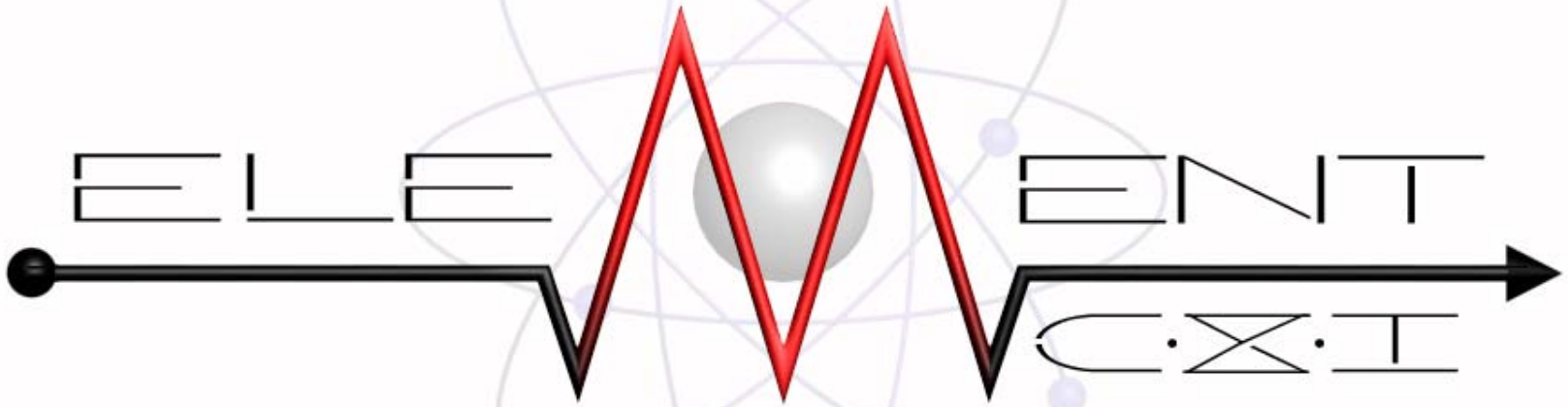
Resilient



A Co-Operative Algorithm Design Framework

Kota Bhaskar

Puneet Maheshwari

NASCUG 2005

Presentation V 2.0

Computing

elementcxi.com

Objective

- Demonstrate
  - Conventional Algorithm design flow
  - The Broken Bridges from “pseudo” Algorithm design to Implementation
- Define the Bridges
- Use of SPW (*Signal Processing Workbench*) to capture Algorithms
- Demonstrate
  - “Implement able Algorithm Design”
  - Performance Analysis of Algorithms on Architectures
  - How SPW and SystemC could be used as an example

Current Methodology

Architecture

Implementation

C/C++

Matlab

SystemC

Word

Architecture Independent

Design Partition

Hardware

Software

HW Architecture

Chip Verification

Chip RTL

Chip Assembly

Chip Simulation Model

FPGA RTL NOISE

90nm Design

Hardware Architecture

Software Implement

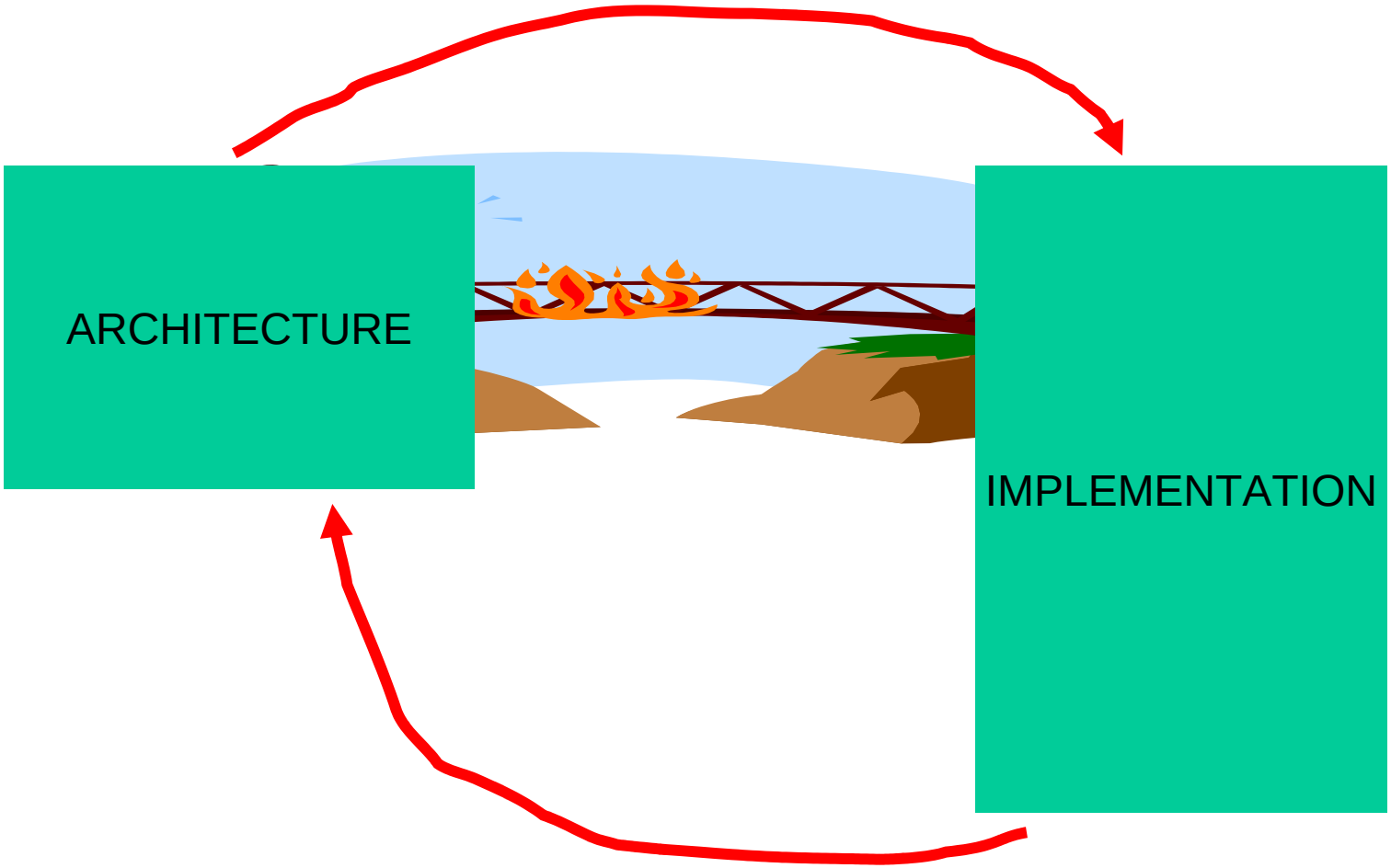
Simulation Platform

10 Cycles/PS Simulation

1M Cycles/PS Simulation

Software Optimize

BROKEN BRIDGE

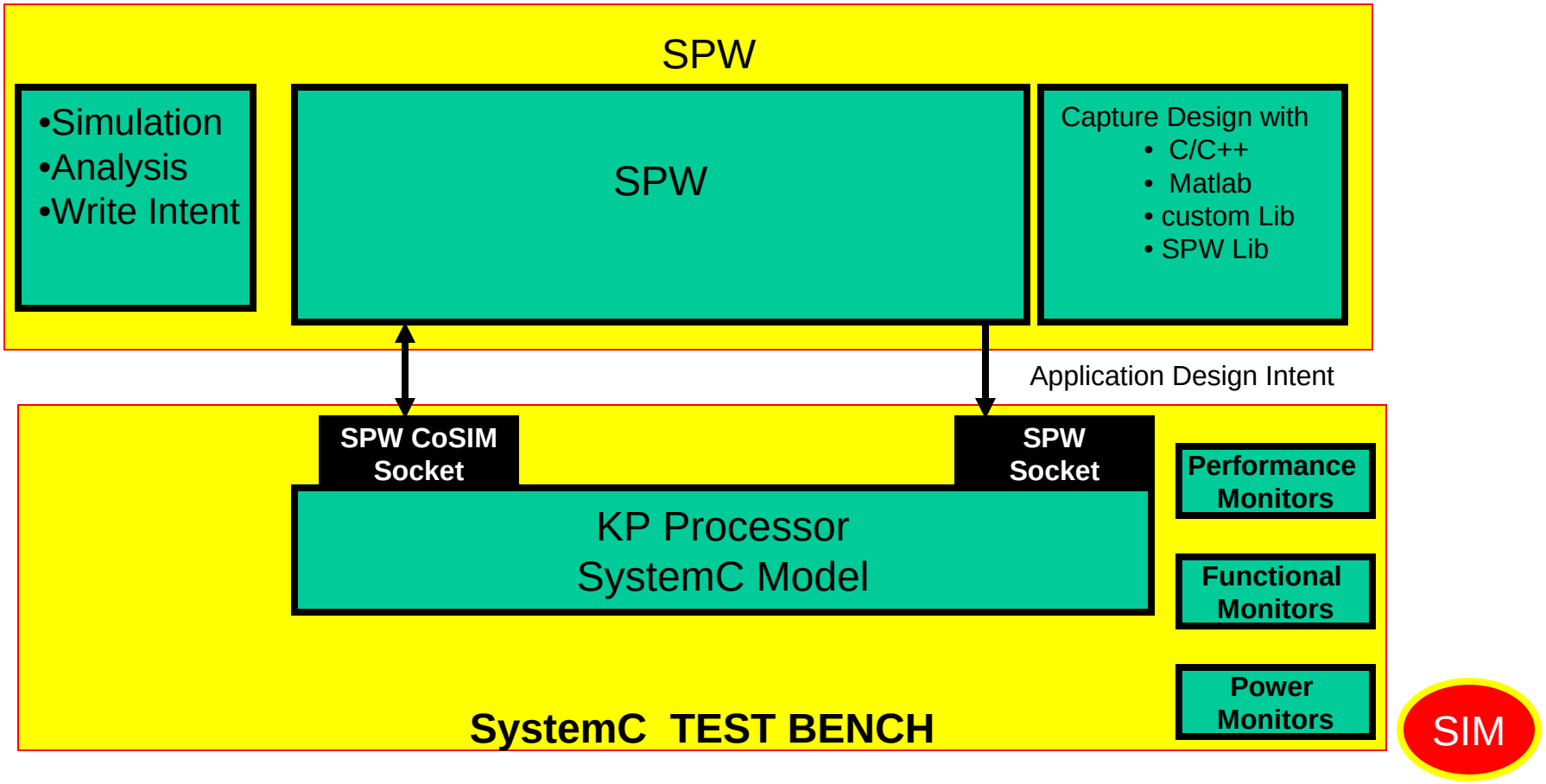


Design Bridges

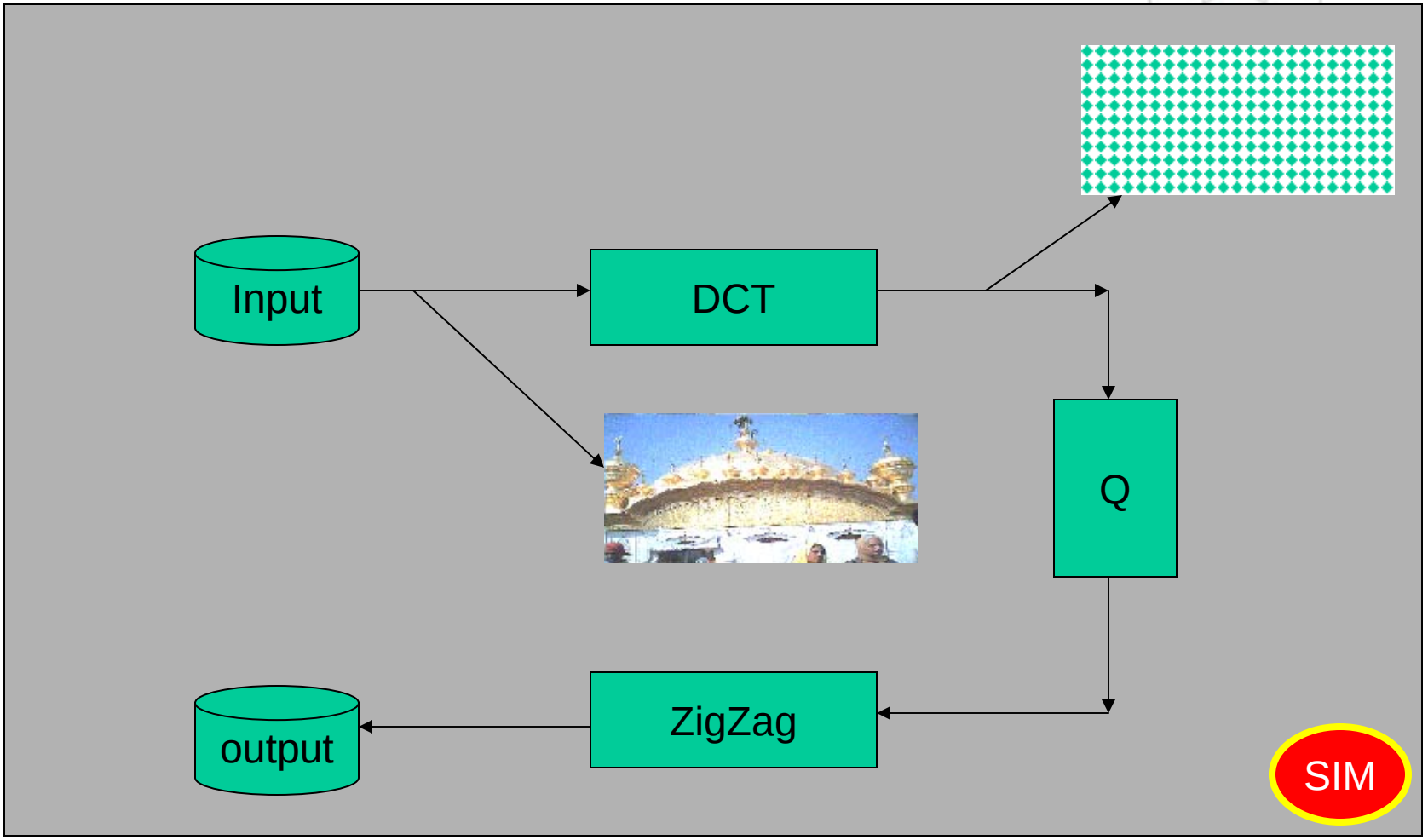
- **Architecture Aware Design**
- **Executable Architecture Specification**
- **Concurrent Design**

- **Architecture Aware Design**
  - **Platform Based Design**
    - Availability of a high Speed Simulation Model of the Platform
    - Platform Specific Design entry Tools for the Architect
    - Capability for Architecture Stage “What IF” Analysis on the Target Platform with :
      - Query on Functionality
      - Query on Performance
      - Query on Power
      - Query on Size
  - **Applications Driven Architecture Sign Off**

Design Tools Flow



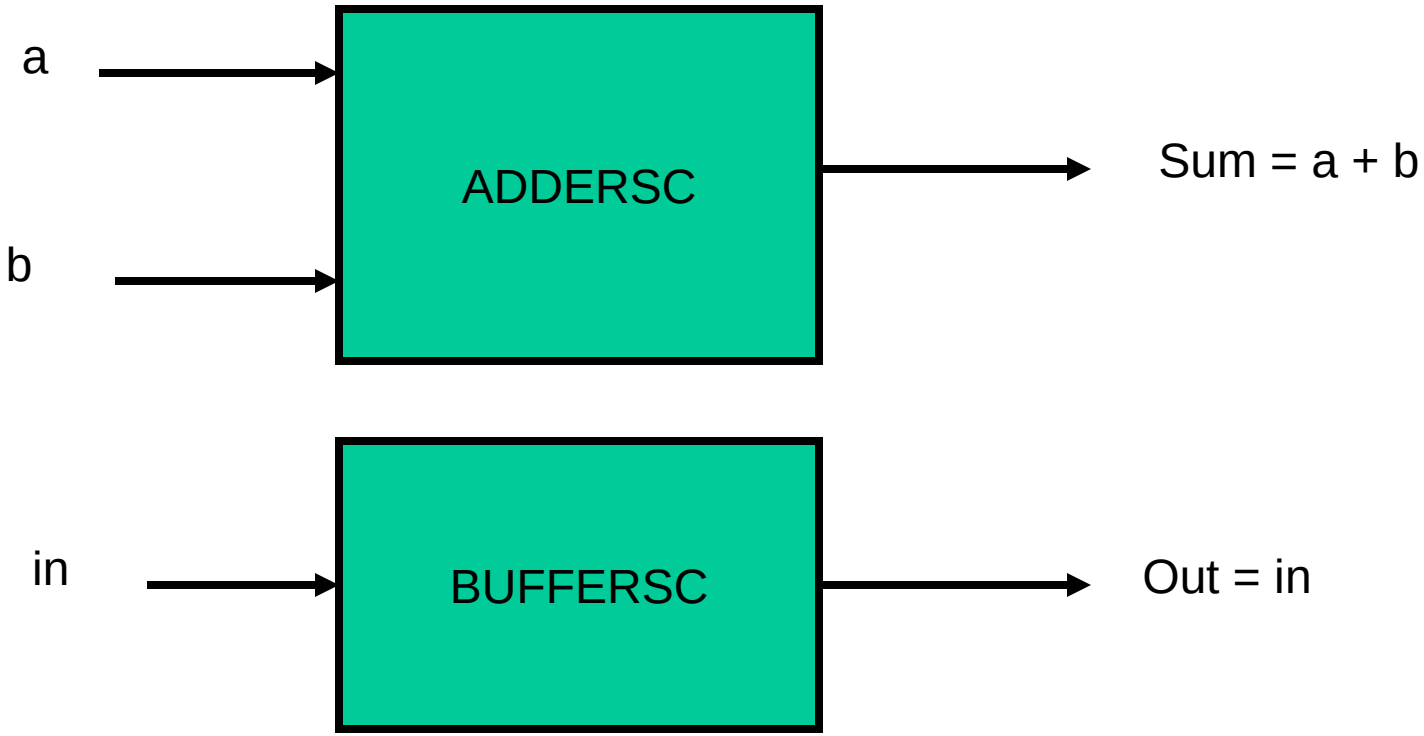
EXAMPLE



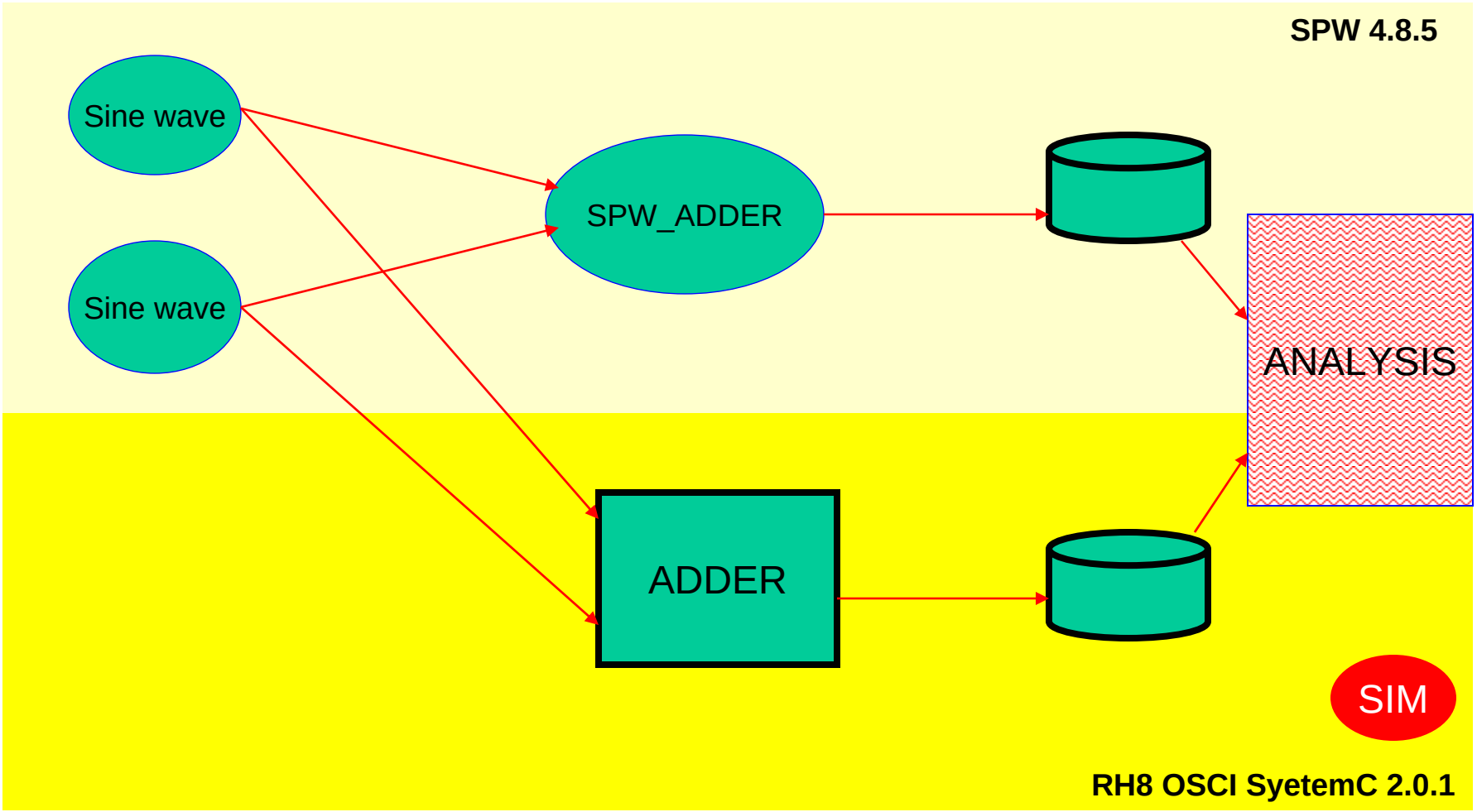
OBJECTIVE

- Demonstrate the interface between SystemC and SPW

Demo Virtual Model



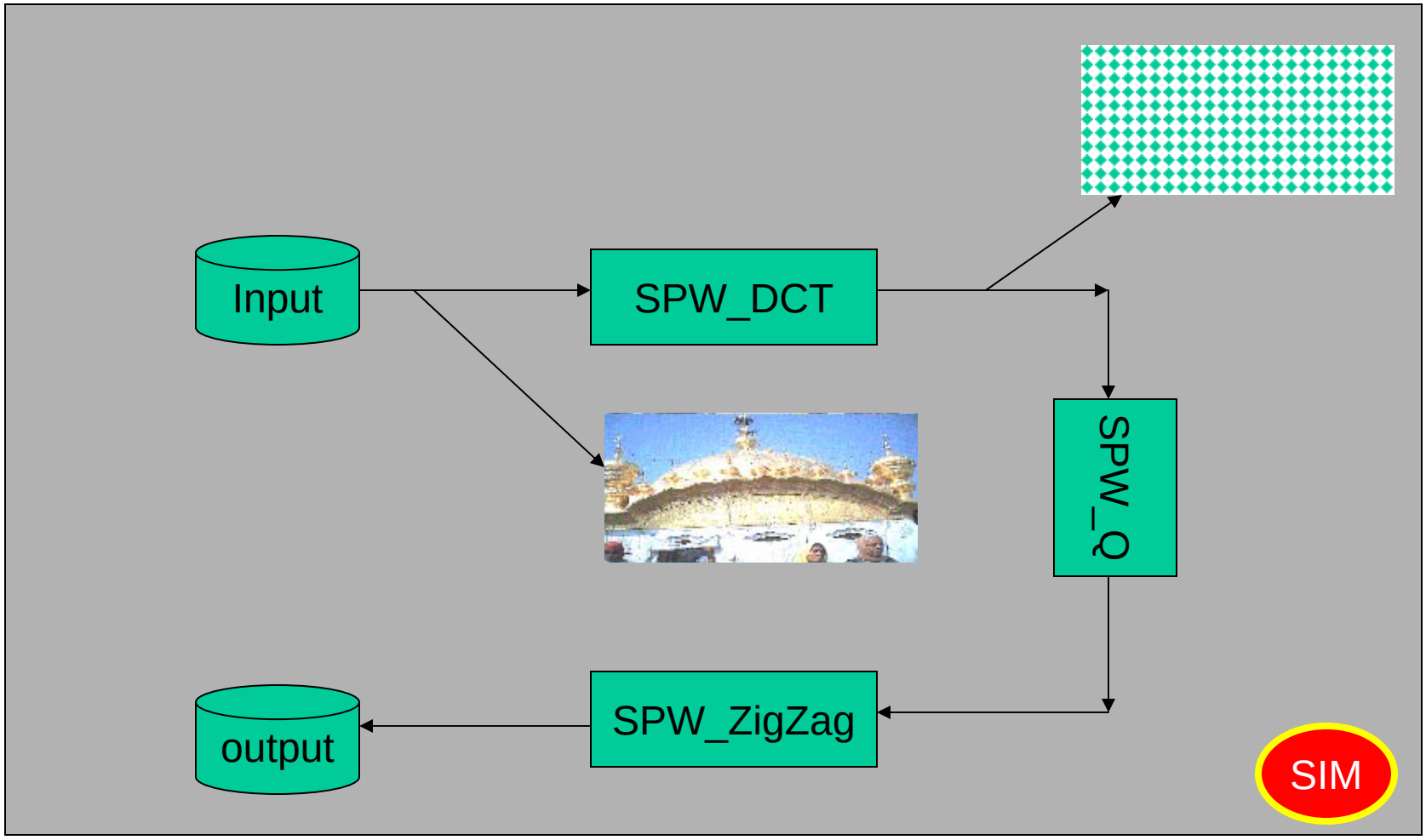
DEMO TESTBENCH



SPW Virtual Model Interface

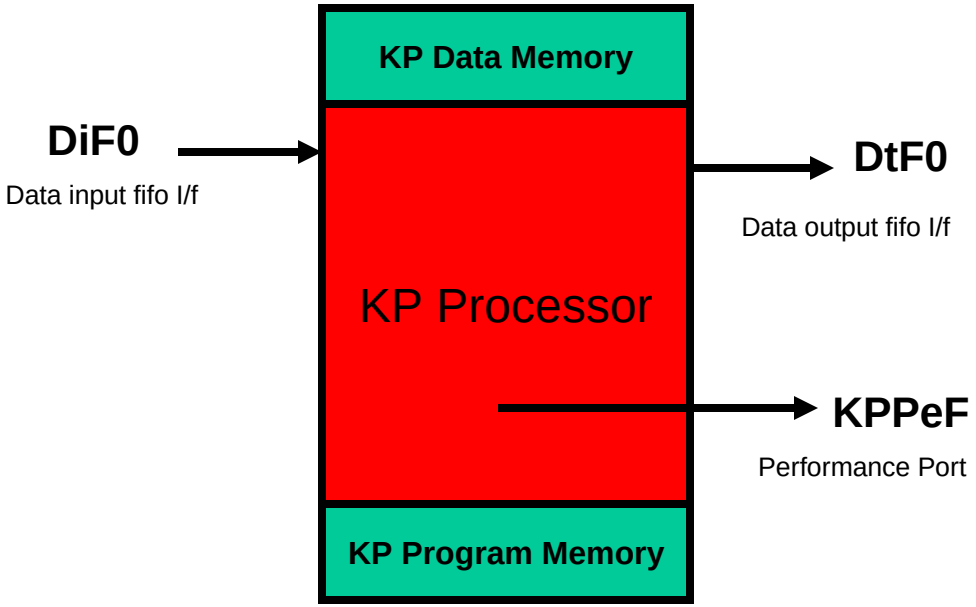
- PORTS
  - Inputs(sc_in)
  - Outputs(sc_out)
  - NO BIDIRECTIOANL PORTS
  - EXTERNAL CLOCKS
  - Sensitive to absolute Port names
- System C library Support
  - OSCI V 2.0.1
- Files Required
  - systemcapp.h
  - Systemcapp.o
  - Systemc.h
  - Spwsystemc_interface directory
- Module
  - Sensitive to absolute module name
- Remote simulation
  - Possible

“Pseudo” Simulation



KP Processor Architecture

- Floating point Data Flow Processor
- Memory Mapped input
- Memory Mapped Output
- Kahn Dataflow model
- Instructions Supported
  - MAD – Multiply and Add
  - SUM – Add two values
  - SUB – Subtract two values
  - BFF – Butterfly operation
  - TPS - Transpose
- Instrumented TLM for the processor
- Instruction Memory
- Intelligent Pipeline controller
- Transposable Register file – 256x Real
- Data memory – 256x Real
- GPR – 26x Real



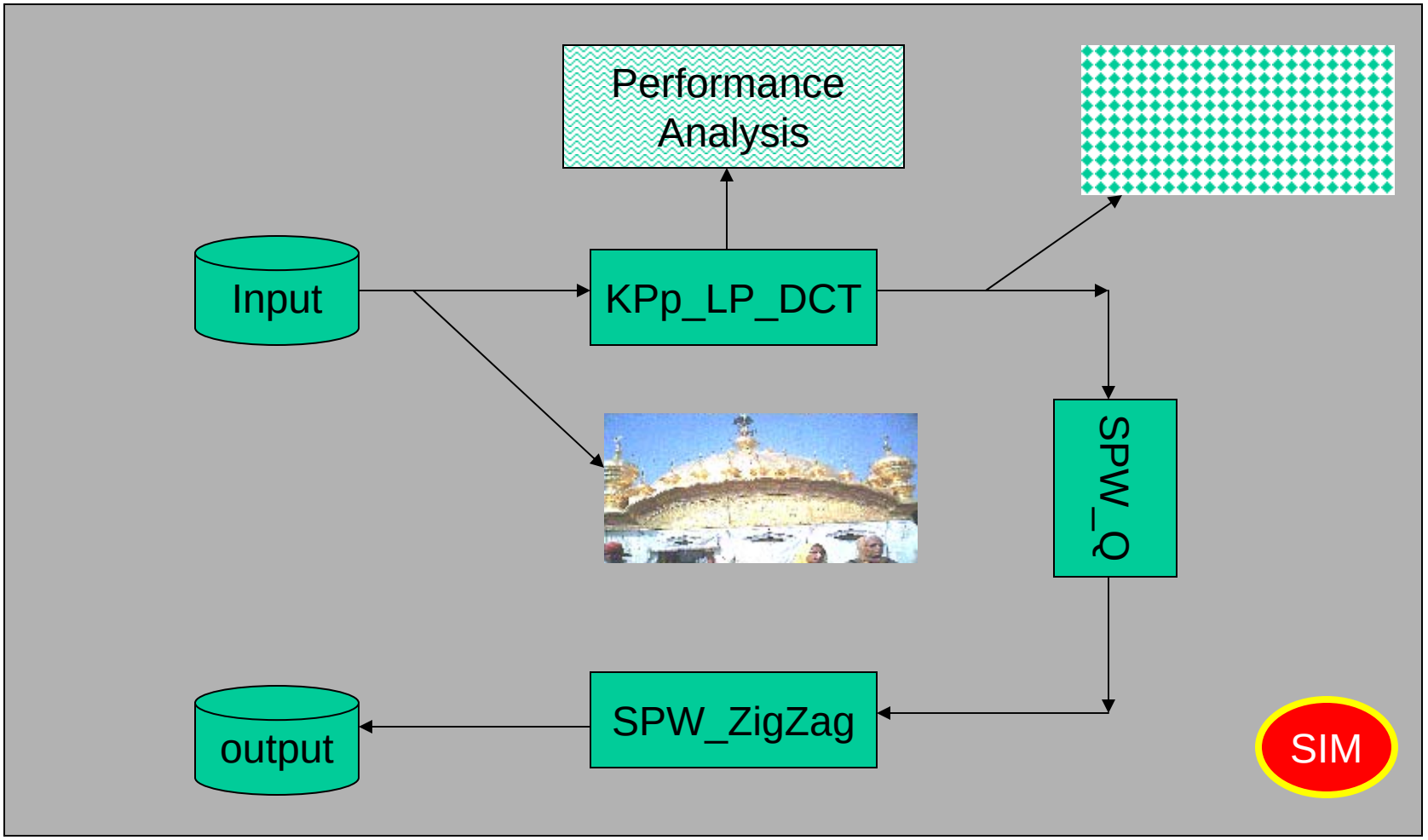
SIM

Processor

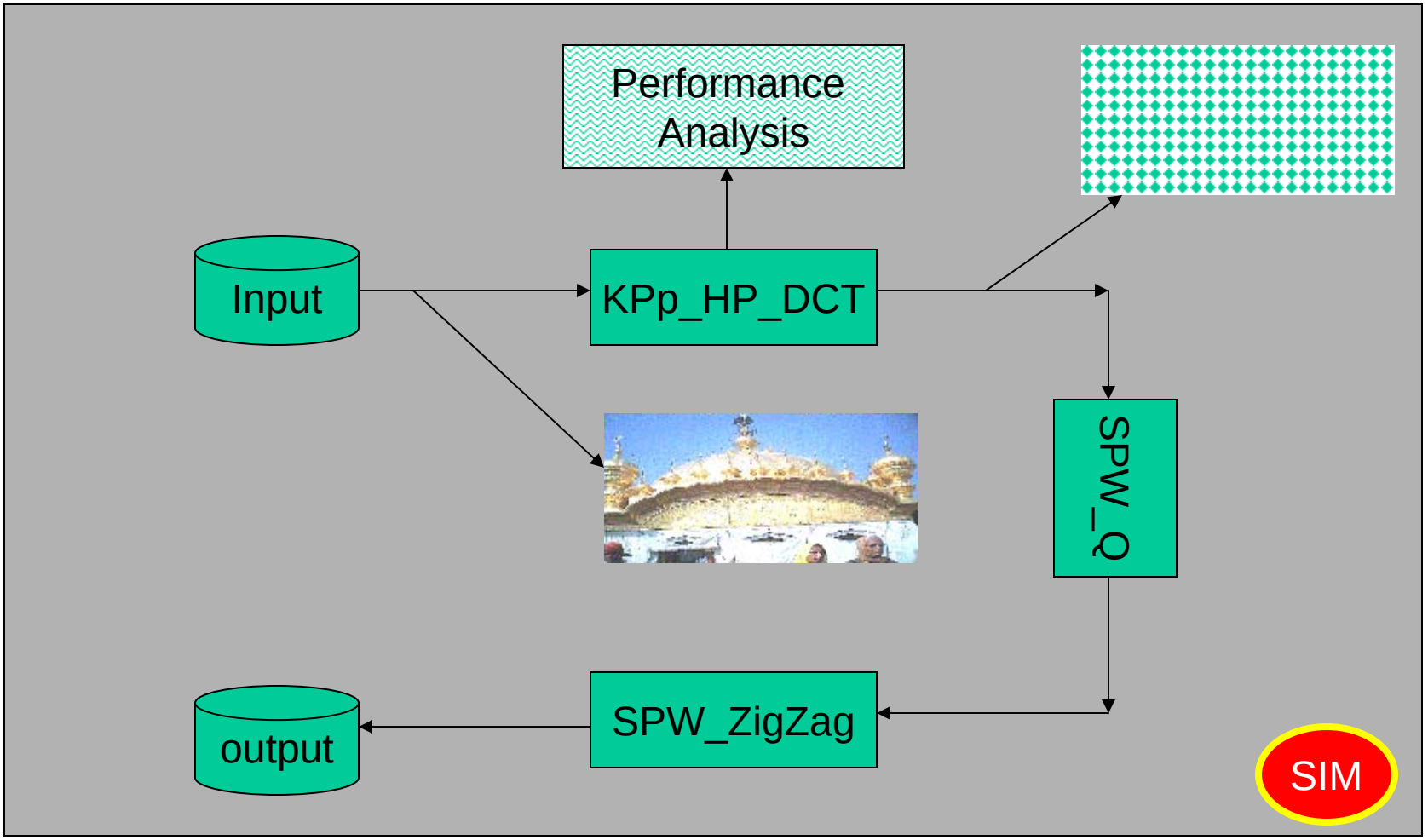
```
#include "systemc.h"
#include "processor.h"
#undef sign
SC_MODULE (my_top) {
  sc_in<double> in;
  sc_out<double> out;
  processor P;
  sc_fifo<double> in_mem;
  sc_fifo<double> out_mem;

  void peripheral() {
        static int ignore=1;
        double input;
        int tmp;
        tmp=(int)in.read();
        input=tmp;
        in_mem.write(input);
        out.write(out_mem.read());
        }
```

```
  SC_CTOR(my_top) : out("out"),
                    in("in"),
                    P("my_processor"),
                    in_mem(65536),
                    out_mem(65536) {

    for (int i =0; i<64; ++i) {
                    out_mem.write(0);
    }

    P.in(in_mem);
    P.out(out_mem);

    SC_METHOD (peripheral);
    sensitive<<in;
    dont_initialize();
  }
};
```

KP processor Enabled



KP processor Enabled



Q&A

Contact

Puneet Maheshwari

Kota Bhaskar

Kota.bhaskar@elementcxi.com