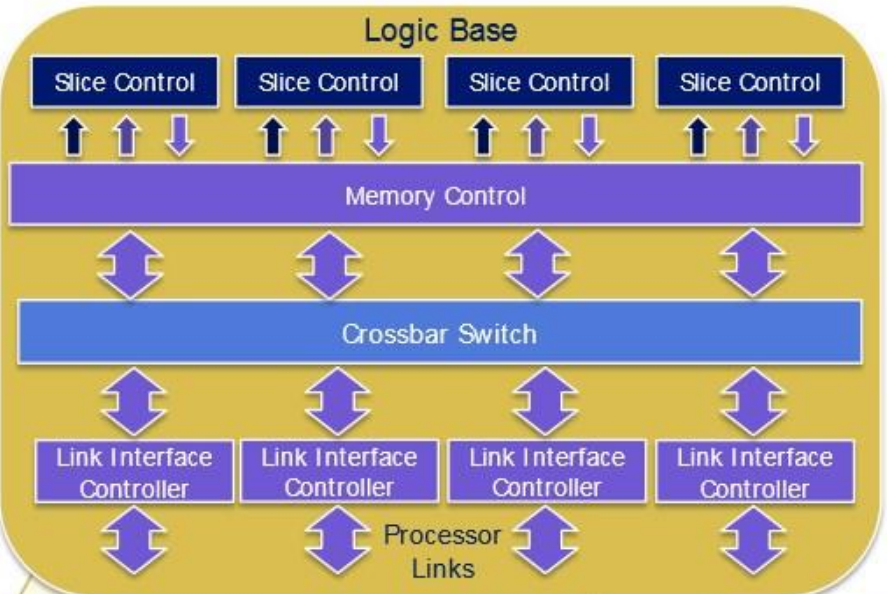
UF UNIVERSITY of FLORIDA
Virginia Tech VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
BYU BRIGHAM YOUNG UNIVERSITY
THE GEORGE WASHINGTON UNIVERSITY WASHINGTON DC

Mar. 30th, 2015

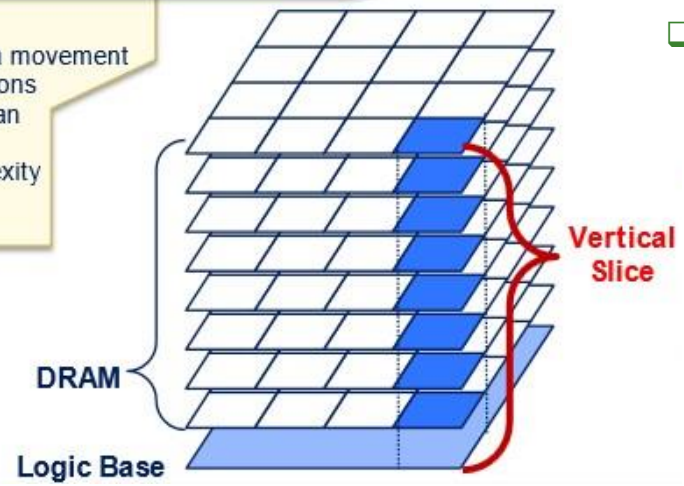
Agenda

- Introduction: Hybrid memory cube (HMC)
- Motivation and goals for CMC
- Approach to CMC
  - Two phases
  - FPGA/HMC board – Merlin
- Programming Merlin
  - Hybrid-threading toolset
- Potential RC class course projects

Introduction



Logic Base
- Wide, high-speed local bus for data movement
- Advanced memory controller functions
- DRAM control at memory rather than distant host controller
- Reduced memory controller complexity and increased efficiency

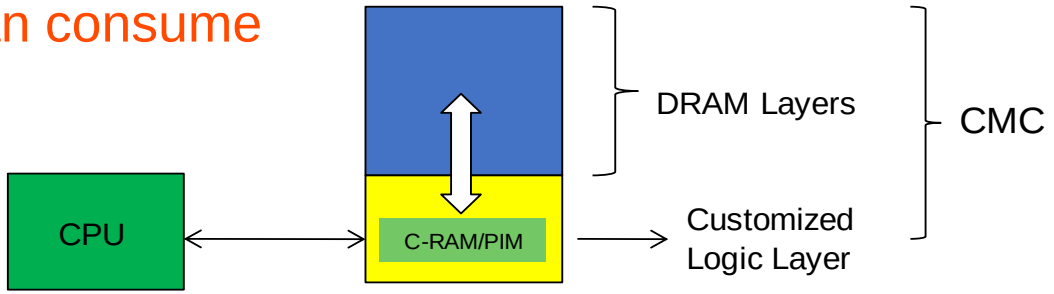


- Hybrid memory cube
  - High-performance RAM interface for through-silicon vias (TSV)-based stacked DRAM memory
  - Logic base (layer)
- High bandwidth
  - ~10% energy per bit of current DDR memories

	Bandwidth	Power	W / GBs
DDR3-1333	10.66 GB/s	5.52 W	518 x
DDR4-2666	21.34 GB/s	6.60 W	309 x
HMC (4 DRAMs)	128.00 GB/s	11.08 W	87 x

Custom Memory Cube (CMC)

- Motivation
  - Increasing need for fast and low-power parallel processing
  - Memory accesses of data-intensive apps consumes considerable time and power
  - HMC presents much higher memory bandwidth than current processors can consume



- Goals
  - HMC as high-bandwidth memory
    - Exploit performance and power advantages
  - CMC: Develop a customizable logic layer for HMC
  - Offload processing tasks to CMC
    - Concepts of C-RAM & PIM (e.g., SIMD, ARM64, & RISC-V in logic layer)

Approach to CMC

HMC apps

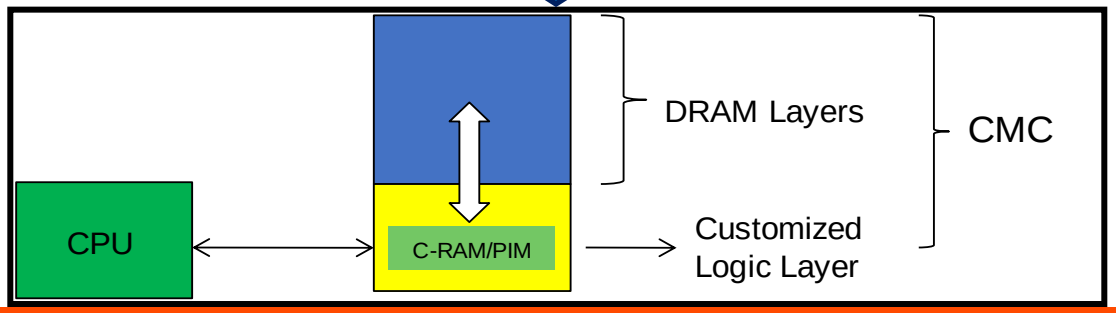
- Study HMC device using FPGA/HMC boards
- Measure and report performance of HMC accesses

HMC Studies — phase 1

CMC apps

- Prototype CMC system
  - Use FPGA/HMC boards
- Experiment with concepts of C-RAM, PIM, etc.

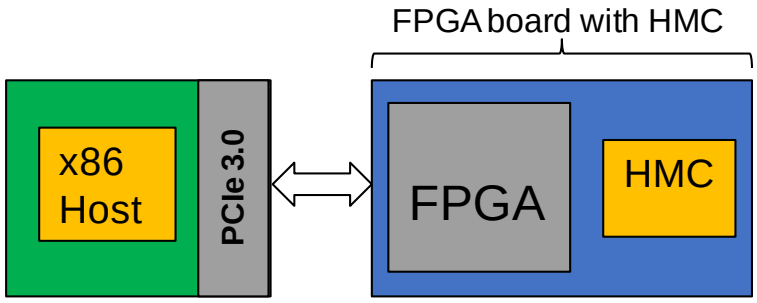
CMC Prototyping — phase 2



HMC: Hybrid-Memory Cube
C-RAM: Computational RAM
PIM: Processor in Memory

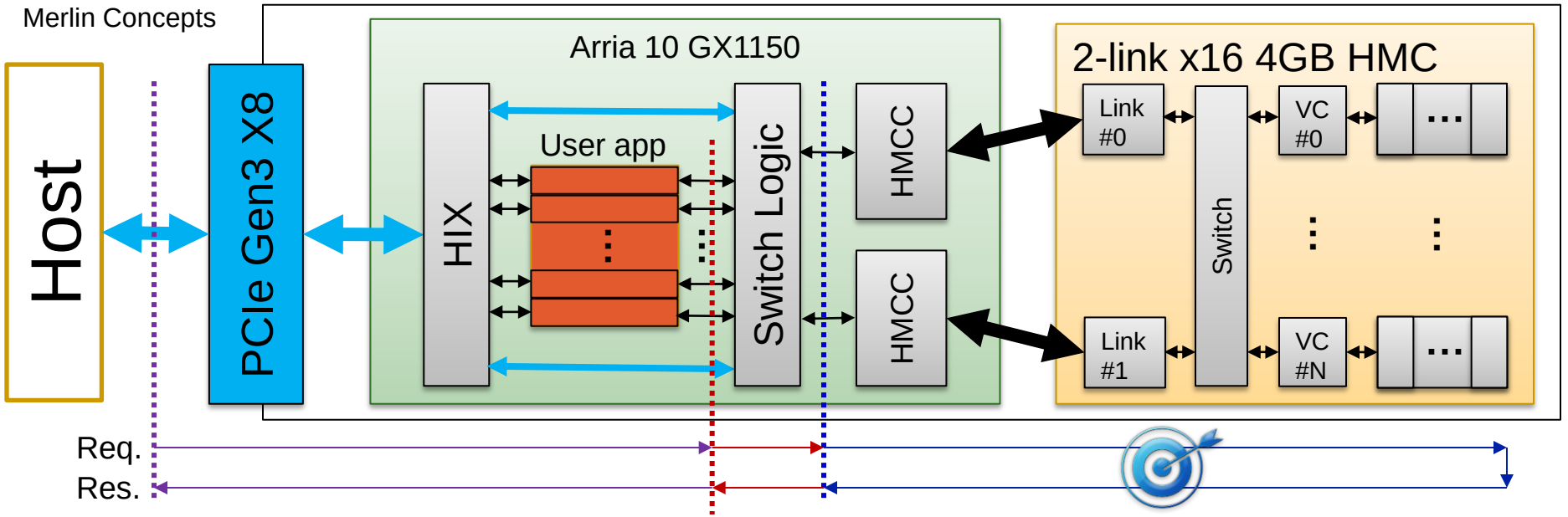
Phase 1

- HMC studies
  - Test and study HMC's amenability in selected app. domains
    - Amenability: bandwidth, latency, power, etc.
    - App. Domains: sort, search, cryptography, FFT, etc.
- Approach
  - Develop test applications on FPGA/HMC board
  - Measure report access time of HMC in FPGA and report results



System architecture

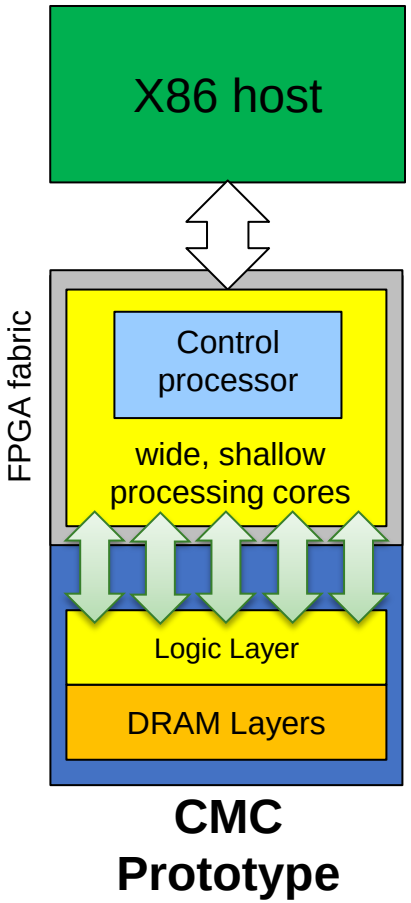
Phase 1 with Merlin Board



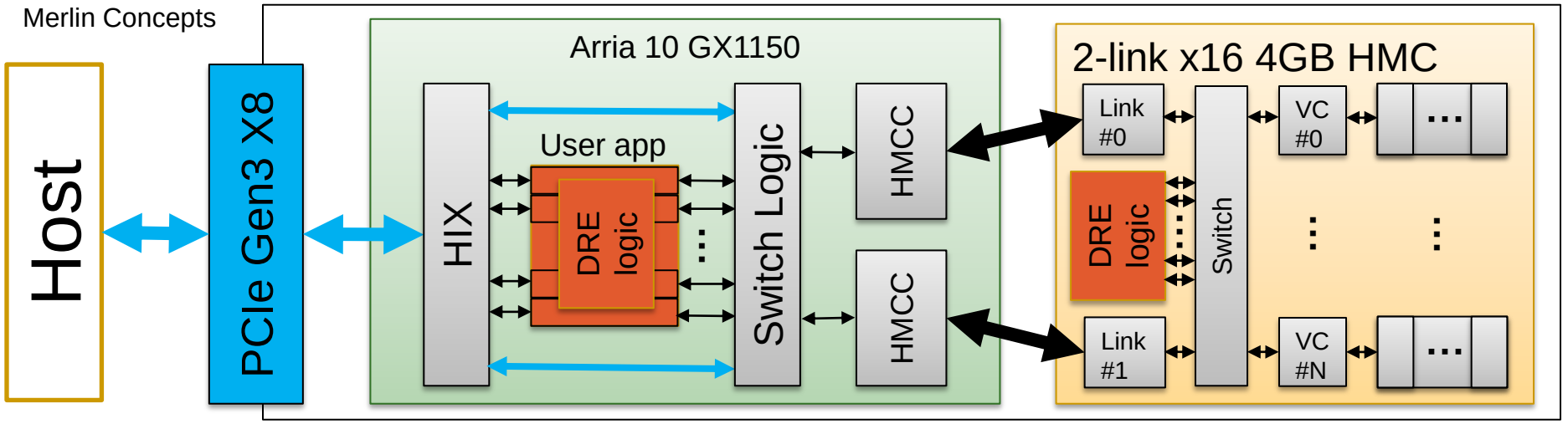
- Goal: test various user apps and report access time of HMC
  - **Wanted latency**: Round-trip latency at inputs of HMCC
  - **Unwanted latencies**: PCIe, HIX, switch logic
  - **Ability to set parameters of HMC access**: packet size, etc.
- Approaches to measuring route-trip latency of HMC access
  - Direct measurement OR subtract unwanted

Phase 2

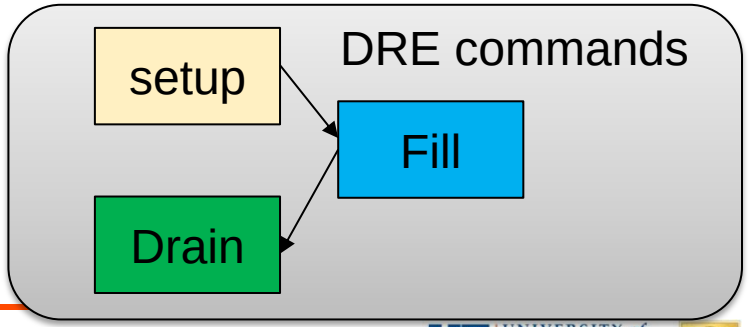
- Prototype CMC using FPGA/HMC boards
  - Emulate processing capabilities of CMC on FPGA
- Tasks
  - Define CMC operations based on apps
  - Explore processing capabilities
    - Data processing: "wide" (many-core) and "shallow" (simple) processors; e.g., SIMD
    - Control: e.g., Simplified RISC-V/ARM64
  - Develop demo applications



Phase 2 with Merlin Board



- Prototype CMC
  - Leverage LLNL's DRE* work for initial prototyping
  - Long term: evaluate CMC based on requirements of other relevant apps
- Tasks
  - Study, port, and modify DRE for our need
    - Port to Arria 10 FPGA on Merlin board
  - Prototype CMC architecture in FPGA
  - Develop demo applications



*DRE: Data Reordering/Rearrangement Engine

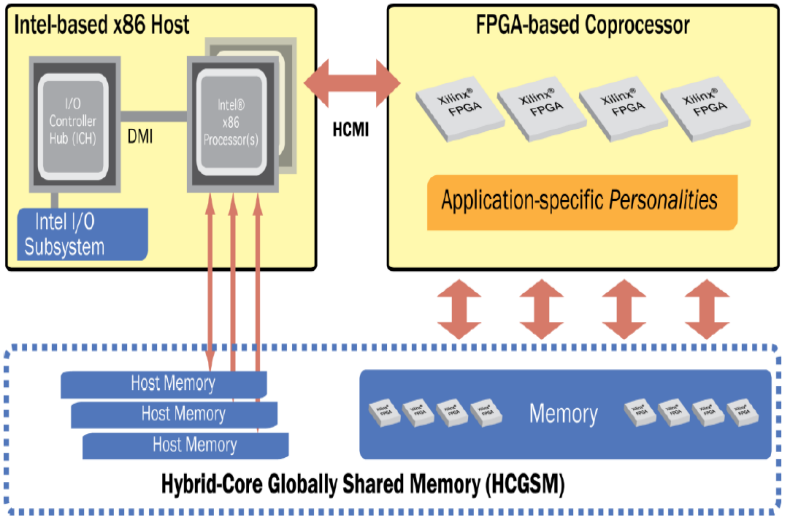
Agenda

- Introduction: Hybrid memory cube (HMC)
- Motivation and goals for CMC
- Approach to CMC
  - Two phases
  - FPGA/HMC board – Merlin

- Programming Merlin
  - Hybrid-threading toolset
- Potential RC class course projects

Convey HT Background

- Convey computers
  - Startup since 2006, based @ Texas
  - Hybrid-core computing platforms
  - Joined CHREC in 2013
    - Via F3, focusing on bioinformatics app acceleration
- Convey system philosophy – hybrid-core computing

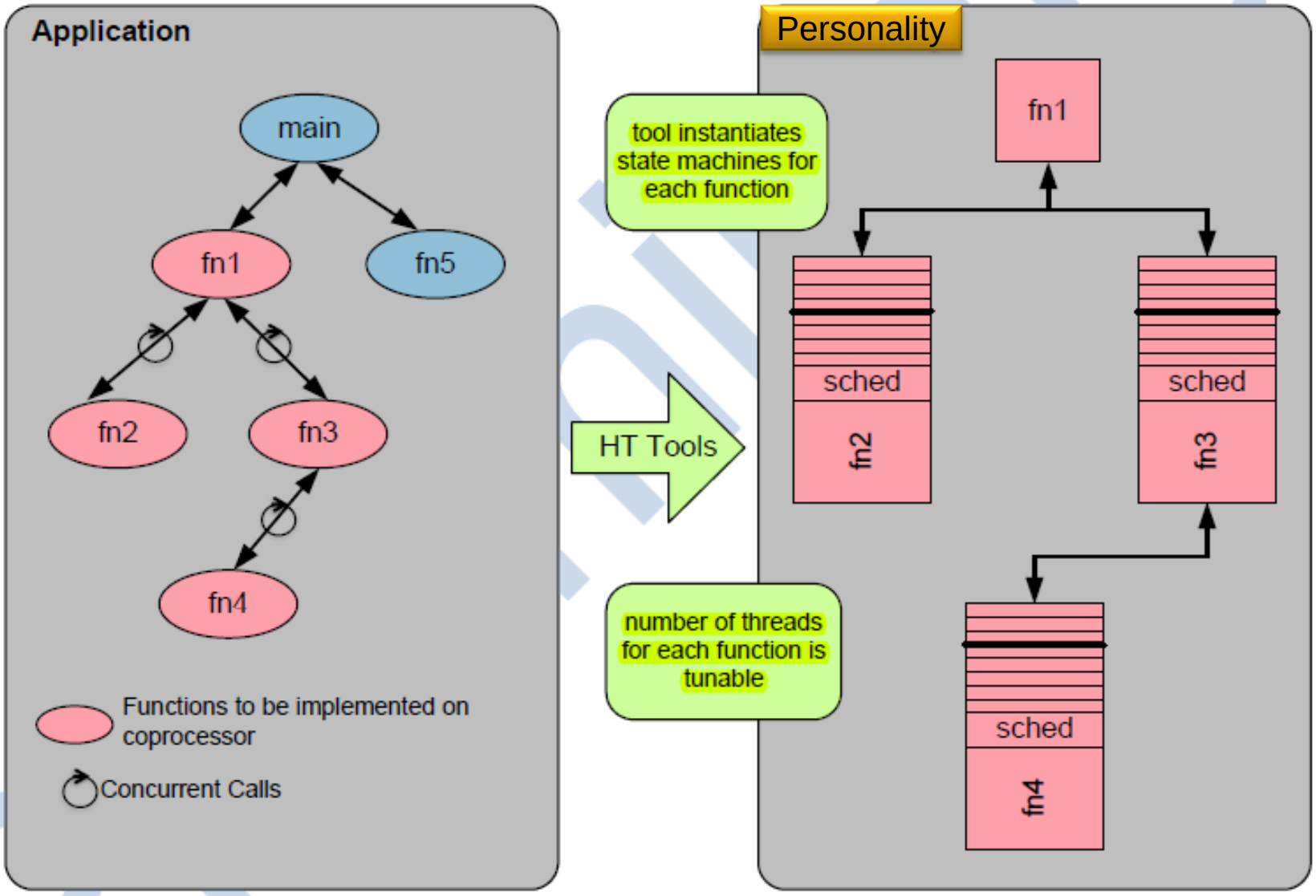


Features	Gidel	Convey	SRC
Programming	C/Cpp + HDL	C/Cpp/ASM + **HT** + (HDL)	C/C++
Data Addressing	explicit, separate	explict, consistent	implicit
Host Connection	PCIe	FSB, QPI*	PCIe

* via PCIe

HT : Hybrid threading - Convey's HLS-like programming tool

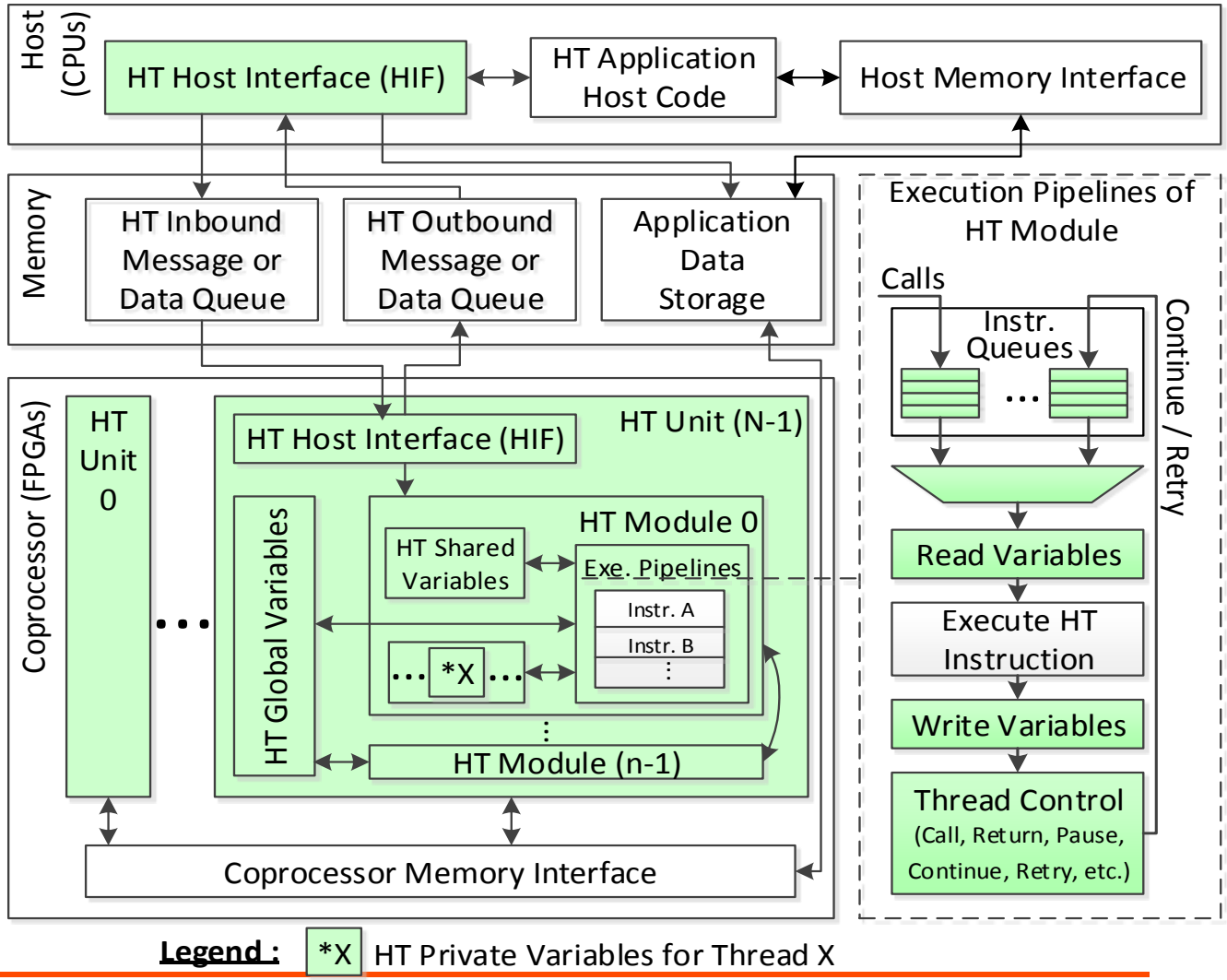
Overview of HT Programming Model



Application "call-graph" structure

HT Infrastructure

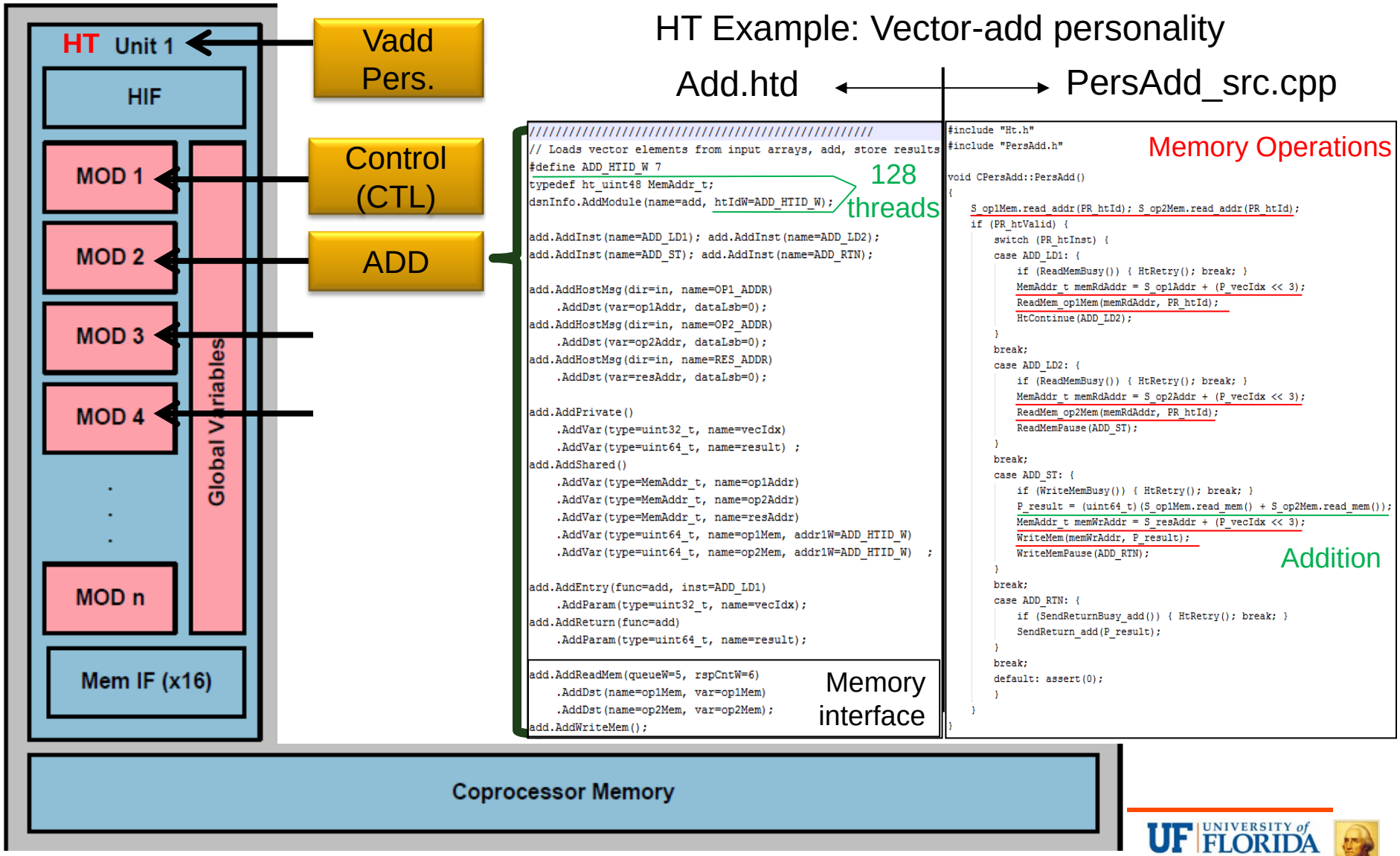
- Host-coproc. strucutre
  - Message/Data interfaces
- HT unit
  - Replication mechanism
- HT module
  - Essential functional blocks
  - Memory hierarchy
  - Can be replicated
  - Messaging interfaces among modules
- Instructions
  - User-defined behaviors of modules



User Programmable Parts

- Hardware description files (.htd)
  - Declare modules, variables, memory interfaces, etc.
- Module instruction files (.cpp)
  - Define behaviors of modules, including memory I/O, messaging interface, computation, etc.
- CTL module example on next slide
  - Responsible for:
    - Accept calls from host
    - Spawn threads onto other HT modules
    - Return to host after all threads returned from other HT modules

HT Code Example




```
//////////////////////////////////////////////////
// Loads vector elements from input arrays, add, store results
#define ADD_HTID_W 7
typedef ht_uint48 MemAddr_t;
dsnInfo.AddModule(name=add, htIdW=ADD_HTID_W);

add.AddInst(name=ADD_LD1); add.AddInst(name=ADD_LD2);
add.AddInst(name=ADD_ST); add.AddInst(name=ADD_RTN);

add.AddHostMsg(dir=in, name=OP1_ADDR)
    .AddDst(var=op1Addr, dataLsb=0);
add.AddHostMsg(dir=in, name=OP2_ADDR)
    .AddDst(var=op2Addr, dataLsb=0);
add.AddHostMsg(dir=in, name=RES_ADDR)
    .AddDst(var=resAddr, dataLsb=0);

add.AddPrivate()
    .AddVar(type=uint32_t, name=vecIdx)
    .AddVar(type=uint64_t, name=result) ;
add.AddShared()
    .AddVar(type=MemAddr_t, name=op1Addr)
    .AddVar(type=MemAddr_t, name=op2Addr)
    .AddVar(type=MemAddr_t, name=resAddr)
    .AddVar(type=uint64_t, name=op1Mem, addr1W=ADD_HTID_W)
    .AddVar(type=uint64_t, name=op2Mem, addr1W=ADD_HTID_W)  ;

add.AddEntry(func=add, inst=ADD_LD1)
    .AddParam(type=uint32_t, name=vecIdx);
add.AddReturn(func=add)
    .AddParam(type=uint64_t, name=result);

add.AddReadMem(queueW=5, rspCntW=6)
    .AddDst(name=op1Mem, var=op1Mem)
    .AddDst(name=op2Mem, var=op2Mem);
add.AddWriteMem();
```

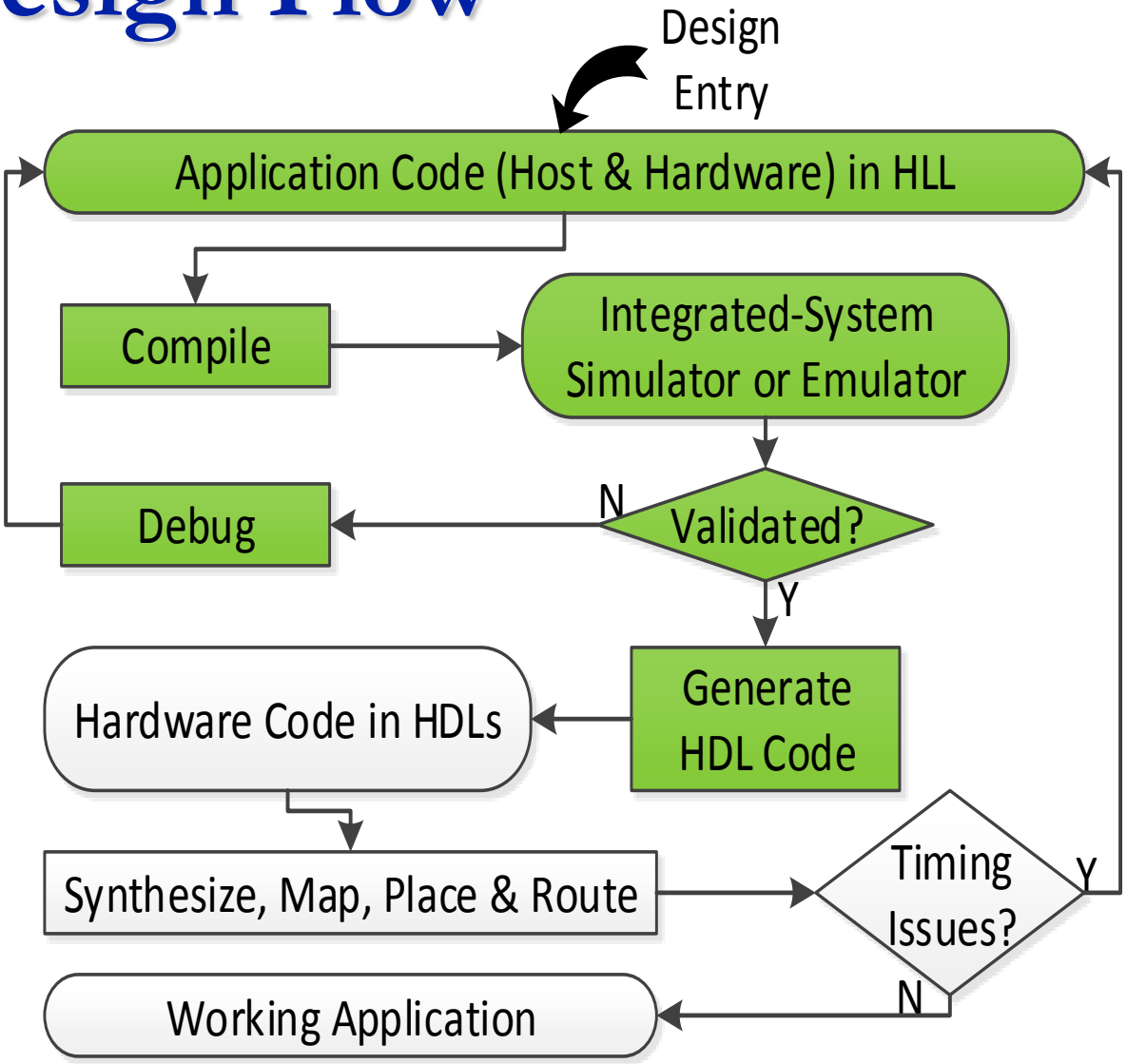
```
#include "Ht.h"
#include "PersAdd.h"

void CPersAdd::PersAdd()
{
    S_op1Mem.read_addr(PR_htId); S_op2Mem.read_addr(PR_htId);
    if (PR_htValid) {
        switch (PR_htInst) {
        case ADD_LD1: {
            if (ReadMemBusy()) { HtRetry(); break; }
            MemAddr_t memRdAddr = S_op1Addr + (P_vecIdx << 3);
            ReadMem_op1Mem(memRdAddr, PR_htId);
            HtContinue(ADD_LD2);
        }
        break;
        case ADD_LD2: {
            if (ReadMemBusy()) { HtRetry(); break; }
            MemAddr_t memRdAddr = S_op2Addr + (P_vecIdx << 3);
            ReadMem_op2Mem(memRdAddr, PR_htId);
            ReadMemPause(ADD_ST);
        }
        break;
        case ADD_ST: {
            if (WriteMemBusy()) { HtRetry(); break; }
            P_result = (uint64_t)(S_op1Mem.read_mem() + S_op2Mem.read_mem());
            MemAddr_t memWrAddr = S_resAddr + (P_vecIdx << 3);
            WriteMem(memWrAddr, P_result);
            WriteMemPause(ADD_RTN);
        }
        break;
        case ADD_RTN: {
            if (SendReturnBusy_add()) { HtRetry(); break; }
            SendReturn_add(P_result);
        }
        break;
        default: assert(0);
        }
    }
}
```

Personality with HT

High-level Design Flow

- HT bases on SystemC
  - Cycle-accurate simulation
  - Weed out inter-cycle bugs
- OpenCL
  - Behavioral emulation
  - Tools handle timing issues



Potential Course Project

- Goals
  - Learn HT programming and develop application for Merlin board using HMC device
  - Get involved with CMC research
- Tasks
  - Start with simple code examples
    - Hello world, Vadd, Bucket Sort, etc.
  - Pick an application
    - Smith-Waterman, Radix Sort, Improved bucket sort, DRE, etc.
    - Or bring your own application
  - Use SystemC simulation to validate design
- Bonus task
  - Code review with Gongyu Wang
  - Compile to bit file and run application on Merlin board