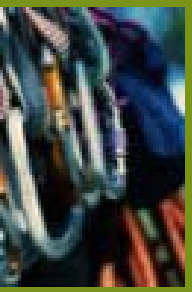


Asynchronous Interaction with Software Components from SystemC

System Verification Research Lab

Clarkson UNIVERSITY defy convention

Research Team

Clarkson UNIVERSITY defy convention

- Timothy C. Fanelli
  - PhD Student, Clarkson University
    Electrical and Computer Engineering
  - Staff Software Engineer, IBM Software Group
- Dr. Abul Khondker
  - Associate Professor, Clarkson University
    Electrical and Computer Engineering
- Dr. Robert Meyer
  - Associate Professor, Clarkson University
    Electrical and Computer Engineering

Background

Clarkson UNIVERSITY defy convention

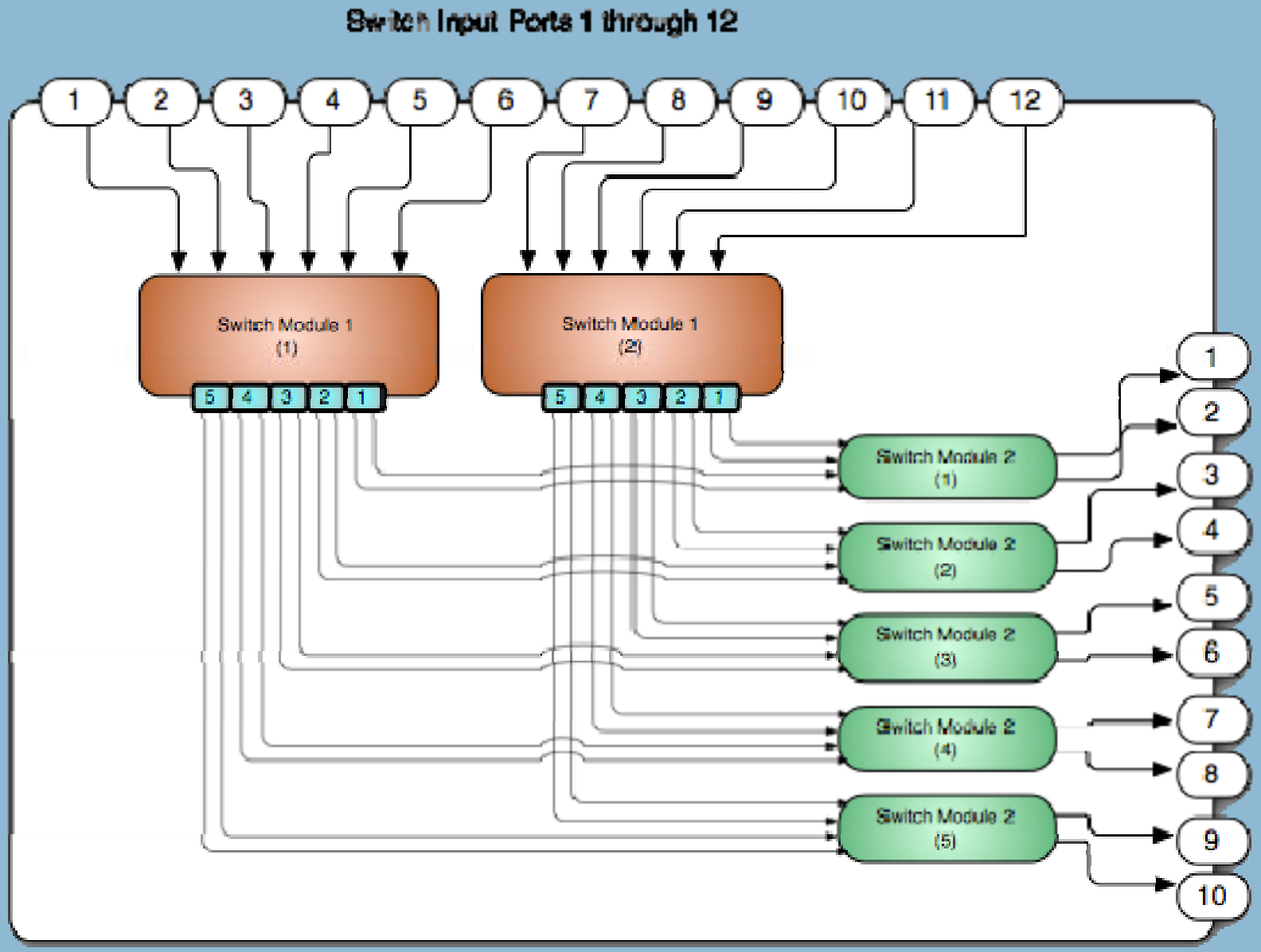
- Project based on “High-speed Packet Router Development in SystemC”
  - Presented to NASCUG 11, San Francisco, CA on 7/27/2009 by W. Gnadt, J. Casper, J. Kanyok of Lockheed Martin
- **Original Goal**: Duplicate and investigate results of the Corner Turn Device by changing FIFO Depth.

Packet Switch Overview

Clarkson UNIVERSITY defy convention

- A simplification of the “Corner Turn Device,” with:
  - 12 inputs
  - 5 packet types
  - 10 output channels

Switch Architecture



Verification Techniques

Clarkson UNIVERSITY defy convention

- Develop a test bench with SystemC modules that simulate packet sources and sinks
  - Run as part of the simulation kernel
  - Packets track sc_time at each stop during transmission
  - Outputs CSV format at sink
- Very fast and efficient
- Allowed us to verify LHM's results quickly

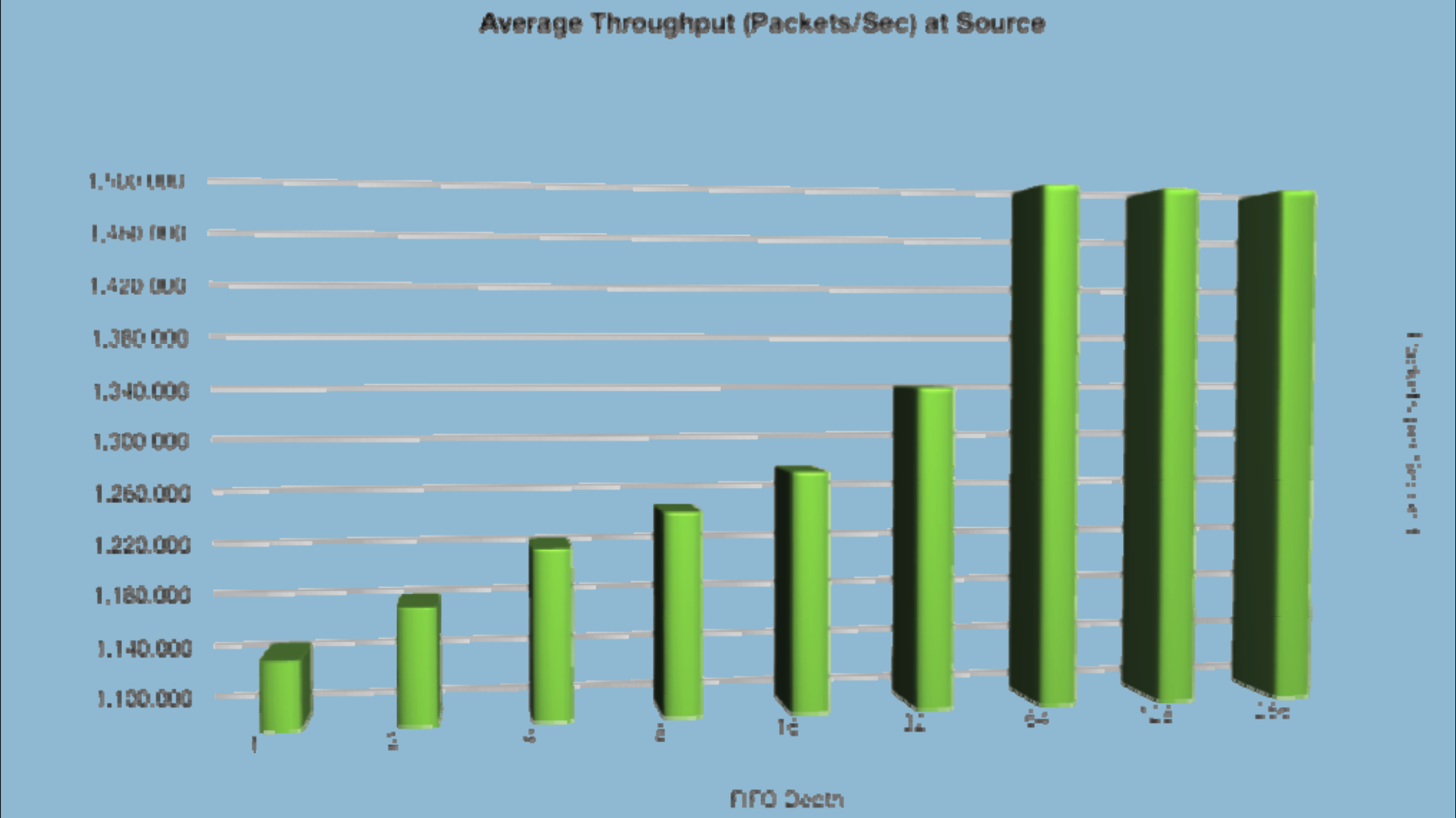
Results

Clarkson UNIVERSITY defy convention

- Each packet target generates a CSV representing all packets delivered to it
- 10 output files are merged and analyzed. 10,000 packets total.

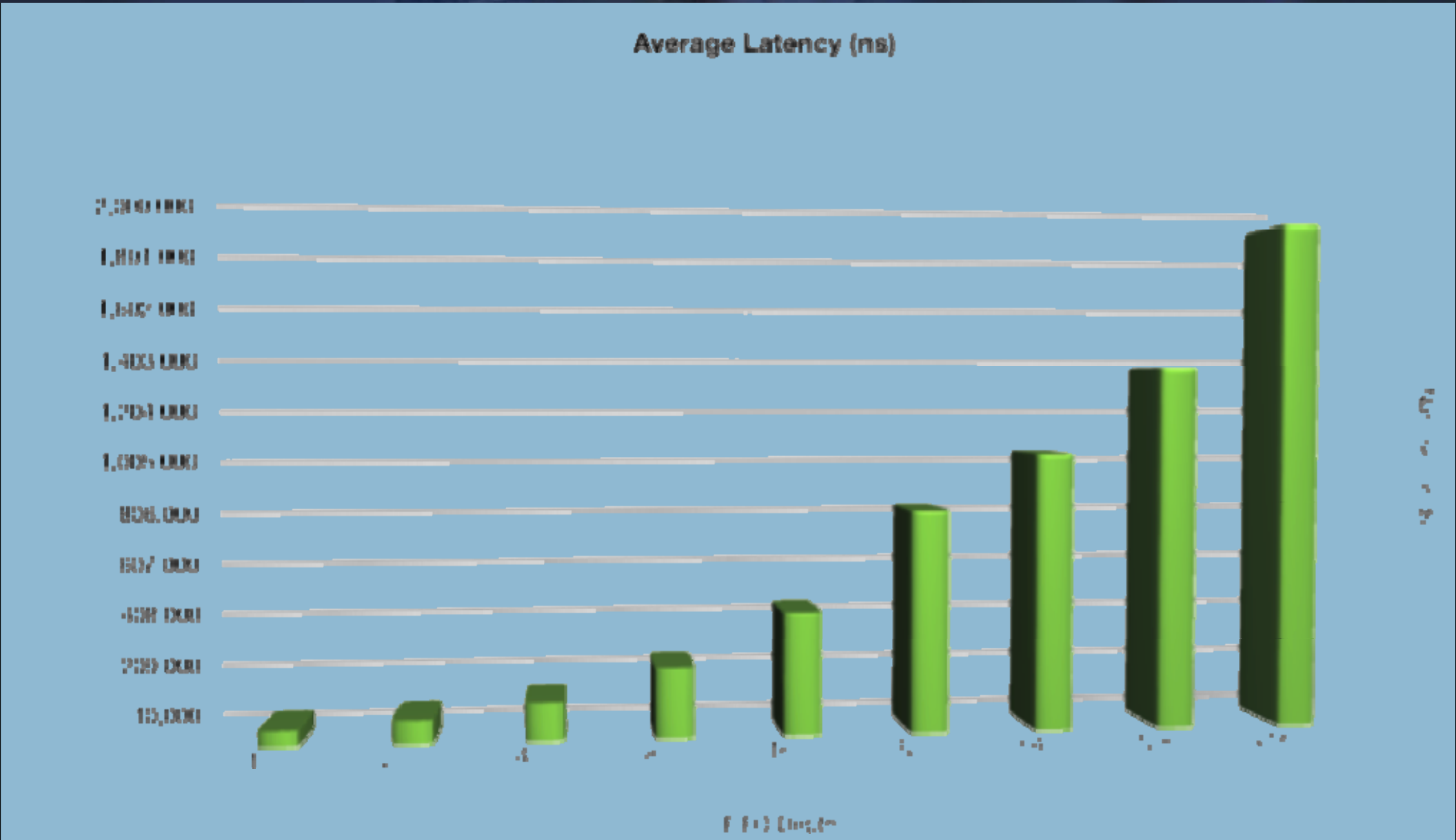
Processor Name	Packet ID	Packet Type	Created Time	Sent to Switch	Received at SM1	Received at SM2	Received at Target	Initiator to SM1 Latency	SM1 to SM2 Latency	SM2 to Target Latency	Total Latency
type2_2	PACKET 9998	2	4.57E+07	4.58E+07	4.60E+07	4.60E+07	4.60E+07	227235	9190	11282	247707
type0_2	PACKET 9996	0	4.57E+07	4.57E+07	4.60E+07	4.60E+07	4.60E+07	227236	9193	11268	247697
type2_1	PACKET 9994	2	4.57E+07	4.57E+07	4.60E+07	4.60E+07	4.60E+07	227211	12538	11272	251021
type2_2	PACKET 9992	2	4.57E+07	4.57E+07	4.59E+07	4.60E+07	4.60E+07	227206	10865	11294	249365
type2_1	PACKET 9990	2	4.57E+07	4.57E+07	4.59E+07	4.59E+07	4.60E+07	227208	9192	11276	247676
type4_2	PACKET 9988	4	4.57E+07	4.57E+07	4.59E+07	4.59E+07	4.60E+07	227208	12539	11278	251025

Analysis


Average Throughput (Packets/Sec) at Source
1.420 000
1.380 000
1.340.000
1.300 000
1.260.000
1.220.000
1.180.000
1.140.000
1.100.000

Analysis

Clarkson UNIVERSITY
defy convention



Software Co-Verification

Clarkson UNIVERSITY defy convention

- SystemC model of the switch was designed in parallel to a software-level protocol stack
- Needed to co-verify the models
  - Striving for a *continuous integration* and *continuous test* workflow

GreenSocs' QEMU-SystemC

Clarkson UNIVERSITY defy convention

- GreenSocs' QEMU-SystemC
  - http://www.greensocs.com/en/Projects/QEMUSystemC
  - Excellent option for very complex system simulations
  - Too detailed for our target abstraction level
- Want to co-verify HW and SW at high levels of abstraction

Interacting with SystemC

Clarkson UNIVERSITY defy convention

- How can our software interact with SystemC?
- Standard techniques in software:
  - Multi-threading with synchronization
  - Multi-process with IPC
- Have seen multi-process approaches using TCP/IP communication
- Initial approach uses multi-threading using OpenMP to prove the concept

SystemC in a Thread

Clarkson UNIVERSITY defy convention

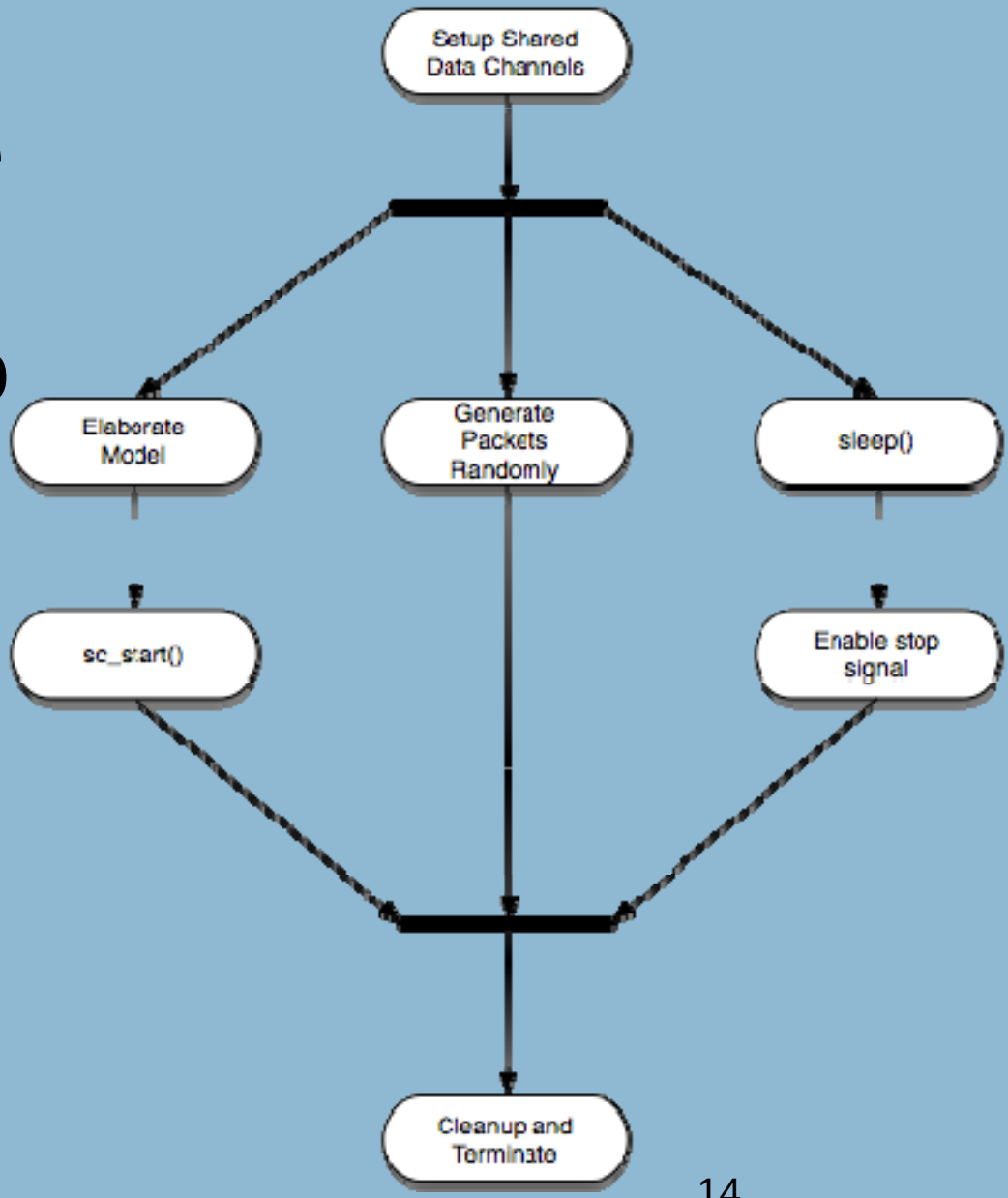
- OpenMP Constructs allow for simple multi-threading.
- 1st OMP Section performs elaboration and starts the SystemC kernel
- 2nd OMP Section, “packetSourceThread, ” generates packets to submit into switch.

```
int main()
{
...

        SharedData data(fifoDepth);
        #pragma omp parallel sections shared(data) \
              default(shared) num_threads(3)
        {
                #pragma omp section
                simulationThread(data);

                #pragma omp section
                packetSourceThread(data);

                #pragma omp section
                timeoutControlThread(data);
        }
}

void simulationThread( SharedData& sharedData )
{
// Perform Elaboration Here

sc_start();
}
```

Multithreaded Workflow

Clarkson UNIVERSITY defy convention

- How do we move Packets from the threaded SW into the simulation?
- Synchronized Channel implementations



Synchronized Channels

Clarkson UNIVERSITY defy convention

- By extending existing SC_FIFO and SC_SIGNAL implementations to include OMP locking semantics, we've:
  - Taken advantage of polymorphic behavior so that no changes are needed to the model
  - Provided a thread-safe bridge through which data may be passed asynchronously into the running simulation

synchronized_signal<T>

Clarkson UNIVERSITY defy convention

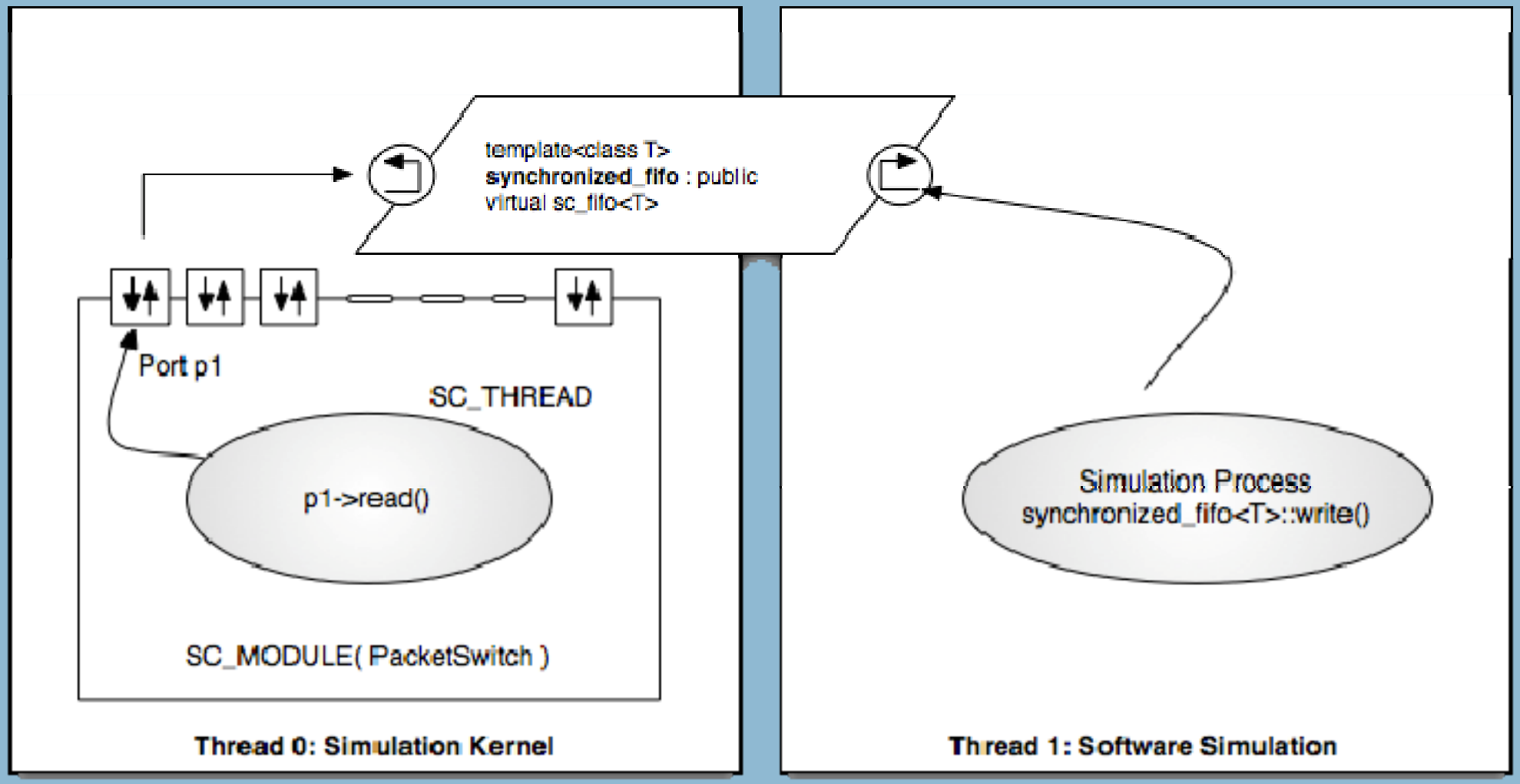
```
template<class T>
class synchronized_signal : public virtual sc_signal<T>
{
public:
      synchronized_signal();
      synchronized_signal(omp_nest_lock_t & lock);
      virtual ~synchronized_signal();

      virtual const T& read() const;
      virtual void write( const T& );

private:
      mutable omp_nest_lock_t accessLock;
};
```

Using Sync Channels

Clarkson UNIVERSITY defy convention

- One end of the channel is bound to a port on the router
- The other is un-bound in the simulation
- Software thread can call read() and write() methods like normal
  - Uses OMP locks in implementation perform synchronization

Synchronized Channel



Results

Clarkson UNIVERSITY defy convention

- Have Shown that SystemC has great potential for high-level co-verification of hybrid HW/SW systems designs
- On-going efforts:
  - Reproduce results from “Parallelizing SystemC Kernel for Faster Simulation on SMP Machines”; Ezudheen, Chandran, Chandra, Simon and Ravi, 2009
  - Develop Complete Library of Synchronized Channel Implementations
  - Provide Application-Level thread synchronization constructs in the Simulation Kernel for increased efficiency/stability.