OpenSMART: An Opensource Single-cycle Multi-hop NoC Generator

Hyoukjun Kwon and Tushar Krishna

Georgia Institute of Technology
Synergy Lab (http://synergy.ece.gatech.edu)
OpenSMART (https://tinyurl.com/Get-OpenSMART)

hyoukjun@gatech.edu

Nov 12, 2017
OpenSMART NoC

NoC generated by OpenSMART
Challenges for NoCs

- **Scalability**
  - Supporting *many-IP* heterogeneous system
  - Lower latency
  - Lower area & energy

- **Flexibility**
  - Support diverse connectivity for custom heterogeneous system
  - Support diverse latency/throughput requirements

- **Design-cost**
  - Automating the design of high-performance, low-energy NoCs
  - Lowering design/verification costs of SoCs with NoCs
OpenSMART

Low Cost  Flexibility  Scalability

User-configurable Automatic NoC Generation
High-level HW Language
Highly-modular Pre-verified Building Blocks

SMART NoC

Krishna et al, HPCA 2013
Chen et al, DATE 2013
Krishna et al, IEEE Micro Top Picks 2014

Arbitrary Topology Support
Area/power-efficient RTL Building Blocks

OpenSMART
SMART NoC

- **Single-cycle Multi-hop Asynchronous Repeated Traversal**

**SMART**: achieve the performance of *dedicated* connections over a network of *shared* links.
Features of SMART

• Low latency network
  - Dynamic bypass of intermediate routers between any two routers
  - Limit: \( \text{HPC}_{\text{max}} \) (hops per cycle max), *maximum number of “hops” that the underlying wire allows the flit to traverse within a clock cycle*

• Separate control path
  - \( \text{HPC}_{\text{max}} \) bits from every router along each direction
  - Arbitration of multiple bypass requests on the same link
  - No ACK required
How to Get the Source Code

• Go to Synergy lab homepage
  (synergy.ece.gatech.edu)
How to Get the Source Code

• In the released tools tap, click OPENSMART
How to Get the Source Code

- You will be forwarded to access request form page.
- Please fill and submit the form, then you will get a link to OpenSMART repository.
How to Get the Source Code

- Using the link, you can access to the repository
Source Tree (Under Backend/BSV)

• **Frontend:** Configuration Parser (under development)

• **Backend/BSV:** BSV implementation (Main files)
  - **src:** Building blocks
    - `Network.bsv`: Connectivity configuration (default: Mesh)
    - `Types/Types.bsv`: Topology (Number of routers), VC, Routing algorithm, SMART (HPCmax) configuration
  - **lib:** Fundamental BSV libraries (FIFOs and CReg)
  - **testbenches:** Include synthetic traffic-based simulation

• **Backend/Chisel:** Chisel implementation (Router only)
OpenSMART Design Flow

- Topology
  - Bandwidth
  - VC
  - Routing
  ...

- Configuration
  - User Specification

- OpenSMART Front-end

- Building Block Library (RTL)
  - Input Unit
  - Output Unit
  - SMART Unit
  - Switch Unit
  ...

- BSV/Chisel Compiler

- ASIC/FPGA Synthesis Tool

- HPCmax Analyzer

- External Tool Chains

- Verilog Files
How to Specify a topology

• In Backend/BSV/src/Network.bsv

... 
64   for(Integer i=0; i < meshHeight; i++) begin
65       for(Integer j=0; j < meshWidth -1; j++) begin
66           mkConnection(routers[i][j].dataLinks[East].getFlit, 
                        routers[i][j+1].dataLinks[West].putFlit)
67           mkConnection(routers[i][j].controlLinks[East].putCredit, 
                        routers[i][j+1].controlLinks[West].getCredit)
68       end
69   end

Can change connectivity using “mkConnection” with different routers/links

Automation of this process is under development
How to Configure OpenSMART

- In Backend/BSV/Types/types.bsv

1. `typedef 100000 Benchmark Cycle

2. `typedef 32 DataSz

3. `typedef 4 NumFlitsPerDataMessage

4. `typedef 6 UserHPCMax

5. `typedef 8 MeshWidth

6. `typedef 8 MeshHeight

7. `typedef 4 NumUserVCs

8. `currentRoutingAlgorithm = XY_;
OpenSMART Design Flow

- Topology
  - Bandwidth
  - VC
  - Routing
  ...

- Configuration
  - User Specification

- Building Block Library (RTL)
  - Input Unit
  - Output Unit
  - SMART Unit
  - Switch Unit

- OpenSMART Front-end

- ASIC/FPGA Synthesis Tool
- BSV/Chisel Compiler

- HPCmax Analyzer

- External Tool Chains
- Verilog Files
OpenSMART Building Blocks

Input Unit

Input buffer + Input VC arbitration

Output Unit

Output VC selection + Output port arbitration + Credit management

Switch Unit

Switching (via crossbar) + Routing calculation

SMART Unit

SSR communication & Arbitration + Bypass flag
OpenSMART Router

Input Unit

Arbiter

Flit

Header

Flit Size

Input Buffers

Number of VCs/VC Depth

Flit Data

Flit

Header

Flit Data

Outgoing Flits

Incoming Flits
OpenSMART Router

Incoming Flits

Output Port Request

Output Port Grant

Arbiter

nextVC

VC Selector

nextVC

VC queue

Credit Manager

hasCredit

Output Unit

Arbiter

Outgoing Flits

Credit

VC
OpenSMART Router

Incoming Flits

Switching Unit

Routing Unit

Crossbar

Routing Algorithm

Outgoing Flits

Outgoing Flits from Input Units
Prioritization by distance

-> SSR from a nearer router gets the higher priority
(Local (distance = 0) has the highest priority)
Walk-through Example

• Router r4 sends a flit to router r7
• Router r5 sends a flit to router r7

HPCmax = 3

SSR (SMART Setup Request)

Cycle 0: SSR Send
Cycle 1: Multi-hop Bypass

SMART Arbiter

Priority

Incoming SSRs

Distance 0

Distance 1

Distance 2

Distance 3

From: r4

From: r5

Winner

SSR From Local Router

SMART Unit in r5

Bypass Flag
OpenSMART Design Flow

Topology
- Bandwidth
- VC
- Routing
...

Configuration

User Specification

OpenSMART

Building Block Library (RTL)
- Input Unit
- Output Unit
- SMART Unit
- Switch Unit
...

OpenSMART Front-end

BSV/Chisel Compiler

External Tool Chains

Verilog Files

ASIC/FPGA Synthesis Tool

HPCmax Analyzer
How to Run OpenSMART

• In Backend/BSV/

> ./OpenSMART –c Compile synthetic traffic-based Simulation

> ./OpenSMART –r Run compiled simulation

> ./OpenSMART –v Generate Verilog code

> ./OpenSMART –clean Clean up build files
How to Run OpenSMART

- Simulation Compilation Print-out Messages

Bluesim object created: build/bdir/mkInputVC Arbiter.{h,o}
Bluesim object created: build/bdir/module_dir2Idx.{h,o}
Bluesim object created: build/bdir/module_smartOutportComputeFunc.{h,o}
Bluesim object created: build/bdir/mkRoutingUnit.{h,o}
Bluesim object created: build/bdir/mkStatLogger.{h,o}
Bluesim object created: build/bdir/mkOutPortArbiter.{h,o}
Bluesim object created: build/bdir/mkOutputUnit.{h,o}
Bluesim object created: build/bdir/mkSmartVCA llocUnit.{h,o}
Bluesim object created: build/bdir/mkReverseCreditUnit.{h,o}
Bluesim object created: build/bdir/mkCrossbarSwitch.{h,o}
Bluesim object created: build/bdir/mkSwitchAllocUnit.{h,o}
Bluesim object created: build/bdir/mkUniformRandom.{h,o}
Bluesim object created: build/bdir/mkCrossbarBuffer.{h,o}
Bluesim object created: build/bdir/mkTrafficGeneratorBuffer.{h,o}
Bluesim object created: build/bdir/mkTrafficGeneratorUnit.{h,o}
Bluesim object created: build/bdir/mkInputUnit.{h,o}
Bluesim object created: build/bdir/mkBaselineRouter.{h,o}
Bluesim object created: build/bdir/mkNetwork.{h,o}
Bluesim object created: build/bdir/mkTestBench.{h,o}
Bluesim object created: build/bdir/model_mkTestBench.{h,o}
Simulation shared library created: sim.so
Simulation executable created: sim
How to Run OpenSMART

• Simulation Print-out Messages

![Simulation Print-out Messages]

Simulation Ticks: every 10,000 cycles
Indicates if the simulation is alive or not
How to Run OpenSMART

- Simulation Print-out Messages

Send/Receive counts for every router
Summary of the total statistics
How to Run OpenSMART

• Generating Verilog files

```
compiling ./BaselineRouter.bsv
code generation for mkBaselineRouter starts
Verilog file created: mkBaselineRouter.v

compiling ./module_getFlitTypes.bsv
code generation for module_getFlitTypes starts
Verilog file created: module_getFlitTypes.v

compiling ./module_getRB.bsv
code generation for module_getRB starts
Verilog file created: module_getRB.v

compiling ./Network.bsv
code generation for mkNetwork starts
Verilog file created: mkNetwork.v

compiling ./TestBench.bsv
code generation for mkTestBench starts
Verilog file created: mkTestBench.v
All packages are up to date.
```

Similar print-out messages as simulation compilation
How to Run OpenSMART

- Generating Verilog files

Verilog files are generated in ./Verilog
OpenSMART

Topology
- Bandwidth
- VC
- Routing
...

Building Block
- Input Unit
- Output Unit
- SMART Unit
- Switch Unit
...

OpenSMART Front-end

BSV/Chisel Compiler

Verilog Files

Thank you!

OpenSMART (https://tinyurl.com/Get-OpenSMART)