VisualSim Training

MIRABILIS design

Planning

Goals & Requirements	Tasks or Operations	Workloads or Traffic	Hardware or Platform	Collaboration & Sharing	Data Management	Support	Look & feel

Modeling

Parameters/ Variables/ DS/ Fields	Flow- System, Beh-Arch, Data flow	Performance Power Functionality	Hierarchical/ Class/ DI Blocks	Traffic/ Results	Hardware/ Software/ Network	Abstraction	Algorithms	Task Graph	Checking

Simulation and Analysis

Types of Studies	Parameter Ranges	Graphical/ Batch/ Multi-Core	Debugging/ Verification	Requirements	Post-Simulation	Types of Plots	Advanced Studies	Accuracy	Compare

Advanced Features

Failure	Integration	Generation	Encryption				

Agenda- Part 2: Modeling

- Basic Components for hardware modeling
- Performance, Power and Functionality
- Performance and power Metrics
- Library components and important parameters which affect performance

Performance, Power and Functionality

- VisualSim provides an integrated solution for performance, power and functional modeling
- The details regarding Power modeling have been dealt in a separate presentation
- The functional modeling aspect is covered in modeling the task graphs section
- The user can also perform functional verification by observing the input, output values, checking for their correctness
- Custom algorithms or functional flows can also be define dusing the Script Block
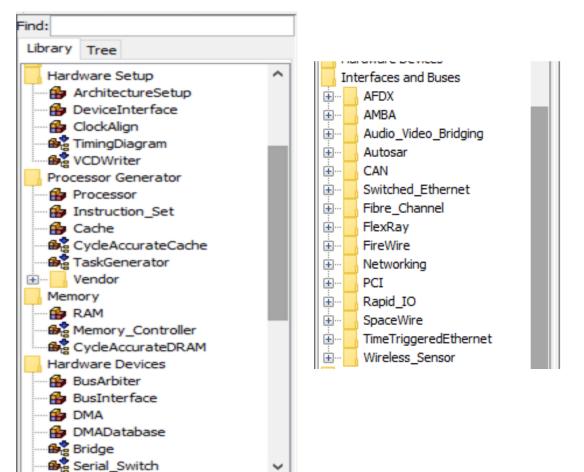
Basic Components for Hardware Modeling

Architecture Library Overview

- Generate architecture with parameterized blocks

- Define hardware and software components

- Create proposed or derivative architectures in few minutes

- Rapidly define application flow diagram and behavior

- Optimize architecture and functionality mapping combination
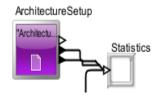
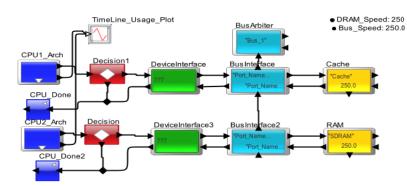*<u>Hardware Modeling Library</u>*

Basic Components to define

1. Hardware modeling requires Architecture_Setup block - has the routing information of all hardware blocks in the model

2. If using Traffic or other custom block to connect to a Bus, use the Device_Interface block in front of the Bus port.
   ◦ Also true for AXI, PCIe and NoC
   ◦ Not required for Standard Blocks Like Processor
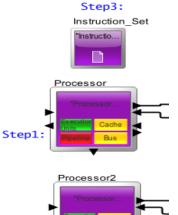
3. Processor block requires Instruction_Set

ArchitectureSetup Block gives out statistics for all the hardware blocks in the model

Use of Device Interface

Processor Blocks with InstructionSet

Architecture Setup

All Bus and Hardware block must associate with Architecture Setup

This Block Handles

➢ Routing

➢ Plotting

➢ Statistics

➢ Debugging for all the Hardware components

ArchitectureSetup

"Architectu...

1. There can be multiple architecture setup blocks
2. Each block must have unique name

Note: this Block can be used as it is after updating the Architecture_Name field

Custom Routing Table Construction - in Architecture Setup

Format Sample:

Source_Node	Destination_Node	Hop	Source_Port
Source Processor	Destination DRAM	Next Block Port_1	Out Port bus_out

Add entries using RegEX

addToRoutingTable (Architecture_Name, Source_Name, Destination_Name, Hop_Name, Source_Port_Name)

Delete entries using RegEX

removeFromRoutingTable (Architecture_Name, Source_Name, Destination_Name, Hop_Name, Source_Port_Name)

DeviceInterface Block

DeviceInterface
"IO_Name"

- Can be used to define the Source Name, data size, command type and destination for a Master

- Can be used to define the Device name on the Slave side

- Add the Master or Slave block to the Linear Bus, Bridge and AHB Buses automatically
  - ✓ Generate Hello Messages
  - ✓ Eliminates the need for a RegEx functions or manually generate Hello message

- Map fields of other data structure formats to the corresponding fields of the Processor_DS

- Used to connect only in the presence of the Linear Bus, AHB and bridge blocks.

- Not useful with a single AXI or a single PCIe

Block_Documentation:	📝 Enter User Documentation Here
Architecture_Name:	"Architecture_1"
A_Source:	"CPU2"
A_Destination:	"Cache"
A_Command:	"A_Command"
A_Instruction:	"Fld_Name_or_String_or_None"
A_Bytes:	"A_Bytes"
A_Priority:	"Fld_Name_or_Integer"
A_Address:	"Fld_Name_or_Integer"

Commit   Add   Remove

Set field values here or keep default to use existing field values

Stochastic Components in the Library

Concept of System Resource

Concept
- Split operation into two parts
- Behavior or mapper
- Resource (similar to Server)

Blocks
- Behavior: Mapper, SoftwareMapper, DynamicMapper
- Architecture: SystemResource_Extend, SystemResource
- Notify: SystemResource_Done

Multiple concurrent requests
- Send from Mapper (Behavior) to the SystemResource with the delay information
- Can be static or dynamic reference
- Scheduler: First Come-First Serve, Round-Robin, Preemption, Non-Locking
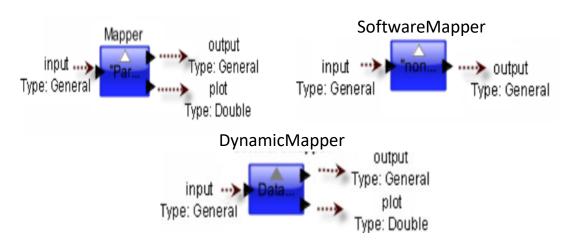
SystemResource_Done block
- Release appropriate SystemResource_Extend block by signaling the completion of an external task
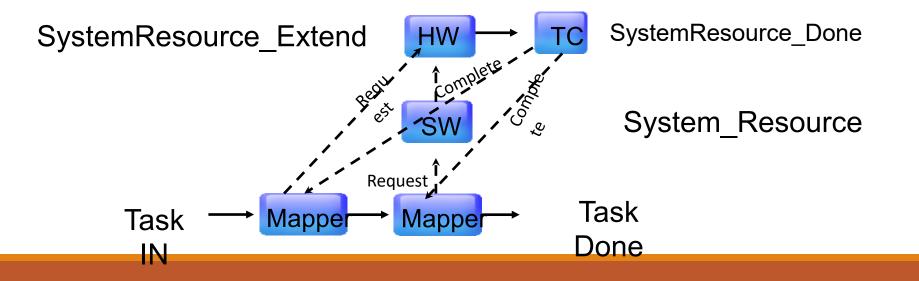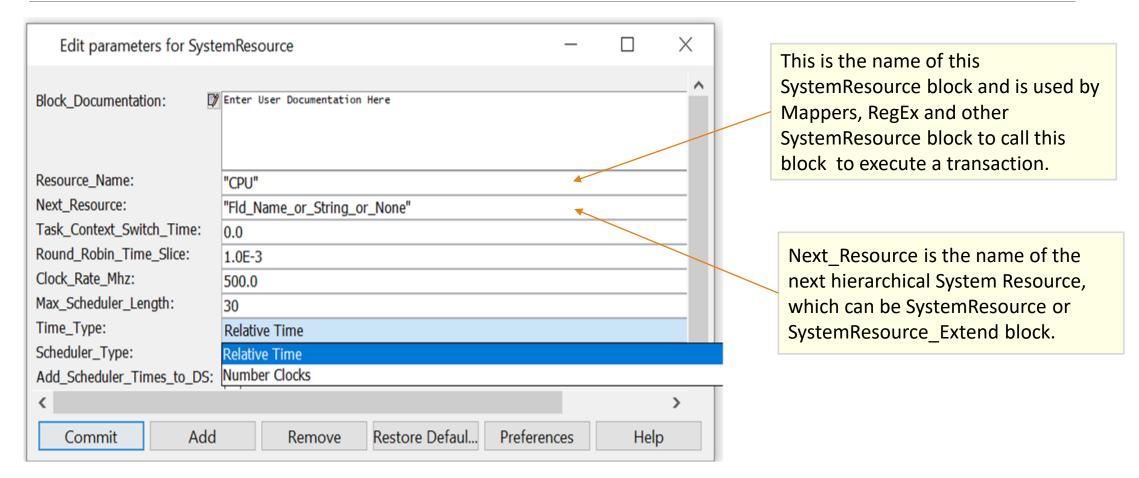

Architecture

Behavior

Mapper to System Resource

.Mapper blocks define the connectivity between the behavior flow and the architecture flow, and within the architecture flow using a named connection

• The block takes the incoming Data Structure and send it to the Scheduler virtually

• This block can send a request to either the SystemResource or SystemResource_Extend.
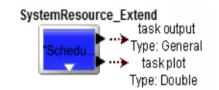
System Resource

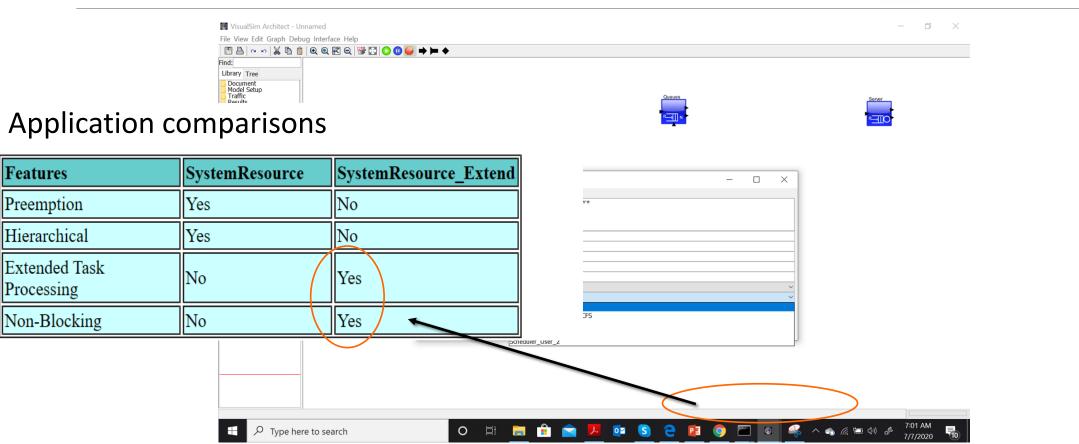**SystemResource**
task_output
Type: General

task_plot
Type: Double

Edit parameters for SystemResource — □ ✕

Block_Documentation:	📝 Enter User Documentation Here
Resource_Name:	"CPU"
Next_Resource:	"Fld_Name_or_String_or_None"
Task_Context_Switch_Time:	0.0
Round_Robin_Time_Slice:	1.0E-3
Clock_Rate_Mhz:	500.0
Max_Scheduler_Length:	30
Time_Type:	Relative Time
Scheduler_Type:	Relative Time
Add_Scheduler_Times_to_DS:	Number Clocks

Commit   Add   Remove   Restore Defaul...   Preferences   Help

This is the name of this SystemResource block and is used by Mappers, RegEx and other SystemResource block to call this block to execute a transaction.

Next_Resource is the name of the next hierarchical System Resource, which can be SystemResource or SystemResource_Extend block.

System Resource (Cont.)

Edit parameters for SystemResource

Block_Documentation: Enter User Documentation Here

Resource_Name: "CPU"
Next_Resource: "Fld_Name_or_String_or_None"
Task_Context_Switch_Time: 0.0
Round_Robin_Time_Slice: 1.0E-3
Clock_Rate_Mhz: 500.0
Max_Scheduler_Length: 30

Time_Type: Relative Time
Scheduler_Type: Scheduler_FCFS

Add_Scheduler_Times_to_DS:
Scheduler_FCFS
FCFS + Preempt
Scheduler_RR
Scheduler_User_1
Scheduler_User_2

Commit   Add

Time the scheduler will devote to each task for Round Robin

Application comparisons

Features	SystemResource	SystemResource_Extend
Preemption	Yes	No
Hierarchical	Yes	No
Extended Task Processing	No	Yes
Non-Blocking	No	Yes

Set Scheduler type from the range of schedulers

System Resource Extended

- Double click to configure

Application comparisons

Features	SystemResource	SystemResource_Extend
Preemption	Yes	No
Hierarchical	Yes	No
Extended Task Processing	No	Yes
Non-Blocking	No	Yes

Return

What is Mapper?

- Takes incoming Data Structure and sends to
  - SystemResource
  - SystemResource_Extend blocks
- Placed in the behavior flow where timed resources required
- Consumes zero time, no queue, no arbitration

Mapper

System Resource

Software Mapper
Hardware or Software Task issuer

input
Type: General
"non...
output
Type: General

- Sends tasks to SystemResource or SystemResource_Extend based on Target_Resource
- Block can either Queue incoming Data Structure or send to SystemResource immediately

Attributes to issue the task to System Resource

Edit parameters for SoftwareMapper

Block_Documentatio... Enter User Documentation Here

Target_Resource: "CPU"
Task_Number: 2
Task_Priority: 2
Task_Mean_Time: Scan_Proc_Time
Task_Spread_Time: Task_Spread_Fld_Int_Dbl_Expr
Random_Seed: 123457L
Task_Distribution: Fixed (Mean)
Task_Type: Queue Task Now
Task_Mutual_Exclusion: No

Commit   Add   Remove   Restore Defaults   Preferences   Help

Lock out all other tasks from preempting this Task at the SystemResource

3 tasks competing for the same resource

$VS/doc/Training_Material/Tutorial/General/System_Resources/Scheduler_Sw_2.xml
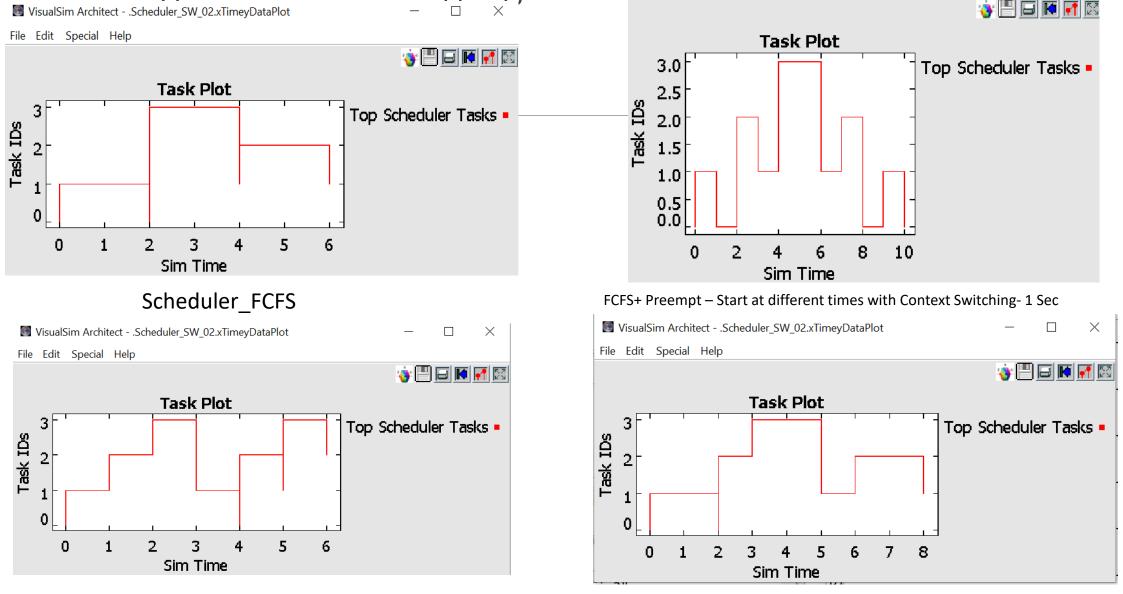
Plots and Stats for Example Model 1

Tasks are not concurrent
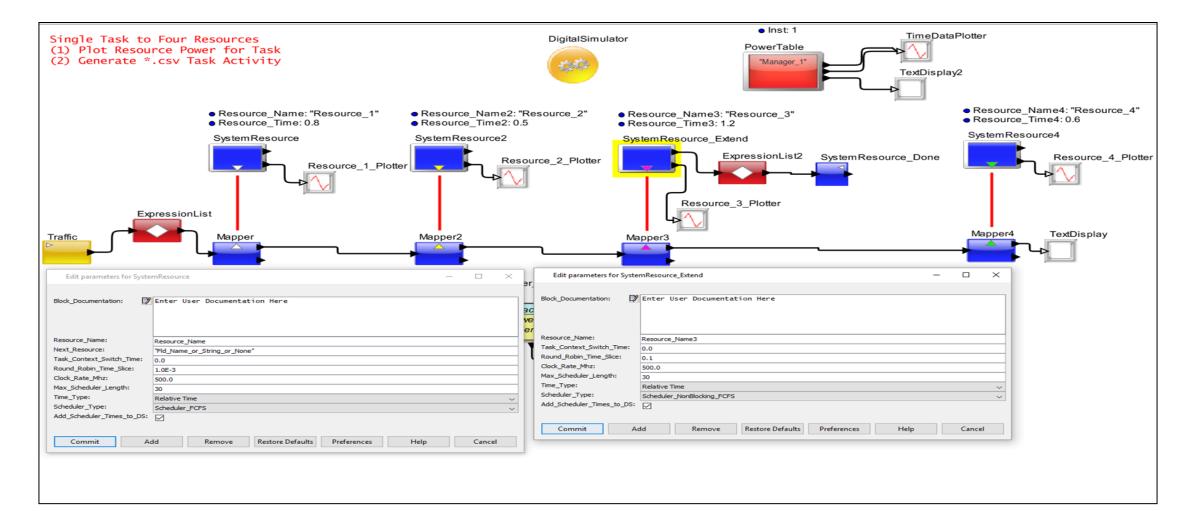
Different Types of Scheduling Algorithms in SystemResource

Scheduler_FCFS

FCFS+ Preempt – Start at different times with Context Switching- 1 Sec

Scheduler_RR

Scheduler_FCFS+Preempt

Example Model – System Resource Extend

Queue
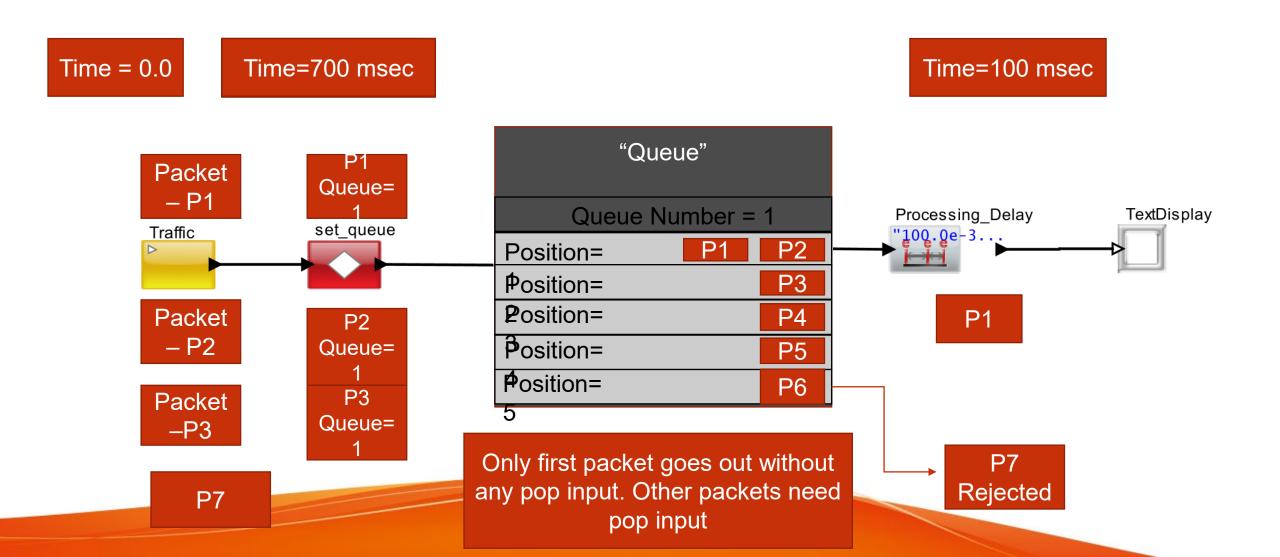
Edit parameters for Queues

Block_Documentation:	Enter User Documentation Here
Block_Name:	"Queue"
Queue_Number_Field:	input.queue
Priority_Field:	input.priority
Max_Queue_Length:	30
Number_of_Queues:	1
Initial_Queue_State:	First_Token_Flow_Through
Queue_Reject_Mechanism:	Incoming_Token_Rejected
Queue_Type:	FIFO

Commit   Add   Remove   Restore Defaul...   Preferences   Help

Provides the priority number for reordering the queue.

Set how the packets should flow

Return

Queue – First Token Flow Through
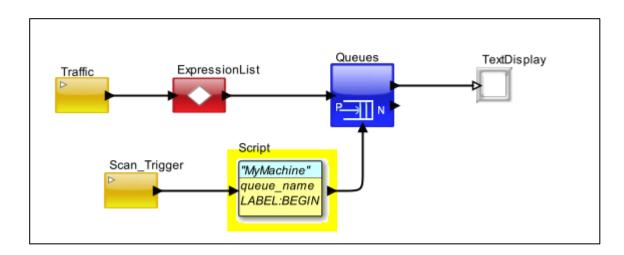
Queue – First token Enqueue

- pop_input is required to send all the packets including the first one

Multiple number of Queues

Internally, it look like this

"Queue"		
Q Num=1	Q Num=2	Q Num=3
P1	P3	P5
P2		
P4		

Pos=

Pos= 2

Pos= 3

Pos= 4

Pos= 5

pop_input value can be 1, 2 or 3

Queue Operation - Summary
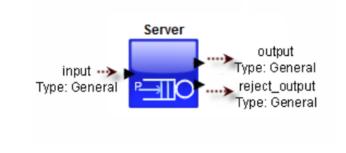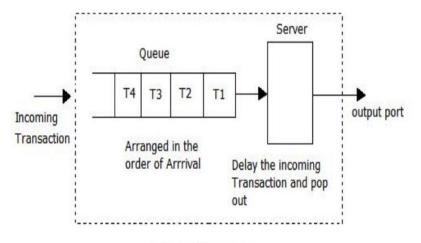
- Data Structures are queued based on priority from high to low number
- Data Structures in the queue are arranged based on FIFO or LIFO setting
- Number_of_Queues defines the number of parallel queues contained by a single Queue block
- Queue Number Field selects the queue to place
- To pop a packet
  - From the head of a queue, Queue_Number must be sent to pop_input port.
- When Maximum_Queue_Length is reached, packets are Rejected based on Rejection_Mechanism and sent to Reject_output
- Based on initial Queue State parameter,
  - Enqueue: First Transaction can be enqueued and wait for the pop
  - First_Packet_Flow_Through: First transaction send without pop. After first packet, head of queue sent if prior was acknowledged with pop

Server

- Define multiple {queues + time delay}
  - Active Resource
  - DataStructures queued in FIFO or LIFO order

- Processing time is known in advance
  - Provided along with the transaction to this block.

- SLOT
  - Special operation mechanism
  - Models any slot-based architecture such as multiple virtual RTOS, TDMA etc.

Basic Timed Queue

Server

Edit parameters for Server

Block_Documentation: Enter User Documentation Here

Block_Name: "server"
Queue_Number_Field: input.queue
Priority_Field: input.priority
Time_Field: input.time
Max_Queue_Length: 30
Number_of_Queues: 1
Queue_Type: FIFO

Commit   Add   Remove   Restore Defaul...   Preferences   Help   Cancel

Server has a special parameter called "Time field" to delay head of queue before sending out

Number_of_Queues can be multiple in the Server as well

Return

Server - Summary

- *Queue_Number_Field* selects the queue

- Queue is reordered based on *Priority field*

- Queue data in FIFO or LIFO based on *Queue_Type*

- Delayed by *Time_Field* value at head of queue and sent out

- Packet is sent to *reject_output* when Max_Queue_Length reached

Edit parameters for Server

Block_Documentation:	Enter User Documentation Here
Block_Name:	"Server"
Queue_Number_Field:	input.Queue_Num
Priority_Field:	1
Time_Field:	input.Delay
Max_Queue_Length:	30
Number_of_Queues:	5
Queue_Type:	FIFO

Commit | Add | Remove | Restore Defaults | Preferences | Help | Cancel

Performance and Power Metrics

- Common Statistics
  - ✓ End-to-end latency and Task Delay
  - ✓ Throughput (MIPS or MB/s), Utilization (%)
  - ✓ Minimum, maximum, mean and standard deviation statistics
  - ✓ Power – Instantaneous, Average and Cumulative
  - ✓ Battery – Current Capacity(Watt - Hr), Battery Life Remaining (%)

- Processor
  - ✓ Individual statistics for Caches, Execution Units and Pipeline Stages
  - ✓ Flush Time, Stall (%), Thread swaps and Context switching
  - ✓ Listener for Real-Time pipeline activity

- Cache
  - ✓ Hit-miss Ratio
  - ✓ Requests, utilization, evictions, prefetches and latency and throughput
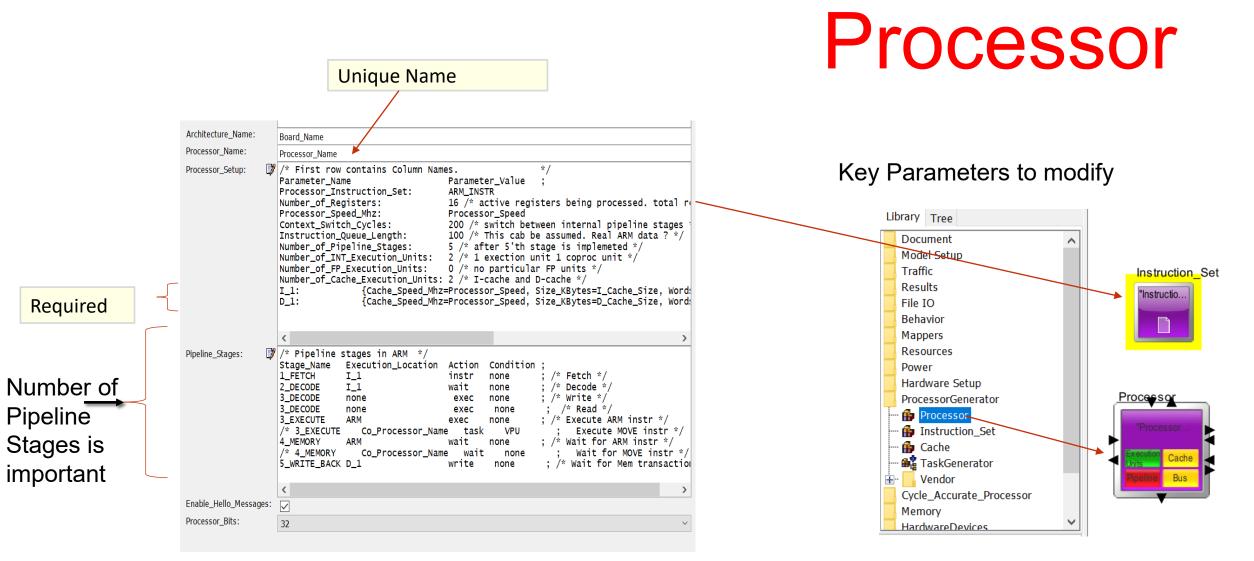
Hardware Statistics

Name	Value	Name	Value	Name	Value
Bus_1_Utilization_Pct_Max	10.1,	Bus_1_Delay_Max	3.600004E-8,	Processor_1_D_1_Hit_Ratio_Max	100.0,
Cache_1_Utilization_Pct_StDev	1.05689435,	Bus_1_IOs_per_sec_StDev	634428.24721,	Processor_1_D_1_KB_per_Thread_StDev	0.0,
Processor_1_D_1_Utilization_Pct_Max	1.45,	Bus_1_Input_Buffer_Occupancy_in_Words_Max	32.0,	Processor_1_I_1_Hit_Ratio_Max	100.0,
Processor_1_INT_1_Utilization_Pct_StDev	0.10969973,	Bus_1_Preempt_Buffer_Occupancy_in_Words_StDev	0.0,	Processor_1_I_1_KB_per_Thread_StDev	0.0,
Processor_1_INT_2_Utilization_Pct_Max	24.55,	Bus_1_Throughput_MBs_Max	176.0,	Processor_1_L_2_Hit_Ratio_Max	100.0,
Processor_1_I_1_Utilization_Pct_StDev	0.01500007,	Cache_1_Delay_Time_StDev	1.499871E-8,	Processor_1_L_2_KB_per_Thread_StDev	0.0,
Processor_1_L_2_Utilization_Pct_Max	3.1,	Cache_1_Hit_Ratio_Max	100.0,	Processor_1_Stall_Time_Pct_Max	1.35,
Processor_1_PROC_Utilization_Pct_StDev	0.03489914,	Cache_1_Memory_Used_By_Processor_1_MB_StDev	8.455181E-5,	Processor_1_Task_Delay_StDev	2.9502E-7,
Processor_1_Pipeline_Utilization_Pct_Max	50.2,	Cache_1_Memory_Used_By_SDRAM_1_MB_Max	2.56E-4,	SDRAM_1_Delay_Time_Max	1.5E-7,
Processor_1_Register_Rd_Utilization_Pct_StDev	0.19330594,	Cache_1_Memory_Used_By_Total_MB_StDev	8.455151E-5,	SDRAM_1_Memory_Used_By_Processor_1_MB_StDev	0.0,
Processor_1_Register_Wr_Utilization_Pct_Max	49.55,	Cache_1_Throughput_MBs_Max	116.0,	SDRAM_1_Memory_Used_By_Total_MB_Max	2.56E-4,
SDRAM_1_Utilization_Pct_StDev	1.91E-07	Processor_1_Context_Switch_Time_Pct_StDev	0.0,	SDRAM_1_Throughput_MBs_StDev	0.0,
				DMA_IO_per_sec_Max	7.45E6,
				DMA_Throughput_MBs_StDev	3.463999999,

Analyzing Performance in VisualSim – Important Parameters in Library Components

Hybrid Processor

Processor

Unique Name

Architecture_Name: Board_Name

Processor_Name: Processor_Name

Processor_Setup:
```
/* First row contains Column Names.              */
Parameter_Name              Parameter_Value  ;
Processor_Instruction_Set:      ARM_INSTR
Number_of_Registers:            16 /* active registers being processed. total re
Processor_Speed_Mhz:            Processor_Speed
Context_Switch_Cycles:          200 /* switch between internal pipeline stages
Instruction_Queue_Length:       100 /* This cab be assumed. Real ARM data ? */
Number_of_Pipeline_Stages:      5 /* after 5'th stage is implemeted */
Number_of_INT_Execution_Units:  2 /* 1 exection unit 1 coproc unit */
Number_of_FP_Execution_Units:   0 /* no particular FP units */
Number_of_Cache_Execution_Units: 2 /* I-cache and D-cache */
I_1:            {Cache_Speed_Mhz=Processor_Speed, Size_KBytes=I_Cache_Size, Words
D_1:            {Cache_Speed_Mhz=Processor_Speed, Size_KBytes=D_Cache_Size, Words
```

Required

Number of Pipeline Stages is important

Pipeline_Stages:
```
/* Pipeline stages in ARM  */
Stage_Name    Execution_Location  Action   Condition ;
1_FETCH       I_1                 instr    none     ; /* Fetch */
2_DECODE      I_1                 wait     none     ; /* Decode */
3_DECODE      none                exec     none     ; /* Write */
3_DECODE      none                exec     none     ;  /* Read */
3_EXECUTE     ARM                 exec     none     ; /* Execute ARM instr */
/* 3_EXECUTE    Co_Processor_Name  task     VPU       ;    Execute MOVE instr */
4_MEMORY      ARM                 wait     none     ; /* Wait for ARM instr */
/* 4_MEMORY     Co_Processor_Name  wait    none      ;   Wait for MOVE instr */
5_WRITE_BACK D_1                   write    none     ; /* Wait for Mem transaction
```

Enable_Hello_Messages: ☑

Processor_Bits: 32

Key Parameters to modify

Library | Tree

- Document
- Model Setup
- Traffic
- Results
- File IO
- Behavior
- Mappers
- Resources
- Power
- Hardware Setup
- ProcessorGenerator
  - Processor
  - Instruction_Set
  - Cache
  - TaskGenerator
  - Vendor
- Cycle_Accurate_Processor
- Memory
- HardwareDevices

Instruction_Set

Processor

Outstanding_Req_Count is used for External Cache

Processor – Internal Cache Configuration

Edit parameters for ARM9

Block_Documentatio...
```
Mapping of ARM9 CPU architecture parameters to generic CPU
Please NOTE that this is not goldenized and is only for Demo purpose.
```

Architecture_Name: Board_Name

Processor_Name: Processor_Name

Processor_Setup:
```
/* First row contains Column Names.                */
Parameter_Name                Parameter_Value   ;
Processor_Instruction_Set:    ARM_INSTR
Number_of_Registers:          16 /* active registers being processed. total reg = 31 */
Processor_Speed_Mhz:          Processor_Speed
Context_Switch_Cycles:        200 /* switch between internal pipeline stages */
Instruction_Queue_Length:     100 /* This cab be assumed. Real ARM data ? */
Number_of_Pipeline_Stages:    5 /* after 5'th stage is implemeted */
Number_of_INT_Execution_Units: 2 /* 1 exection unit 1 coproc unit */
Number_of_FP_Execution_Units: 0 /* no particular FP units */
Number_of_Cache_Execution_Units: 2 /* I-cache and D-cache */
I_1:         {Cache_Speed_Mhz=Processor_Speed, Size_KBytes=I_Cache_Size, Words_per_Cache_Line=8, Hit_Ratio=0.4, Cache_Miss_Name=Cache}
D_1:         {Cache_Speed_Mhz=Processor_Speed, Size_KBytes=D_Cache_Size, Words_per_Cache_Line=8, Hit_Ratio=0.6, Cache_Miss_Name=Cache}
```

Micro Architecture details – Number of Registers, ROB size, Number of Registers

Hit_Ratio by default will be 0.99

Pipeline_Stages:
```
/* Pipeline stages in ARM  */
Stage_Name    Execution_Location  Action  Condition ;
1_FETCH       I_1                 instr   none    ; /* Fetch */
2_DECODE      I_1                 wait    none    ; /* Decode */
3_DECODE      none                exec    none    ; /* Write */
3_DECODE      none                exec    none    ;  /* Read */
3_EXECUTE     ARM                 exec    none    ; /* Execute ARM instr */
/* 3_EXECUTE    Co_Processor_Name  task    VPU      ;   Execute MOVE instr */
4_MEMORY      ARM                 wait    none    ; /* Wait for ARM instr */
/* 4_MEMORY     Co_Processor_Name  wait    none     ;   Wait for MOVE instr */
5_WRITE_BACK  D_1                 write   none    ; /* Wait for Mem transaction */
```

Mention next Level Cache to be accessed in case of misses

Enable_Hello_Messages: ☑

Processor_Bits: 32

Processor Instruction Set

Processor name

MIRABILIS design

```
File  Edit  Help
1   /* Instruction Set or File Path. */
2
3   Mnew Ra Rb Rc Rd Re Rf Rg Rh ; /* Label */
4   RISCV INTG ALU FPNEOSIMD LDSTR ;
5
6
7   ALU        INT_1 INT_2 INT_3 ; /* Branch, Mul/Div, Simple ALU  */
8   FPNEOSIMD FP_1            ; /*Floating point/NEON/ASIMD instruction */
9   LDSTR      INT_4          ; /*Load/Store instructions*/
10
11
12  begin size_config                    ;
13   Read    5    128    INT_4[1:2]      ;
14   Read    5    64     INT_4[3:7]      ;
15   Read    5    32     INT_4[8:16]     ;
16   Read    5    16     INT_4[17:18]    ;
17   Read    5    8      INT_4[19:20]    ;
18   Write   5    128    INT_4[21:22]    ;
19   Write   5    64     INT_4[23:27]    ;
20   Write   5    32     INT_4[28:35]    ;
21   Write   5    16     sh              ;
22   Write   5    8      sb              ;
23  end size_config                      ;
```

Execution and Load Store Units within RISCV and the corresponding instruction groups they execute

```
begin execUnit_config              ;
Queue_Size    INT_1        16       ;
Queue_Size    INT_2        16       ;
Queue_Size    INT_3        16       ;
Queue_Size    INT_4        16       ;
Queue_Size    INT_5        16       ;
Queue_Size    FP_1         16       ;
Queue_Size    FP_2         16       ;
Queue_Size    INT_6        12       ;
Queue_Size    INT_7        12       ;
end execUnit_config                ;
```

Queue_Size for each Instruction Group

```
begin INT_1      ;  /* Group */
ADD    2         ;
SUB    2         ;
*b     2         ;
MUL    4         ;
DIV    4         ;
LDR    1         ;
LDUR   1         ;
STR    1         ;
end    INT_1     ;

begin FP_1        ;  /* Group */
FADD   2          ;
FSUB   2          ;
FMUL   4     8       ;
FDIV   4     12      ;
end    FP_1       ;
```

Instruction Group

The entry FADD 2   ; means that the instruction "FADD" will take 2 cycles to complete execution (Without including the I_Cache access latency and pipeline transfer latency).

The entry FMUL 4   8   ; means that the instruction FMUL can take a random delay cycle between 4 and 8 cycles to complete execution.
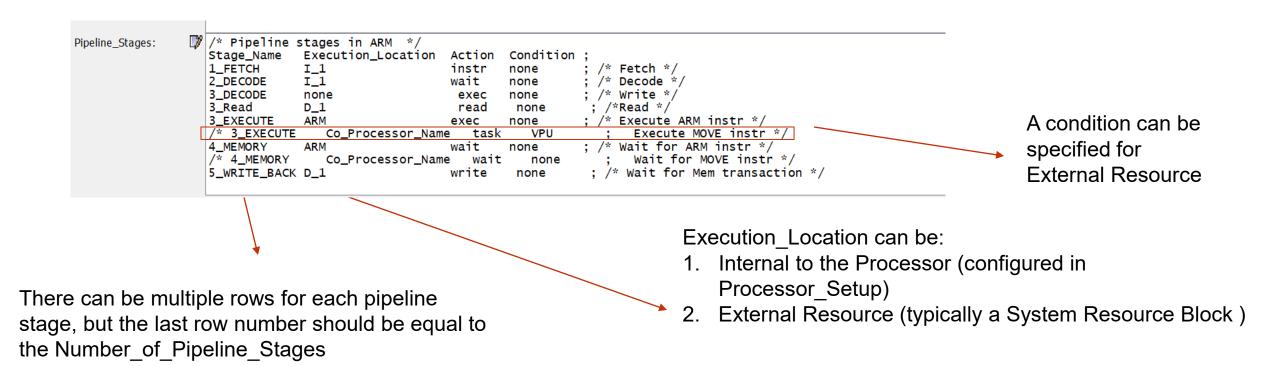
Instruction width in bits can vary for different load/store instructions
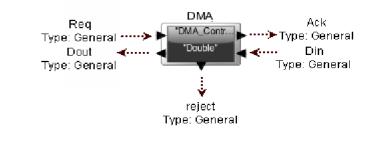
Example of Pipeline usage


```
Pipeline_Stages:    /* Pipeline stages in ARM  */
                    Stage_Name    Execution_Location  Action  Condition ;
                    1_FETCH       I_1                 instr   none       ; /* Fetch */
                    2_DECODE      I_1                 wait    none       ; /* Decode */
                    3_DECODE      none                exec    none       ; /* Write */
                    3_Read        D_1                 read    none       ; /*Read */
                    3_EXECUTE     ARM                 exec    none       ; /* Execute ARM instr */
                    /* 3_EXECUTE     Co_Processor_Name    task     VPU       ;    Execute MOVE instr */
                    4_MEMORY      ARM                 wait    none       ; /* Wait for ARM instr */
                    /* 4_MEMORY      Co_Processor_Name   wait    none       ;    Wait for MOVE instr */
                    5_WRITE_BACK D_1                  write   none       ; /* Wait for Mem transaction */
```

A condition can be specified for External Resource

Execution_Location can be:
1. Internal to the Processor (configured in Processor_Setup)
2. External Resource (typically a System Resource Block)

There can be multiple rows for each pipeline stage, but the last row number should be equal to the Number_of_Pipeline_Stages

DMA BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT?

1. **Req port** receives request from the source (Traffic generator in this case). It will be connected to DeviceInterface
2. **Ack port** will have transactions received with additional fields like Time_Array, Trace_Array useful for debug
3. **Ack and Din** are connected to the output blocks via DeviceInterface or directly
4. **Dout** carries response from output devices like Memory. Connected to Bus, DeviceInterface or Memory
5. **Dout** will have Task_Latency and Task_Throughpout as additional fields
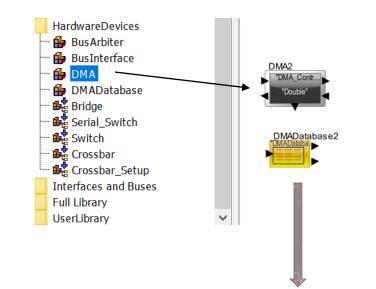
DMA – DEFINING ACTIONS USING DATA STRUCTURE

Edit parameters for DMA

Block_Documentation: [✎] Enter User Document

Architecture_Name: "Architecture_1"
DMA_Controller_Name: "DMA_Controller_1"
Memory_Database_Reference: "None"
DMA_to_Device_Cycles: RegEx_or_None
Device_to_DMA_Cycles: RegEx_or_None
Channel_FIFO_Buffers: Integer
Speed_Mhz: Double
DMA_Channels: 8
Outstanding_Req_Count: {1, 1, 1, 1, 1, 1, 1, 1}
Width_Bytes: 4

Traffic
ExpressionList
DeviceInterface   DMA
"IO_Name"   "DMA"
             "100.0"

ExpressionList2

Task_Latency

Task_Throughput

Use Input (none) while using data structure and database name when using DMA database

Required

Per channel, num sent before Ack

Expression_List: [✎]

```
/* Template to enter multiple RegEx lines*/
input.A_Task_Name = "Activity1"
input.A_DMA_Command = {"Read","Write"}
input.A_DMA_Destination = {"RAM","Display"}
input.A_Priority = irand(1,10)
input.A_DMA_Bytes = {1024,1024}
input.A_DMA_Channel = {1,1}
input.A_DMA_Burst_Bytes = {64,64}
```

Input to DMA block

If the user wants to define actions for the DMA from Data structure fields, then above fields are necessary

USING THE DMA DATABASE

HardwareDevices
- BusArbiter
- BusInterface
- **DMA**
- DMADatabase
- Bridge
- Serial_Switch
- Switch
- Crossbar
- Crossbar_Setup

Interfaces and Buses
Full Library
UserLibrary

DMA2
"DMA_Contr..."
"Double"
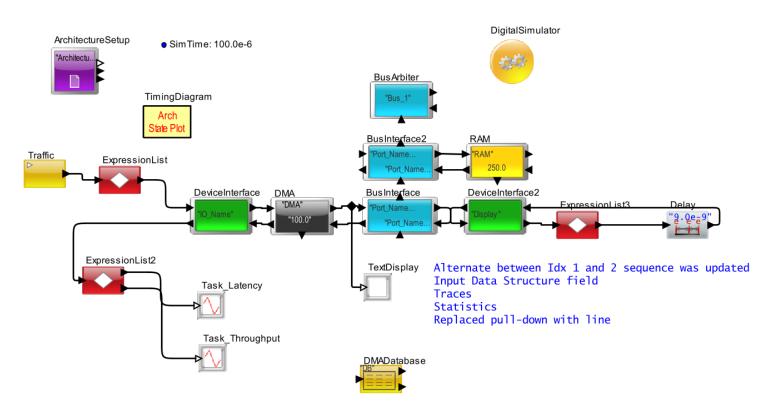
DMA Controller Block

DMADatabase2
"DMADataba"

Linking_Name:	"DMADatabase"
fileOrURL:	Browse

Data_Structure_Text:

A_Task_Name	A_Instruction	A_IDX	A_Task_Source	Burst_Word_Size	A_Task_Address	A_Command	A_Bytes	A_Priority	A_Destination |;
Activity1	Load	0	SRAM	32	1	Read	32	0	DMA ;
Activity1	Store	0	DRAM	32	1	Write	32	1	DMA ;
Activity2	Store	0	DRAM	32	2	Write	32	3	DMA ;

Match input fields

Sequence Operation

Destination

Send as Burst size

Channel Number

Transfer Size

DMA – DEMO MODELS AND RESULTS

MODEL SHOWCASING DS FIELDS AS WELL AS DMA DATABASE OPTION
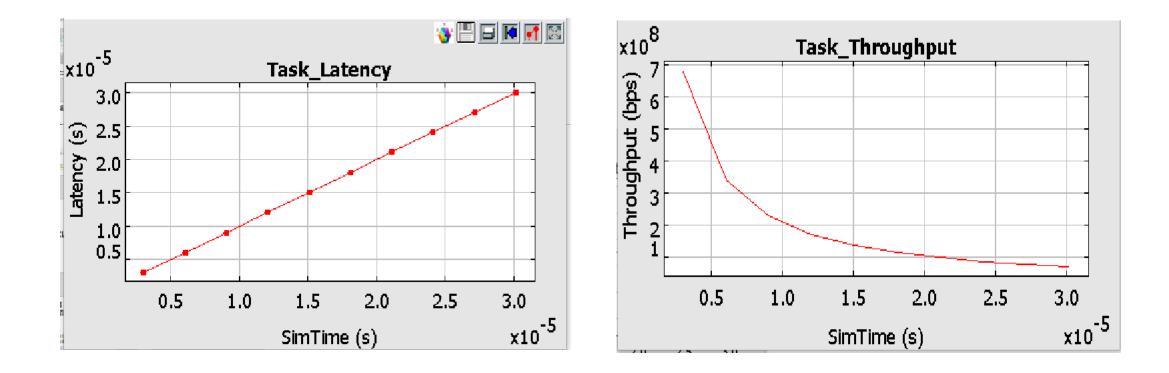
Model Location :
$VS/doc/Training_Material/Tutorial/Architecture/DMA/DMA_Demo_Model_Field_n_DB.xml
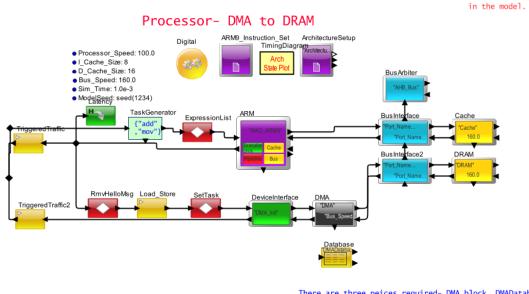
RESULTS FOR THE PREVIOUS MODEL

CONNECTING PROCESSOR – DMA - DRAM

Model Location :

$VS/doc/Training_Material/Tutorial/Architecture/DMA/Processor_DMA.xml
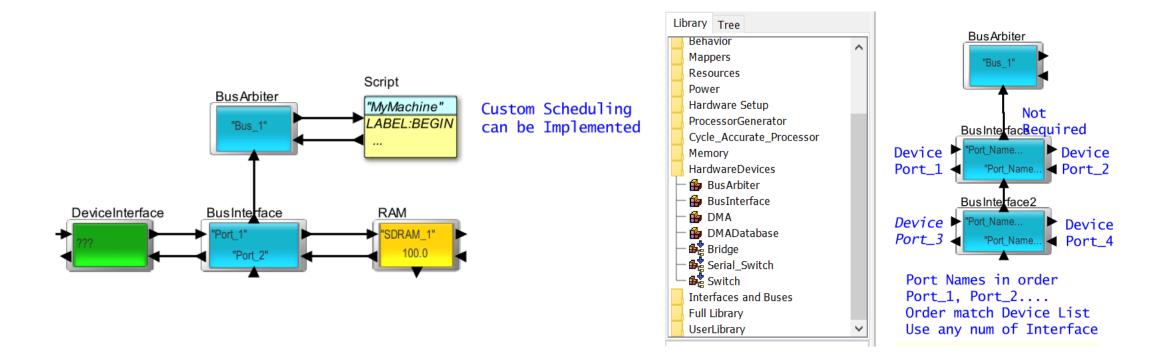
RESULTS FOR THE PREVIOUS MODEL

BUS ARBITER AND BUS INTERFACE BLOCKS AND ITS USAGE

PORTS AND HOW TO CONNECT

KEY CONFIGURATION

Edit parameters for BusArbiter2   — □ ×

Block_Documentation:

Enter User Documentation Here

Architecture_Name: "Architecture_1"

_explanation: HardwareDevices->BusArbiter

Bus_Name: Cluster_Name+"_DSU_Cache_Bus"   **Unique Name**

Bus_Speed_Mhz: Core_Speed_Mhz

Burst_Size_Bytes: 100

Round_Robin_Port_Array: {"Port_1", "Port_2"}

Devices_Attached_to_Slave_by_Port: 1"}, {"Device_2"}, {"Device_3"}, {"Device_4"}, {"Device_5"}, {"Device_6"}, {"Device_7"}, {"Device_8"}}

Width_Bytes: {16,64}   **{Read,Write}**

Arbiter_Mode: FCFS

Split_Retry_Flag: ☑

Enable_Plots: ☐

Required

List all devices connected via this port. Index is in order of name- Port_1, Port_2 etc
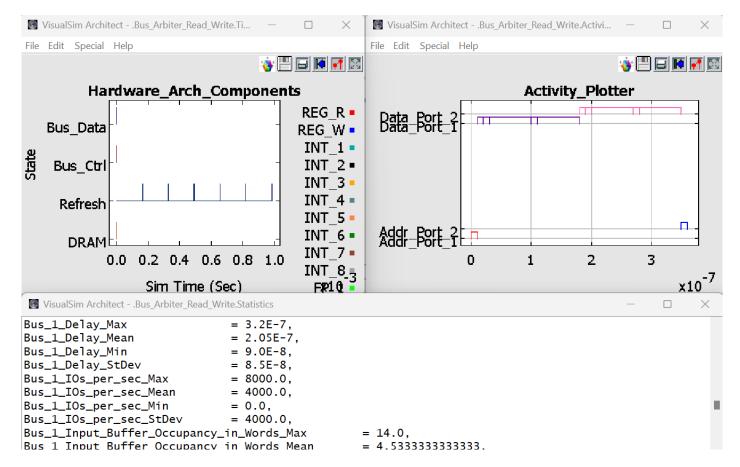
DEMO MODEL

Model Location :
$VS/doc/Training_Material/Architecture/AHB_AXI/Bus_Arbiter_Read_Write.xml

RESULTS FOR THE PREVIOUS MODEL
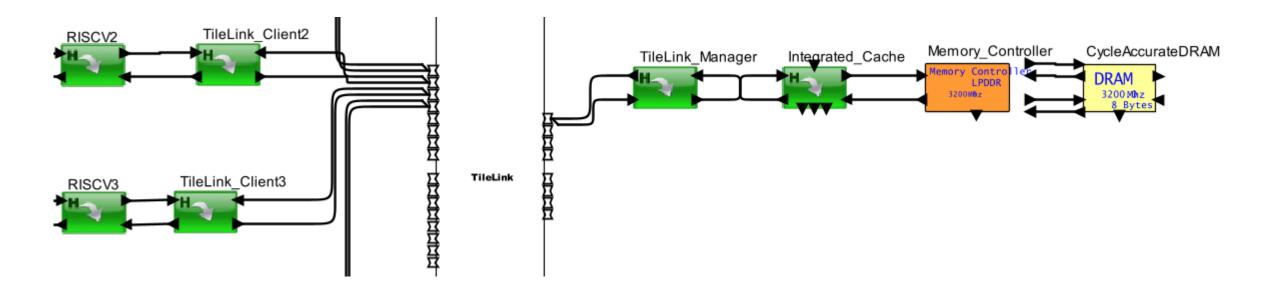
TILELINK AND ITS USAGE

PORTS AND HOW TO CONNECT?
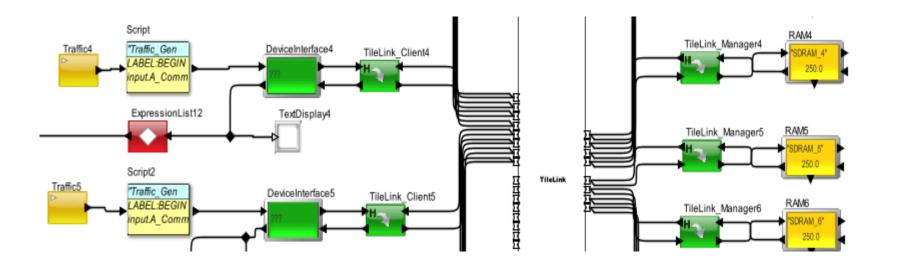
1. Source (master devices that request data) like Processor and DMA (standard blocks) have to be connected to the master ports via TileLink_Client (Library -- > Interfaces and Buses --> Tilelink --> TileLink_Client)

CONTINUED
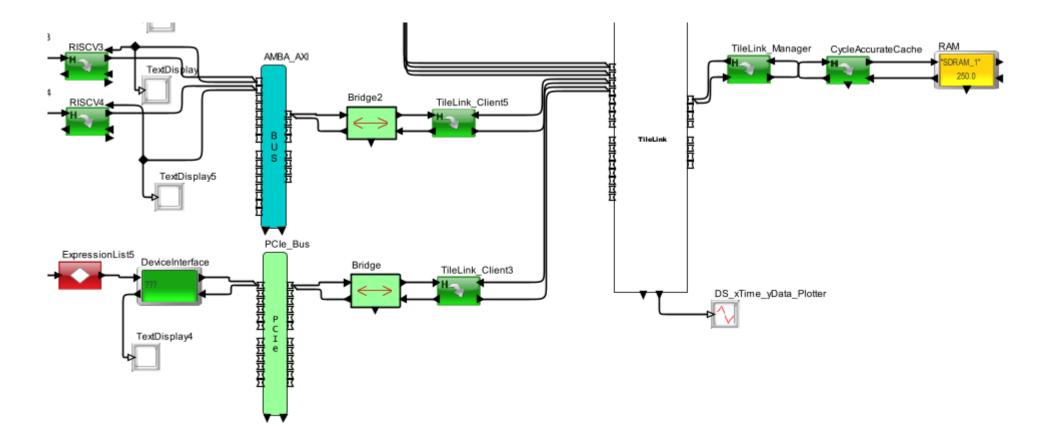
2. If the Master device is designed with simple library block (as logic) then user has to include Device Interface block to provide the interface with master.

CONTINUED

3. If the TileLink_Client is connecting to another BUS like PCIe or AMBA-AXI, then the user has to use a Bridge (Library --> HardwareDevices --> Bridge)

KEY CONFIGURATION

Edit parameters for TileLink

TileLink_Name:	"TileLink"
TileLink_Speed_Mhz:	1000.0
TileLink_Cycle_Time:	(1.0E-6/TileLink_Speed_Mhz)
Bus_Width:	8
Number_Masters:	4
Number_Slaves:	1
Slave_Speeds_Mhz:	{TileLink_Speed_Mhz, TileLink_Speed_Mhz, TileLink_Speed_Mhz, TileLink_Speed_Mhz, TileLin
Extra_Cycles_for_RdReq_WrReq_RdData_WrData:	{0, 0, 0, 0, 0, 0, 0, 0}
Devices_Attached_to_Slave_by_Port:	{{"L2","SDRAM_1"},{"SDRAM_2"},{"SDRAM_3"},{"SDRAM_4"},{"SDRAM_5"},{"SDRAM_6"}
Master_First_Word_Flag:	true
DEBUG:	false
Ports_to_Plot:	{1,1} /* master n, slave m, 0 disables */
Managers_Attached_to_Slave_by_Port:	{{"Manager1"},{"Manager2"},{"Manager3"},{"Manager4"},{"Manager5"},{"Manager6"},{"Ma
Architecture_Name:	"Architecture_1"

Commit   Add   Remove   Restore Defaults   Preferences   Help   Cancel

Required

From Architecture Setup

TileLink Block

Edit parameters for TileLink_Client2

Client_Name:	"Client2"
Client_Speed:	1000.0
Release_Threshold:	5
Architecture_Name:	"Architecture_1"
No_Of_Retry_FirstBurst:	3

Commit   Add   Remove

TileLink_Client
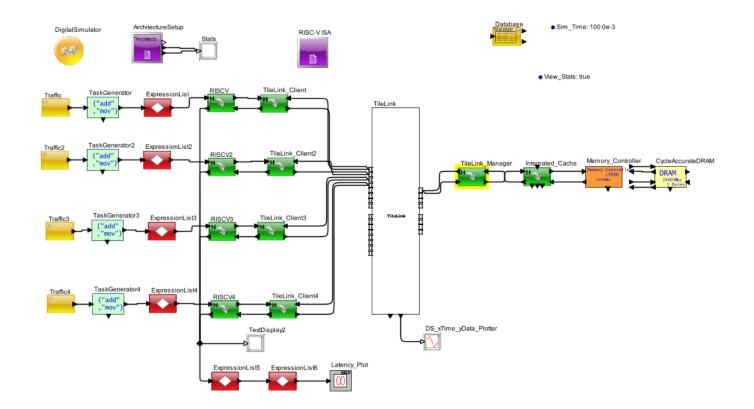
refers to how many times a TileLink client will attempt to send a burst of data to a target device before giving up and retrying the transaction.

Edit parameters for TileLink_Manager

Manager_Name:	"Manager1"
Manager_Speed:	1000.0
Architecture_Name:	"Architecture_1"
TileLink_Name:	"TileLink"

Commit   Add   Remov

All the fields are Required

TileLink_Manager

DEMO MODEL
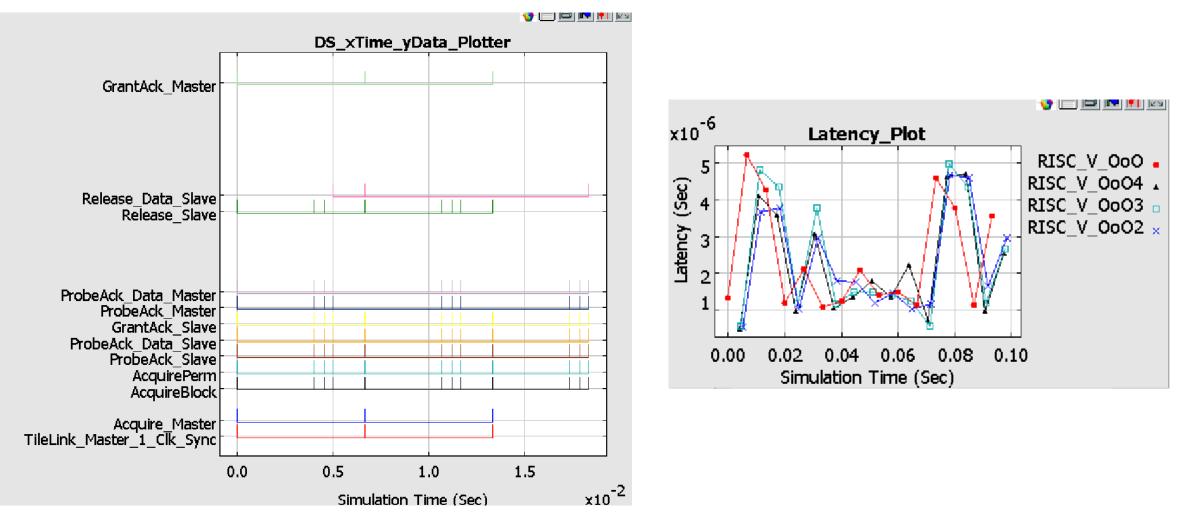
Model Location : $VS/demo/Bus_Std/TileLink_Models/TileLink_RISCV_SoC_Approach.xml

PLOTS FOR THE PREVIOUS MODEL

Timing Diagram for TileLink Bus showing the messages that were sent during the transactions

STATS FOR THE PREVIOUS MODEL

VisualSim Architect - .TileLink_RISCV_SoC_Approach.TextDisplay2

```
CYCLES_PER_INSTRUCTION          = 6.7945205479452,
DELTA                           = 0.098333333338,
DS_NAME                         = "Processor_DS",
FP_1_Max_Buffer_Size            = 5,
FP_1_Usage_Pct                  = 6.029810298103,
ID                              = 15,
INDEX                           = 0,
INT_1_Max_Buffer_Size           = 2,
INT_1_Usage_Pct                 = 3.6585365853659,
INT_2_Max_Buffer_Size           = 22,
INT_2_Usage_Pct                 = 21.6124661246612,
INT_3_Max_Buffer_Size           = 1,
INT_3_Usage_Pct                 = 1.6260162601626,
INT_4_Max_Buffer_Size           = 78,
INT_4_Usage_Pct                 = 100.1355013550135,
INT_5_Max_Buffer_Size           = 0,
INT_5_Usage_Pct                 = 0.4065040650407,
MHZ_PROCESSOR                   = 500.0,
MIPS_IN_PROCESSOR               = 73.5850007560032,
Pipeline_Stall_Pct              = {5.4878048780488, 1.6937669?
Processor_Total_Stall_Pct       = 0.8130081300813,
ROB_Stall_Pct                   = 0.0,
Stall_Fetch_Pct                 = 0.0,
Stall_Load_Pct                  = 0.0,
TIME                            = 0.098333333338,
TIME_IN_PROCESSOR               = 2.9761500000003E-6}
```

```
{BLOCK                 = "TileLink_RISCV_SoC_Approach.TileLink.TileLink_Channel_C",
DS_NAME                = "TileLink_Channel_C_4",
Number_Entered         = 145,
Number_Exited          = 145,
Occupancy_Max          = 3.0,
Occupancy_Mean         = 0.5896551724138,
Occupancy_Min          = 0.0,
Occupancy_StDev        = 0.5877161099728,
Total_Delay_Max        = 3.0E-9,
Total_Delay_Mean       = 1.0896551724786E-9,
Total_Delay_Min        = 9.9999999947364E-10,
Total_Delay_StDev      = 3.0888545764435E-10}
{BLOCK                 = "TileLink_RISCV_SoC_Approach.TileLink.TileLink_Channel_D",
DS_NAME                = "TileLink_Channel_D_1",
Number_Entered         = 124,
Number_Exited          = 124,
Occupancy_Max          = 1.0,
Occupancy_Mean         = 0.5,
Occupancy_Min          = 0.0,
Occupancy_StDev        = 0.5,
Total_Delay_Max        = 1.0000000012084E-9,
Total_Delay_Mean       = 1.0000000000004E-9,
Total_Delay_Min        = 9.9999999947364E-10,
Total_Delay_StDev      = 2.4812421759788E-17}
```

Processor Stats

Channel wise Occupancy and
Delay for TileLink
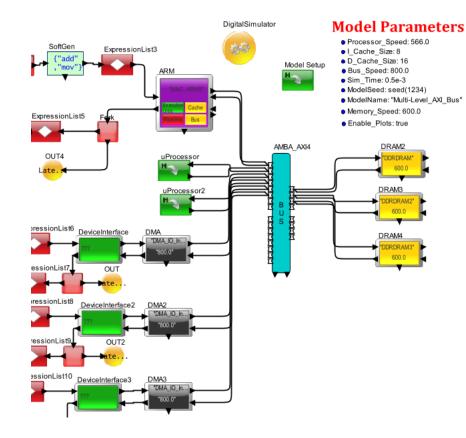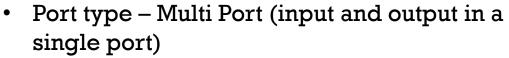
AXI BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT?

- Support 16 masters and 8 slaves
- Master and slave ports can be increased if it is required. Eg: AXI_16_x16 block

- **Master and Slave Ports** :
  - Port type – Multi Port (input and output in a single port)
  - Port data type – general
  - Support only 2 wire connection per port (input and output)
- **Input ports**: Accept read/write requests from master devices and forward read/write respons to master devices
- **Output ports**: Forward read/write requests to slave devices and accept read/write response from slave devices.
- **Stats_out**: provide debug messages during the simulation if "DEBUG" parameter is set.
- **Plot_out:** Not used in the current version.

KEY CONFIGURATION

Bus_Width:	8
Read_Threshold:	2
Write_Threshold:	2
Master_Request_Threshold:	{2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2}

- **AXI_Speed_Mhz**: Bus speed in Mhz
- **Bus_Width**: Width of the AXI Data and response channel
- **Read_Threshold**: Total number of outstanding reads in each slave ports.
- **Write_Threshold**: Total number of outstanding writes in each slave ports if A_Task_Flag is true.
- **Master_Request_Threshold**: Input queue size of master ports.
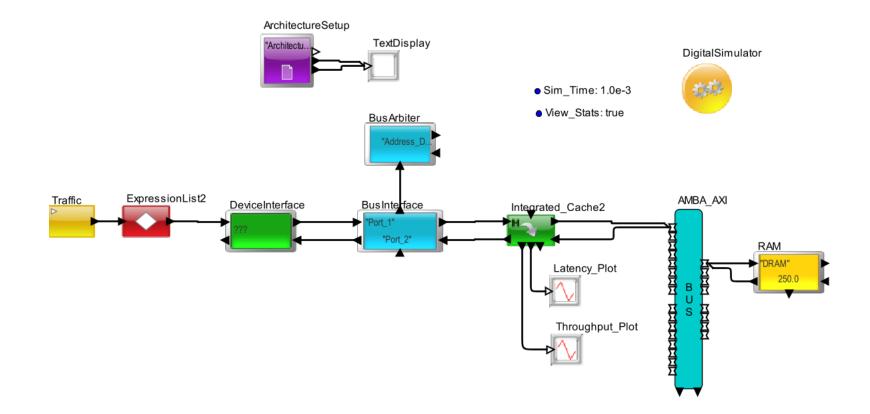
Arbiter_FIX_1_RR_2_CUSTOM_3:	1
Slave_Speeds_Mhz:	z, AXI_Speed_Mhz,AXI_Speed_Mhz, AXI_Speed_Mhz, AXI_Speed_Mhz, AXI_Speed_Mhz}
Extra_Cycles_for_RdReq_WrReq_RdData_WrData:	{0, 0, 0, 0, 0, 0, 0, 0}
Devices_Attached_to_Slave_by_Port:	e_2"},{"Device_3"},{"Device_4"},{"Device_5"},{"Device_6"},{"Device_7"},{"Device_8"}}

- **Device_Attached_to_Slave_by_Port**: List of slave devices connected in each slave port.
- **Arbiter_FIX_1_RR_2_CUSTOM_3**: Arbitration algorithm for multi master single slave cases. (fixed priority uses Fixed_Priority_Array parameter)

DEMO MODEL

Model Location : $VS/demo/memory/Cache_Demo.xml
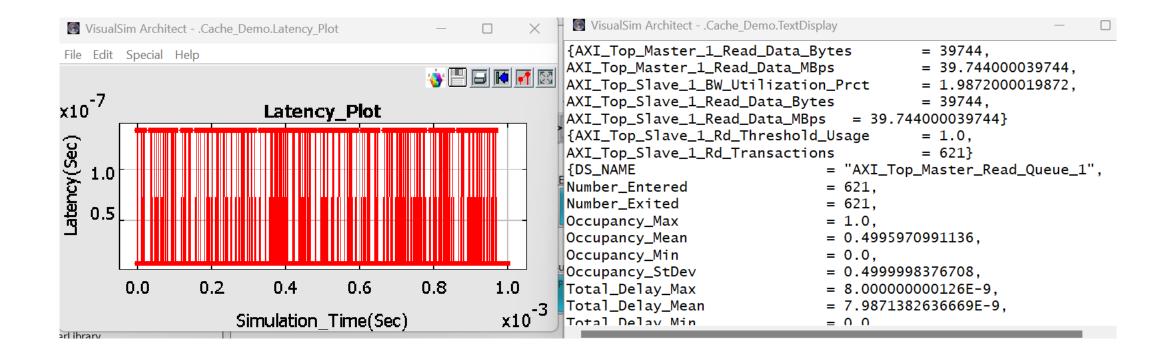
RESULTS FOR PREVIOUS MODEL
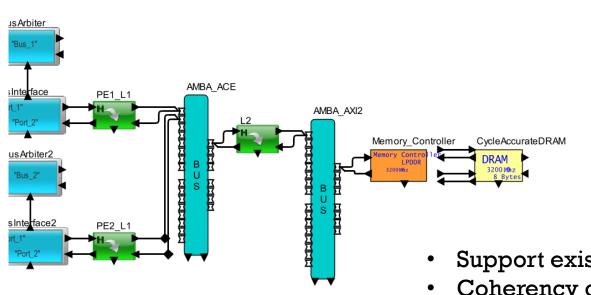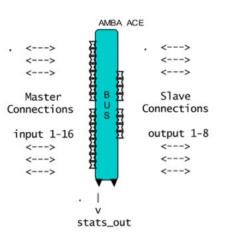
CHI BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT

- Support existing AMBA AXI protocol.
- Coherency can be enabled for specific masters
- If coherency is not enabled for any master, bus acts like a AXI bus and performs non coherent data transfer
  - Coherency channels will not be used.
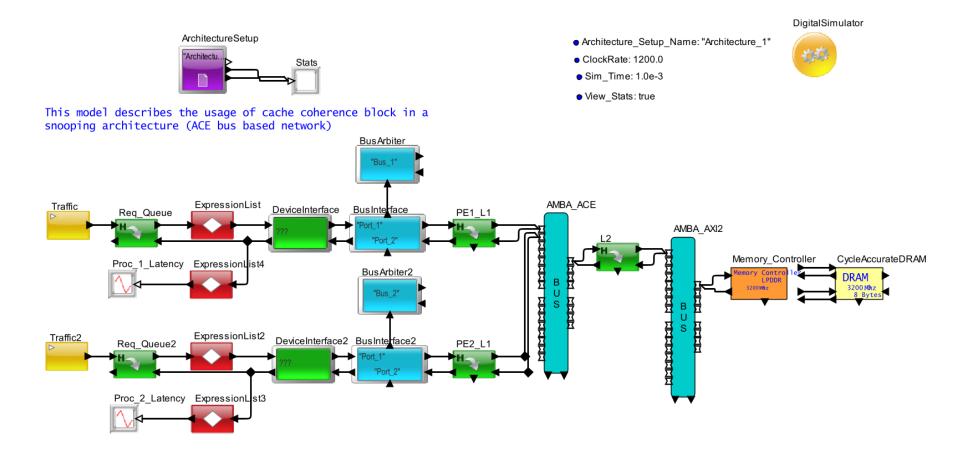
KEY CONFIGURATION

Fixed_Priority_Array: ,15,16},{1,2,3,4,5,6
Slave_First_Word_Flag: true /* Not Active
Custom_Slave_File: "none"
Ports_to_Plot: {0,0} /* master n,
Coherent_Masters: {1,2}
AXI_Sync_Speed_Mhz: 1000.0

[Commit]   [Add]   [Remove]   [Restor

Lis of coherent masters in an array.
Empty array is acceptable, and it implies none of the masters are coherent

DEMO MODEL

Model Location : $VS/demo/Coherency/Snooping_Protocol.xml (2340 version)

RESULTS FOR PREVIOUS MODEL

VisualSim Architect - .Snooping_Protocol.Proc_1_Late...

File   Edit   Special   Help

Proc_1_Latency

x10⁻⁷

Latency (sec): 0.0, 0.5, 1.0, 1.5

Simulation_Time(sec): 0.0, 0.2, 0.4, 0.6, 0.8, 1.0 x10⁻³

VisualSim Architect - .Snooping_Protocol.Proc_2_Lat...

File   Edit   Special   Help

Proc_2_Latency

x10⁻⁷

Latency (sec): 0, 1, 2

Simulation_Time(sec): 0.0, 0.2, 0.4, 0.6, 0.8, 1.0 x10⁻³

VisualSim Architect - .Snooping_Protocol.Stats

```
{Coherence_Cache_Proc2_L1_A_Hit_Ratio       = 83.3766233766234,
Coherence_Cache_Proc2_L1_A_Miss_Ratio       = 16.6233766233766,
Coherence_Cache_Proc2_L1_A_Read_MBs         = 0.030112,
Coherence_Cache_Proc2_L1_A_Read_MBs_per_Second    = 60.22400120448
Coherence_Cache_Proc2_L1_A_Total_MBs        = 0.033856,
Coherence_Cache_Proc2_L1_A_Total_MBs_per_Second = 67.71200135424
Coherence_Cache_Proc2_L1_A_Write_MBs        = 0.003744,
Coherence_Cache_Proc2_L1_A_Write_MBs_per_Second = 7.48800014976,
Coherence_Cache_Proc2_L1_Fill_Buffer_Occupancy    = 0,
Coherence_Cache_Proc2_L1_Input_Buffer_Occupancy   = 0,
Coherence_Cache_Proc2_L1_Number_Entered   = 770,
Coherence_Cache_Proc2_L1_Number_Returned        = 770,
Coherence_Cache_Proc2_L1_Snoop_Invalidates      = 147,
Coherence_Cache_Proc2_L1_Snoop_Requests   = 259,
Coherence_Cache_Proc2_L1_Snooped_Count_per_Device       = {},
Coherence_Cache_Proc2_L1_Snooped_Devices        = {},
Coherence_Cache_Proc2_L1_Utilization      = 1.7432000348639}
{Coherence_Cache_L2_Home_A_Hit_Ratio      = 0.0,
Coherence_Cache_L2_Home_A_Miss_Ratio      = 100.0,
Coherence_Cache_L2_Home_A_Read_MBs        = 0.008192,
Coherence_Cache_L2_Home_A_Read_MBs_per_Second     = 16.38400032768
Coherence_Cache_L2_Home_A_Total_MBs       = 0.008192,
Coherence_Cache_L2_Home_A_Total_MBs_per_Second    = 16.38400032768
Coherence_Cache_L2_Home_A_Write_MBs       = 0.0,
Coherence_Cache_L2_Home_A_Write_MBs_per_Second    = 0.0,
Coherence_Cache_L2_Home_Fill_Buffer_Occupancy     = 0,
Coherence_Cache_L2_Home_Input_Buffer_Occupancy    = 0,
Coherence_Cache_L2_Home_Number_Entered    = 128,
Coherence_Cache_L2_Home_Number_Returned   = 128,
Coherence_Cache_L2_Home_Snoop_Invalidates       = 0,
Coherence_Cache_L2_Home_Snoop_Requests    = 0,
Coherence_Cache_L2_Home_Snooped_Count_per_Device        = {},
Coherence_Cache_L2_Home_Snooped_Devices   = {}
```

BRIDGE BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT

- Blocking
- Delay is variable based on data size and speed

KEY CONFIGURATION

Edit parameters for Bridge

Architecture_Name:	"Architecture_1"
Bridge_Name:	"Bridge"
Bridge_Speed_in_Mhz:	100.0
Bridge_Width_in_Bytes:	4
Overhead_Cycles:	1
Bridge_Sync_Speed_in_Mhz:	Bridge_Speed_in_Mhz

Unique Name

Bridge synchronization clock

Width of the bridge

Commit   Add   Remove   Restore Defa

DEMO MODEL
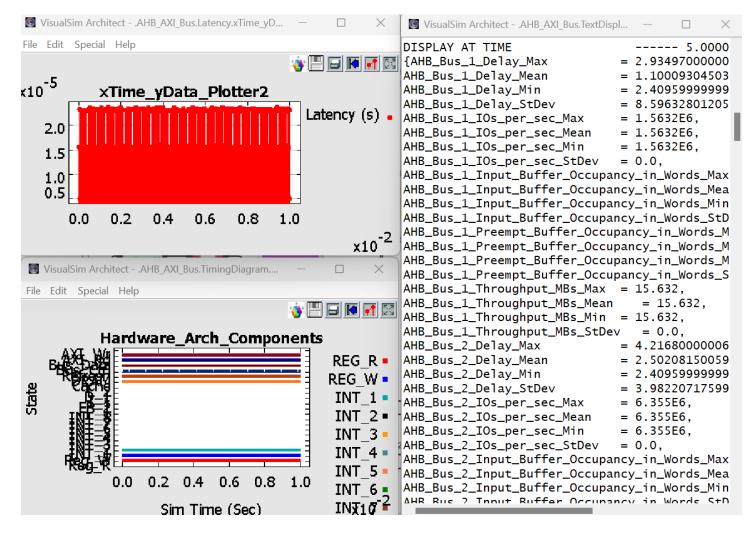
Model Location :
$VS/doc/Training_Material/Architecture/bus/AHB_AXI/AHB_AXI_Bus.xml
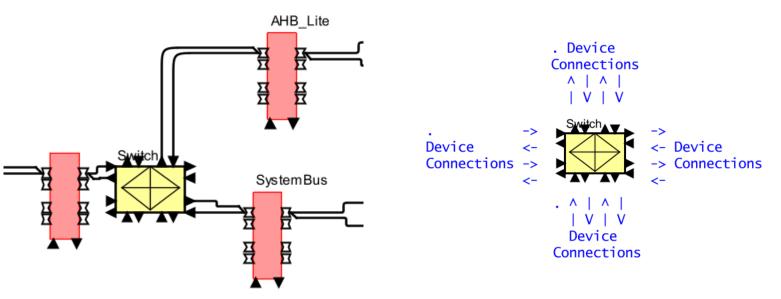
RESULTS FOR THE PREVIOUS MODEL

SWITCH BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT

Blocking

◦ Single interconnect

◦ 4 connected Devices and can add more

◦ Single-cycle delay for Read and Multi-cycle delay for Write

Non blocking

◦ Point-to-point mesh with multiple channels per wire

◦ Basic block contains for 4 device connections; Can be expanded

◦ Single-cycle delay for Read and Multi-cycle delay for Write

KEY CONFIGURATION
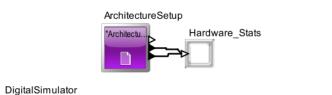
Edit parameters for Switch

Architecture_Name: "Architecture_1"
Switch_Name: "Switch"
Speed_Mhz: 100.0
Width_Bytes: 4
Blocking_Mode: false
_explanation: Hardware_Modeling->Bus_Switch_Ctrl
Overhead_Cycles: 1
Address_Bits: 32
Switch_Levels: 3
Switch_Prefix_Name: "Switch"
Switch_Devices_to_Ports: {{"Device1"},{"Device2"},{"Device3"}

Commit   Add   Remove   R
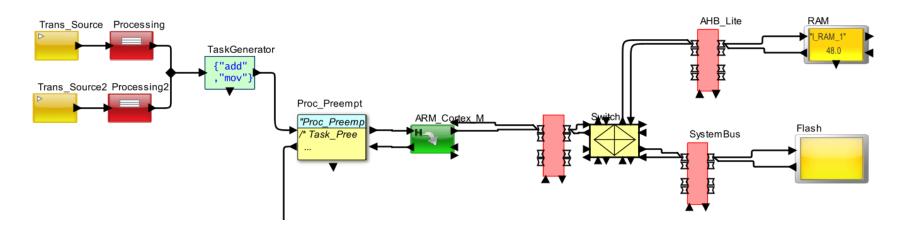
Width of the switch

Blocking or non blocking mode

Unique Name

DEMO MODEL
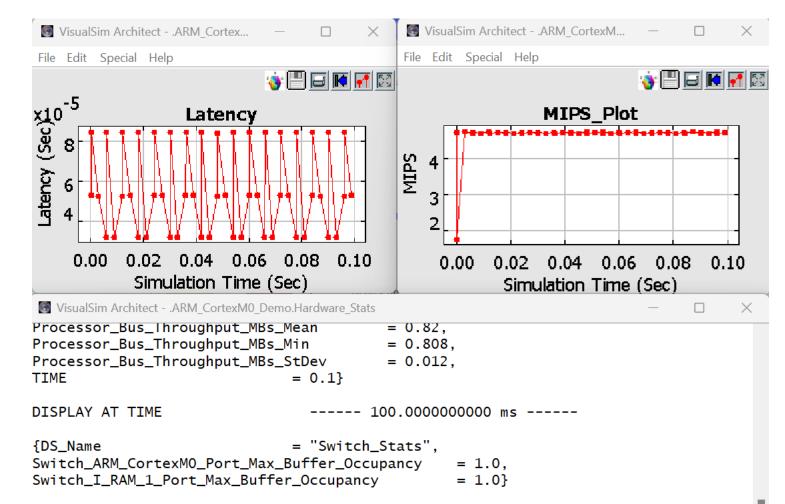
Model Location :
$VS/doc/Training_Material/Architecture/Processor/Cortex_M0/ARM_CortexM0_Demo.xml

RESULTS FOR THE PREVIOUS MODEL
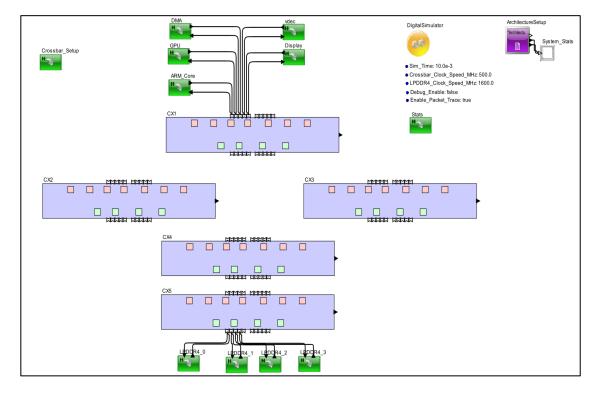
CROSSBAR BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT

- Master devices can be connected to the top multi ports of the block
- Slave blocks can be connected to the bottom multi ports of the block
- The packet can reach a slave device that is in different crossbar by hopping through the crossbar according to the configuration in Crossbar_Setup block.

KEY CONFIGURATION-CROSSBAR

Crossbar_Name:	"CX2"
Architecture_Setup_Name:	"Architecture_1"
Speed_Mhz:	Crossbar_Clock_Speed_MHz
Width_Bytes:	4
Crossbar_QoS:	Rate_Regulation
Buffer_Size:	100
Enable_Flow_Control:	☐
Enable_Hello_Message_Forwardin...	☑
Arbitration_Algorithm:	FCFS
Routing_Option:	Interleave

Width of the crossbar

Crossbar QoS – Rate Regulation or Bandwidth limitation

Buffer Size – Max number of frames that can be stored at the port interface

Routing option – Address based or interleave (detailed configuration in Crossbar_Setup)

KEY CONFIGURATION-CROSSBAR SETUP

Crossbar_Routing_Table:	Crossbar_Name	Port_Name	Connected_Crossbar_Name	Connected_Crossbar_Port	Port_Bandwidth_Mbps	;
	CX1	Slave_1	CX2	Master_1	200.0	;
	CX1	Slave_2	CX2	Master_2	200.0	;
	CX1	Slave_3	CX3	Master_1	200.0	;
	CX1	Slave_4	CX3	Master_2	200.0	;
	CX2	Slave_1	CX4	Master_1	200.0	;
	CX2	Slave_2	CX4	Master_2	200.0	;
	CX3	Slave_3	CX4	Master_3	200.0	;
	CX3	Slave_4	CX4	Master_4	200.0	;
	CX4	Slave_1	CX5	Master_1	200.0	;
	CX4	Slave_2	CX5	Master_2	200.0	;
	CX4	Slave_3	CX5	Master_3	200.0	;
	CX4	Slave_4	CX5	Master_4	200.0	;
	CX5	Slave_1	Wired	DRAM_0	200.0	;
	CX5	Slave_2	Wired	DRAM_1	200.0	;
	CX5	Slave_3	Wired	DRAM_2	200.0	;
	CX5	Slave_4	Wired	DRAM_3	200.0	;
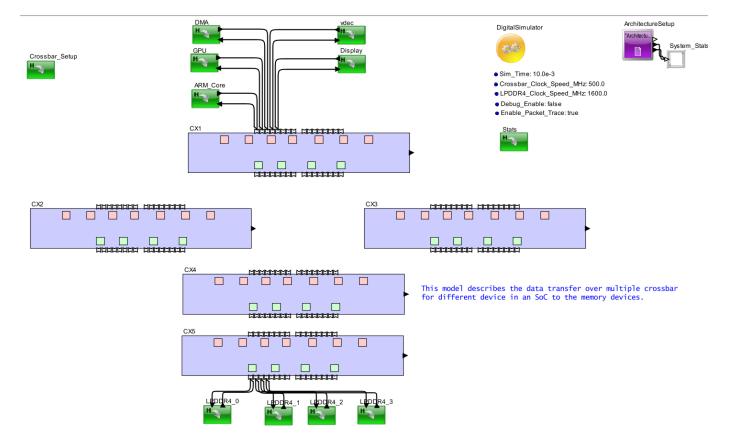
Specifies the Crossbar routing and bandwidth configuration

Crossbar_Settings_Table:	Crossbar_Name	Port_Address
	CX1	{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{"0x00","0x0000000050000"},{"0x00000...
	CX2	{},{}
	CX3	{},{}
	CX4	{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{"0x00","0x005000000000"},{"0x005000...
	CX5	{},...

Specifies the addressing parameters and interleave settings

DEMO MODEL
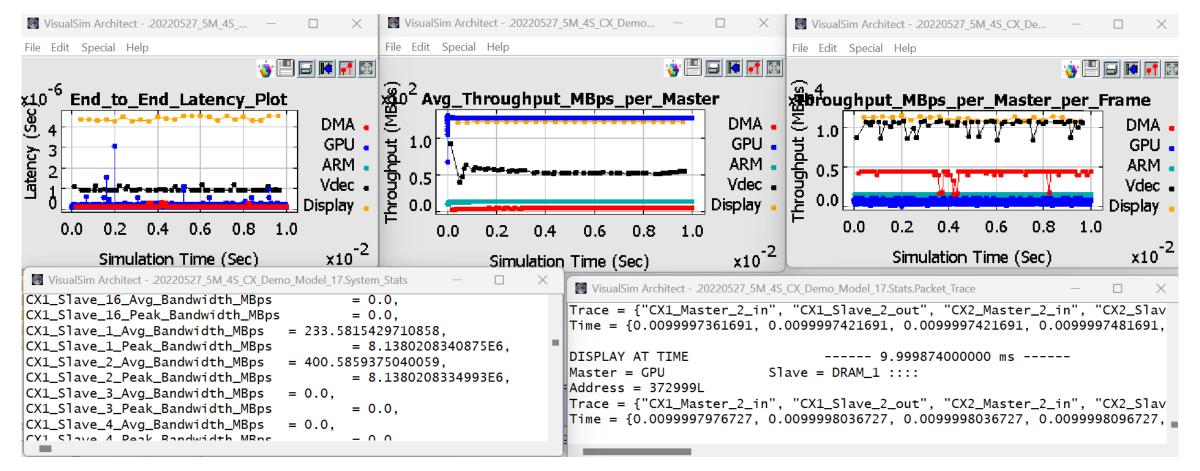
Model Location :
$VS/demo/NoC/Crossbar/20220527_5M_4S_CX_Demo_Model_17.xml

RESULTS FOR THE PREVIOUS MODEL
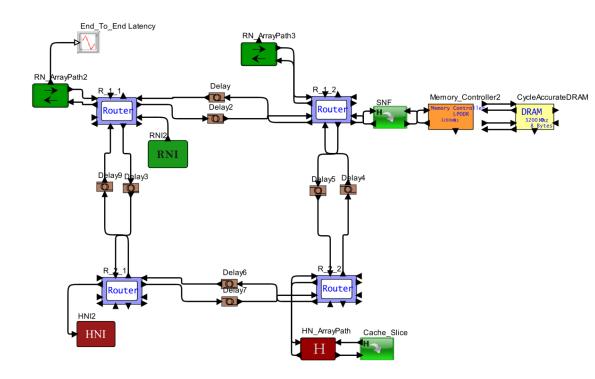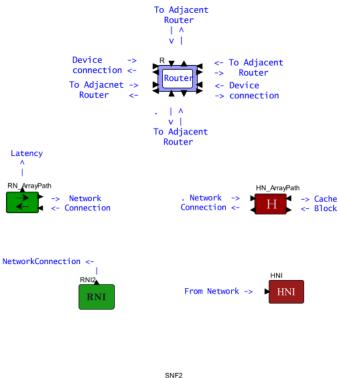
GENERIC NOC BLOCKS AND ITS USAGE

PORTS AND HOW TO CONNECT

KEY CONFIGURATION

Router

Edit parameters for R_1_1

Ingress_Buffer_Size:	Ingress_Buf
VC_Buffer_Size:	VC_Buf
Router_Address:	"R_1_1"
Router_Frequency:	Router_Frequency
Node_Name:	"R_1_1"
VLAN_Q:	4
Router_Coordinate:	{1,1}
Router_Queue_Length:	600
Width:	8

Commit   Add   Re

VC size and device interface queue size

Router Coordinates

Router width in bytes
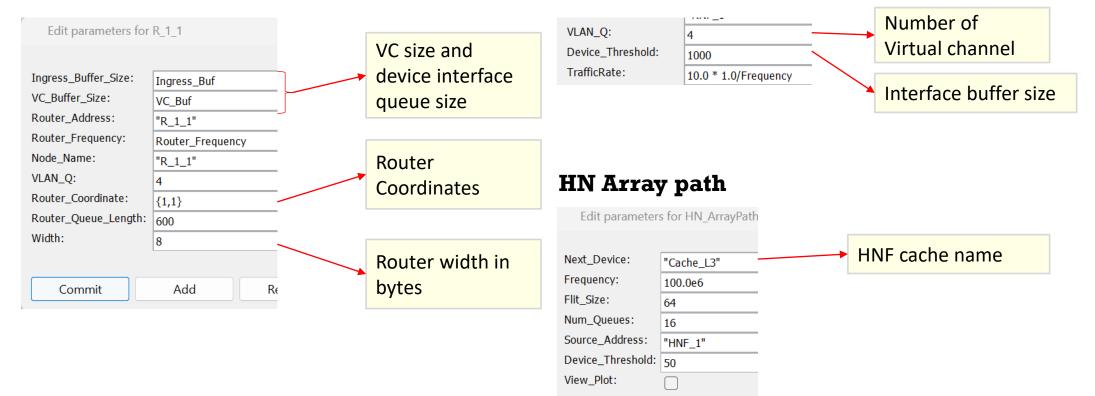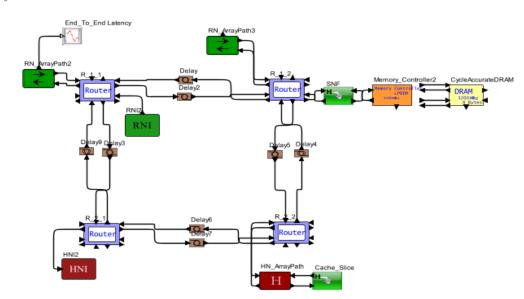
RN Array Path, RNI, HNI and SNF

VLAN_Q:	4
Device_Threshold:	1000
TrafficRate:	10.0 * 1.0/Frequency

Number of Virtual channel

Interface buffer size

HN Array path

Edit parameters for HN_ArrayPath

Next_Device:	"Cache_L3"
Frequency:	100.0e6
Flit_Size:	64
Num_Queues:	16
Source_Address:	"HNF_1"
Device_Threshold:	50
View_Plot:	☐

HNF cache name

DEMO MODEL
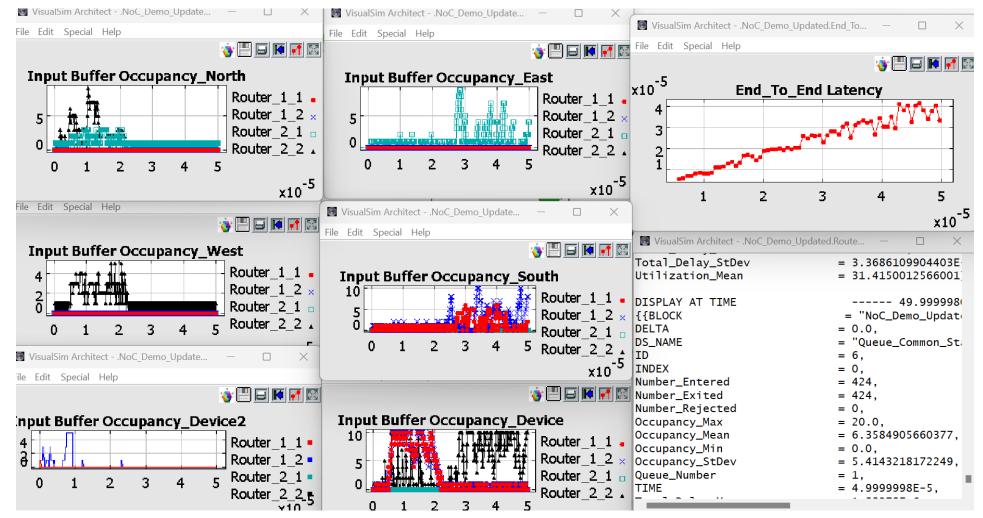
Model Location : $VS/demo/networking/Noc/NoC_Demo_Updated.xml

RESULTS FOR PREVIOUS MODEL
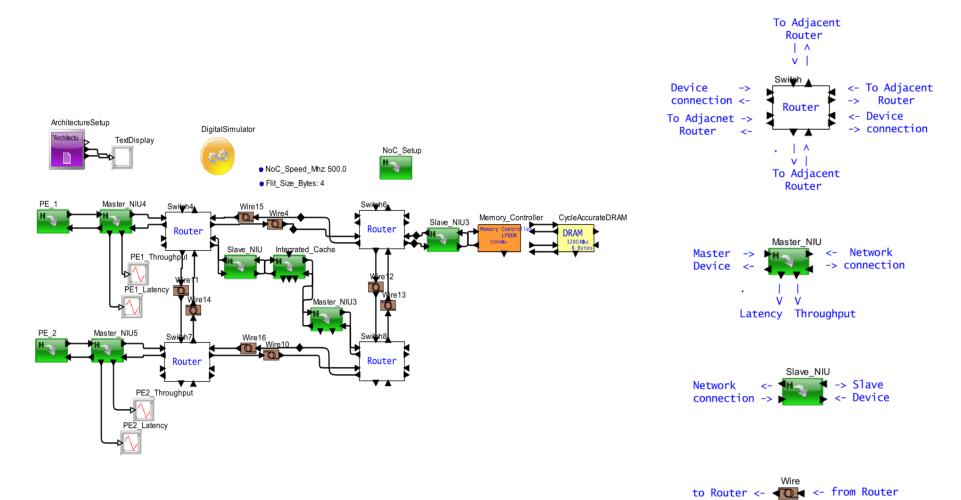
ARTERIS NOC BLOCKS AND ITS USAGE

HOW TO CONNECT?
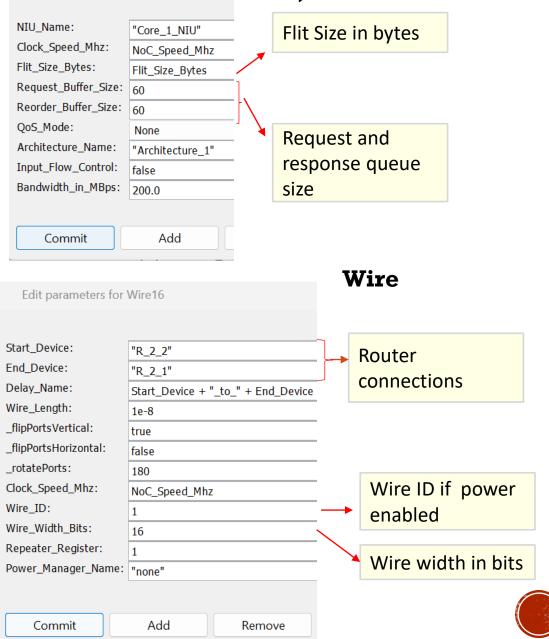
KEY CONFIGURATION

Switch (Router)

Edit parameters for Switch4
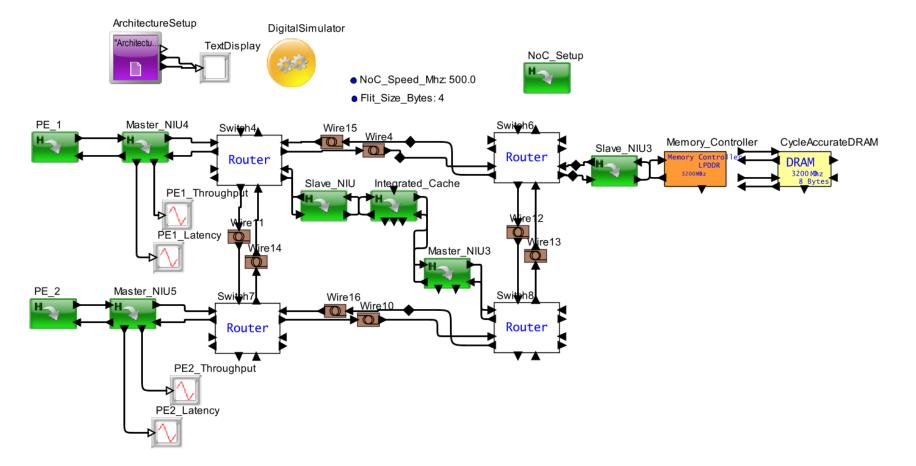
Router_Speed_Mhz:	NoC_Speed_Mhz
Node_Name:	"R_"+x_val+"_"+y_val /* Do not Modify */
Power_Manager_Name:	"none"
x_val:	1
y_val:	1
Topology:	Mesh
No_of_Routers:	4 /* configure only in Loop Topology*/
Priority_Enable:	true
Stats_Enable:	☑
Interconnect_QoS:	Bandwidth_Limiter
No_Of_VC_Per_Port:	1
Bandwidth_per_Port_Mbps:	{200.0,200.0,200.0,200.0,200.0,200.0} /*
Interface_Buffer_Size:	500

Commit   Add   Remove   Restore

Router Coordinates

Device Interface and Virtual Channel Buffer Size

NIU (Network Interface Unit)

Edit parameters for Master_NIU4

NIU_Name:	"Core_1_NIU"
Clock_Speed_Mhz:	NoC_Speed_Mhz
Flit_Size_Bytes:	Flit_Size_Bytes
Request_Buffer_Size:	60
Reorder_Buffer_Size:	60
QoS_Mode:	None
Architecture_Name:	"Architecture_1"
Input_Flow_Control:	false
Bandwidth_in_MBps:	200.0

Commit   Add

Flit Size in bytes

Request and response queue size

Wire

Edit parameters for Wire16

Start_Device:	"R_2_2"
End_Device:	"R_2_1"
Delay_Name:	Start_Device + "_to_" + End_Device
Wire_Length:	1e-8
_flipPortsVertical:	true
_flipPortsHorizontal:	false
_rotatePorts:	180
Clock_Speed_Mhz:	NoC_Speed_Mhz
Wire_ID:	1
Wire_Width_Bits:	16
Repeater_Register:	1
Power_Manager_Name:	"none"

Commit   Add   Remove

Router connections

Wire ID if power enabled

Wire width in bits

DEMO MODEL

Model Location : $VS/demo/NoC/Arteris/Noc_Arteris_Demo.xml

RESULTS FOR PREVIOUS MODEL
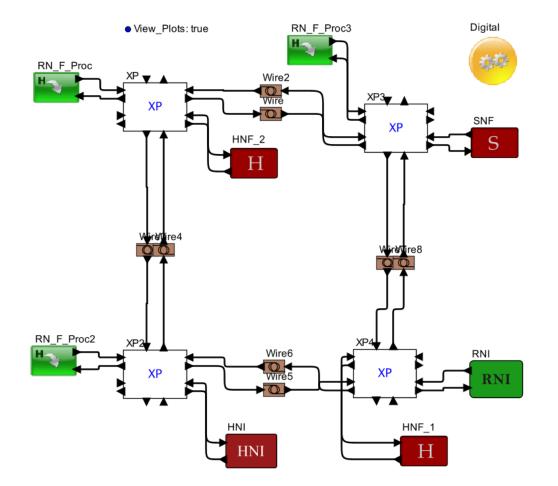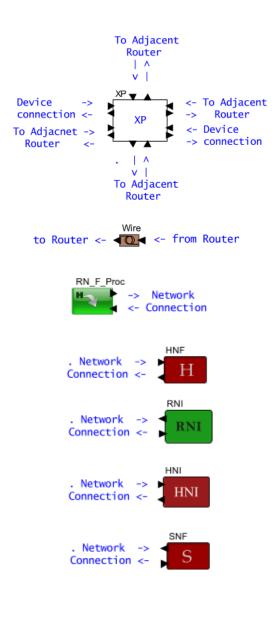
CMN600 BLOCKS AND ITS USAGE
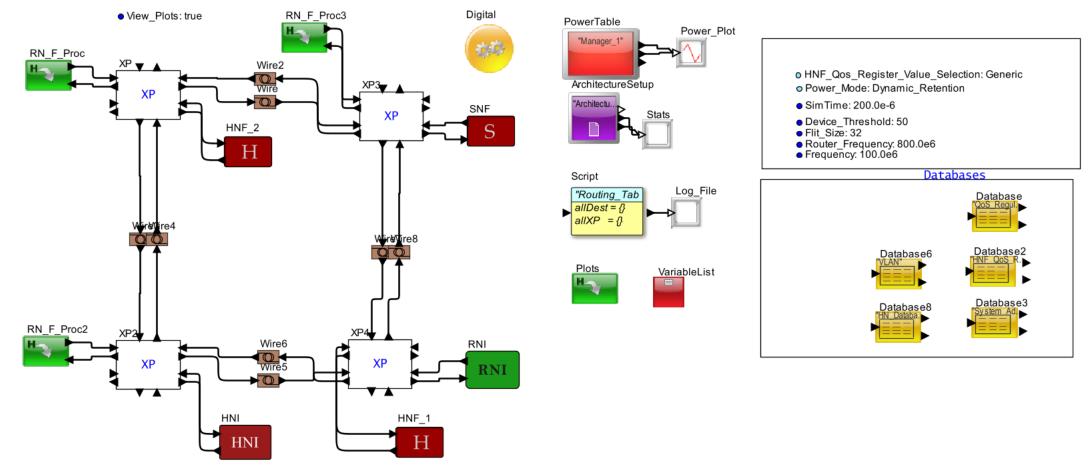
PORTS AND HOW TO CONNECT

KEY CONFIGURATION

RNF, RNI, HNI and SNF

VLAN_Q:	4/*Total number of VLAN_Q available*/
Device_Threshold:	Device_Threshold
TrafficRate:	10.0 * 1.0/Frequency
Address_Low:	0
Address_High:	1023
Random_Address:	lse/*true for generating random addr re
Request_Priority:	"All"/*ALL,HH,H,M,L are the available o
QoS_Regulator_Mode:	None
Flit_Size:	Flit_Size

- No of VC in the router
- Device buffer size
- Packet Priority
- Flit size in bytes

XP (Router)

Edit parameters for XP

Ingress_Buffer_Size:	1000
VC_Buffer_Size:	10
Router_Frequency:	Router_Frequency
Node_Name:	"R_"+x_val+"_"+y_val
Power_Manager_Name:	"Manager_1"
Single_bit_error_ratio:	0.4
Double_bit_error_ratio:	0.2
VLAN_Q:	4
x_val:	1
y_val:	1
Stats_Enable:	☐

| Commit | Add | R |

- Size of Virtual Channel and device interface queue
- Bit error ratio
- Number of VC
- Router Coordinates

HNF

Flit_Size:	Flit_Size
Num_Resources:	50/*Total number of resources available for POCQ*/
Source_Address:	"HNF_2"
Device_Threshold:	Device_Threshold
Power_Manager_Name:	"Manager_1"
SLC_SIZE_KB:	256
View_Plots:	☑
VLAN_Q:	4/*Total number of VLAN_Q available*/

- HNF input Queue size
- HNF cache size

Wire

Edit parameters for Wire2

Start_Device:	"R_1_2"
End_Device:	"R_1_1"
Delay_Name:	Start_Device + "_to_" + End_Device
Wire_Length:	1e-8
_flipPortsVertical:	true
_flipPortsHorizontal:	false
_rotatePorts:	180
Clock_Speed:	200e6
Delay_Cycles:	1
Wire_ID:	2

- Router connections
- Delay cycle for flits

DEMO MODEL
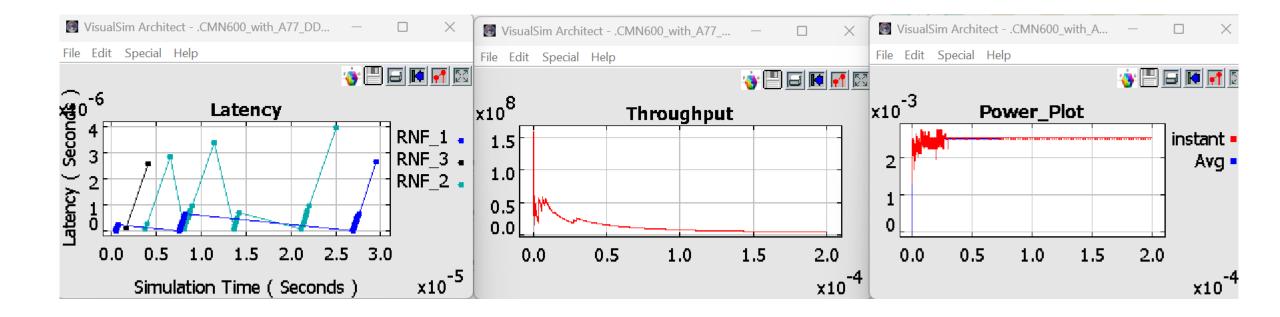
Model Location : $VS/demo/NoC/Corelink/CMN600_with_A77_DDR5.xml
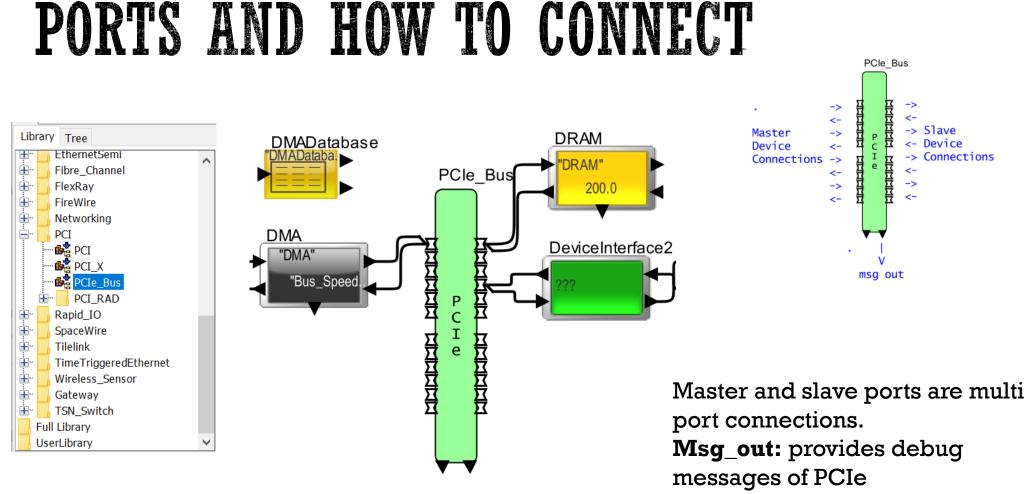
RESULTS FOR PREVIOUS MODEL

PCIE BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT

Master and slave ports are multi port connections.
Msg_out: provides debug messages of PCIe
Stats_out: not used in the current version

KEY CONFIGURATION

Edit parameters for PCIe_Bus —
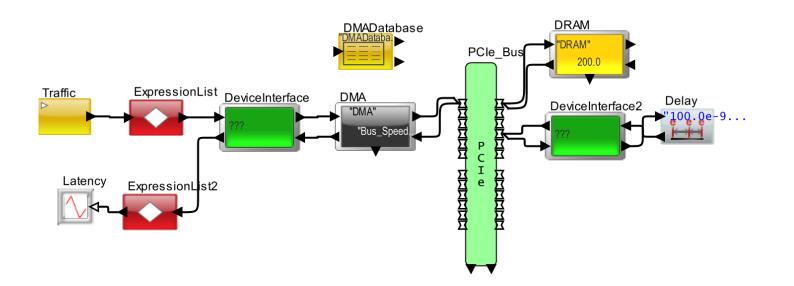
Architecture_Name:	"Architecture_1"
Bus_Name:	"PCIe_1"
Number_of_Lanes:	16 /* Can be an array */
Slave_Buffer:	512 /* Max Bytes @ Slave */
Master_Buffer:	512 /* Max Bytes @ Master */
_explanation:	Interfaces and Buses->PCI->PCIe_Bus
Header_Bytes:	16 /* 32 Bit Mode, includes CRC Bytes */
Number_of_Ports:	{12, 12} /* Master, Endpoint Ports */
BER:	1E-11
Max_Payload_Size:	64 /* Write, Read Data */
Max_Payload_Req_Size:	128 /* Read Requests */
Read_to_Write_Ratio:	0.5 /* 0.0 to 1.0 */
Devices_Attached_to_Slaves:	)RAM_3"},{"Dev_4"},{"Dev_5"},{"Dev_6"},{"Dev_7"},{"Dev_8"},{"Dev_9"},{"Dev_10"},{"De
Root_Complex_Flow_Control:	{false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false}
Endpoint_Flow_Control:	{false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false}
Enable_Plots:	☑
Bit_64_Mode:	☑
NumOfRetry:	4
Timeout:	6E-6
PCIe_MBps:	PCIe_Gen_1

Can be common or array for each port in order starting from top-left and continuing through top-right

Required: List of all the Devices connected to each End-Point

Required: Determines speed

DEMO MODEL
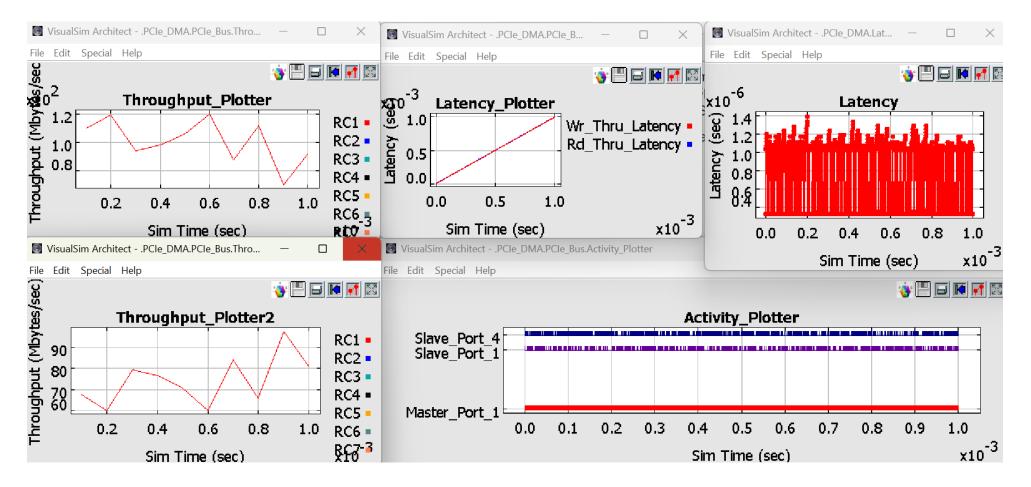
This model describes the usage of DMA with the PCIe Bus.

Model Location : $VS/doc/Training_Material/Architecture/bus/PCIe/PCIe_DMA.xml

RESULTS FOR THE PREVIOUS MODEL
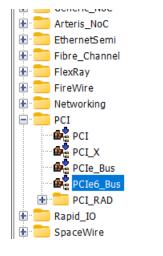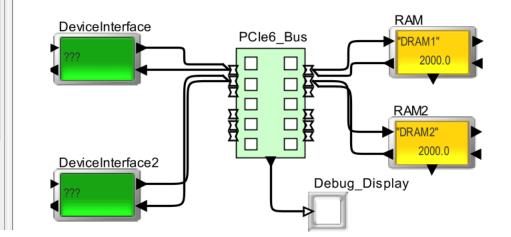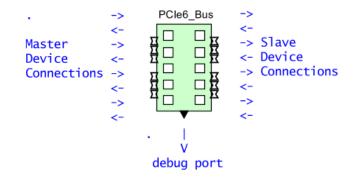
PCIE6.0 BLOCK AND ITS USAGE

PORTS AND HOW TO CONNECT

Master and slave connections use multi port for input and output.
Debug_port: provides debug message of the PCIe6 bus
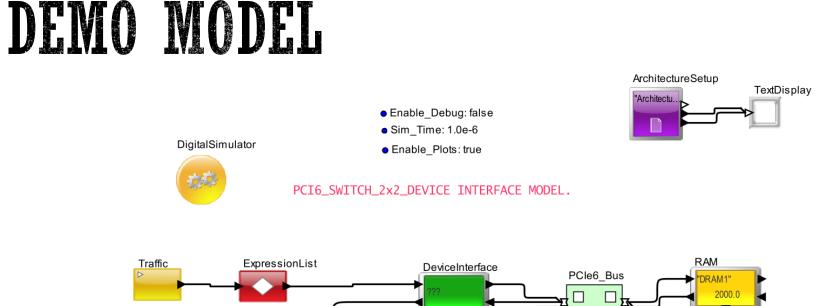
KEY CONFIGURATION
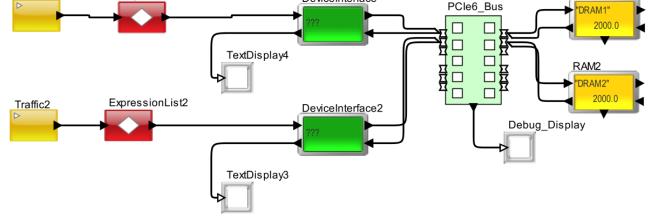
Edit parameters for PCIe6_Bus

Architecture_Setup_Name:	"Architecture_1"
PCIe_Switch_Name:	"PCIe_Switch"
Max_Read_Req_Size_Bytes:	4096
Flit_Size_Bytes:	256
Number_of_Lanes:	16
Buffer_Size_Bytes:	{4096,4096} //Rx,Tx
Overhead_Cycles:	0
Devices_Attached_to_Ports:	)ev_4"},{"Dev_5"},{"Dev_6"},{"Dev_7","DRAM","Reg"},{"Dev_8"},{"Dev_9"},{"Dev_10"},{"Dev_11"},{"Dev_12"}}
Enable_Debug:	true
BER:	1.0e-11
NumOfRetry:	4
Timeout:	6E-6
Enable_Master_Flow_Control:	☐
Enable_Slave_Flow_Control:	☐
Replay_Buffer_Size_Bytes:	1024
Number_Of_Successive_Acks:	3
Enable_Selective_Ack:	true
Enable_Hello_Msg_Forwarding:	true

[Commit] [Add] [Remove] [Restore Defaults] [Preferences] [Help] [Cancel]

Maximum request size of a packet

Number of Lanes

Maximum bytes that can be stored in Rx and Tx buffer

List of Devices connected to each of the port

DEMO MODEL

Model Location : $VS/doc/Training_Material/Architecture/bus/PCIe6/PCIe6_Device Interface2x2_base_model.xml

RESULTS FOR THE PREVIOUS MODEL

VisualSim Architect - .PCIe6_Device_Interface2x2_base_model.TextDisplay

```
PCIe_Switch_PCIe_Switch_Port_12_Tx_MBps    = 0.0,
PCIe_Switch_PCIe_Switch_Port_1_Drop_Count          = 9372,
PCIe_Switch_PCIe_Switch_Port_1_Rx_MBps    = 88152.08815208815,
PCIe_Switch_PCIe_Switch_Port_1_Total_MBps          = 178252.17825217824,
PCIe_Switch_PCIe_Switch_Port_1_Tx_MBps    = 90100.0901000901,
PCIe_Switch_PCIe_Switch_Port_1_to_Port_8_Max_Latency      = 1.23986E-7,
PCIe_Switch_PCIe_Switch_Port_1_to_Port_8_Mean_Latency     = 1.12386033928
PCIe_Switch_PCIe_Switch_Port_1_to_Port_8_Min_Latency      = 4.538E-9,
PCIe_Switch_PCIe_Switch_Port_2_Drop_Count          = 9961,
PCIe_Switch_PCIe_Switch_Port_2_Rx_MBps    = 88684.08868408868,
PCIe_Switch_PCIe_Switch_Port_2_Total_MBps          = 178844.17884417885,
PCIe_Switch_PCIe_Switch_Port_2_Tx_MBps    = 90160.09016009016,
PCIe_Switch_PCIe_Switch_Port_2_to_Port_7_Max_Latency      = 1.24181E-7,
PCIe_Switch_PCIe_Switch_Port_2_to_Port_7_Mean_Latency     = 1.0468030285
PCIe_Switch_PCIe_Switch_Port_2_to_Port_7_Min_Latency      = 4.738E-9,
PCIe_Switch_PCIe_Switch_Port_3_Drop_Count          = 0,
PCIe_Switch_PCIe_Switch_Port_3_Rx_MBps    = 0.0,
PCIe_Switch_PCIe_Switch_Port_3_Total_MBps          = 0.0,
PCIe_Switch_PCIe_Switch_Port_3_Tx_MBps    = 0.0,
PCIe_Switch_PCIe_Switch_Port_4_Drop_Count          = 0,
PCIe_Switch_PCIe_Switch_Port_4_Rx_MBps    = 0.0,
PCIe_Switch_PCIe_Switch_Port_4_Total_MBps          = 0.0,
PCIe_Switch_PCIe_Switch_Port_4_Tx_MBps    = 0.0,
PCIe_Switch_PCIe_Switch_Port_5_Drop_Count          = 0,
PCIe_Switch_PCIe_Switch_Port_5_Rx_MBps    = 0.0,
```

INTEGRATED CACHE AND ITS USAGE

Integrated Cache

Integrated Cache can be used as L1(Instruction and/or Data), L2 and L3 cache in both stochastic and cycle accurate mode.

Stochastic Mode:

◦ Hit or miss of input request will be determined by the instruction hit ratio/data hit ratio.

◦ If it is a hit, request will be processed and response will be returned to the source. If it is a miss, request will be sent to next level cache or memory to fetch whole block of data, while the request is waiting in the buffer,.

Address_Based Mode

◦ Hit or miss of the input request will be determined by the availability of the requested address in the cache.

◦ If it is a hit, request will be processed and response to the requested address will be returned. If it is a miss, whole block of address range will be fetched from next block of memory and the request waiting in the buffer will be processed.

Flow control:

◦ User can run the model either with flow control or without flow control. By default block will be used in without flow control.

◦ Input flow control can be achieved by including a field named "Event_Name" in the input data structure and a TIMEQ to trigger the next request. The next request will be triggered only when the data is processed by the cache.

◦ Output flow control can be achieved by setting the "Output_Flow_Control" parameter as true.

Integrated_Cache

Required Parameters

- Outstanding_Requests
- Number of Outstanding misses and prefetches
- Number of Hit before Prefetch

Edit parameters for L2_Cache

Cache_Name:	Cluster_Name+"_L2_Cache"
Cache_Speed_Mhz:	L2_Cache_Speed_Mhz
Cache_Width_Bytes:	L2_Cache_Width_Bytes
Cache_Size_Bytes:	L2_Cache_Size_Bytes
Block_Size_Bytes:	Cache_Block_Size_Bytes
Cache_Type:	D_Cache
Stochastic_or_Address_Based:	Address_Based
Hit_Ratio:	0.8
Loop_Ratio:	0.2
Overhead_Cycles:	0 /* For read and write set it as array {Read,Write}, eg: {1,3}*/
Input_Flow_Control:	☐
Req_Buffer_Size:	16
Output_Flow_Control:	☐
Total_Outstanding:	10
N_Way_Associativity:	8
Cache_Replacement_Policy:	Pseudo-LRU
Cache_Write_Policy:	Write_Back
Inclusion_Policy:	NINE /* Mandatory for L1 cache */
Miss_Memory_Name:	Next_Level_memory_Name
Power_Manager_Name:	"None" /*To analyse power, link the manager name */
No_of_Statistics:	cast(1,Number_of_samples)
Word_Access:	First_Word
Total_Outstanding_Miss:	10
N_Hits_Before_Prefetch:	1
Main_Memory_Size_KB:	4*1024*1024
Architecture_Name:	Architecture_Setup_Name

Commit | Add | Remove | Restore Defaults | Preferences | Help | Cancel

Integrated Cache Statistics

Number of Statistics samples can be generated during the simulation time using the parameter "No_of_Statistics'

VisualSim Architect - .Cache_and_mem.L2_Cache_Statistics

```
DISPLAY AT TIME                        ------ 500.2795000 us ------
{A_Hit_Ratio                           = 93.2203389830508,
A_Miss_Ratio                           = 6.7796610169492,
A_Prefetch_Ratio                       = 0.0,
A_Read_MBs                             = 0.181248,
A_Read_MBs_per_Second                  = 362.4960072499201,
A_Total_MBs                           = 0.193536,
A_Total_MBs_per_Second                 = 387.0720077414401,
A_Write_MBs                           = 0.012288,
A_Write_MBs_per_Second                 = 24.57600049152,
BLOCK                                  = "L2_Cache",
Buffer_Occupancy                       = 0,
DS_NAME                                = "Header_Only",
ID                                     = 1,
Number_Entered                         = 177,
Number_Returned                        = 177,
TIME                                   = 5.002795E-4, |
Utilization                            = 12.1845002436899}
```

STOCHASTIC CACHE AND IT'S USAGE

Stochastic Cache

Emulate a cache in stochastic architecture mode where addresses are not available- Used with hit=ratio

Handles
o Request are Queued
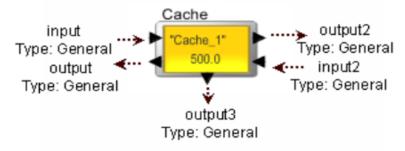
o Measures actual Cache hit-miss ratio

o Cache Prefetch,

o Read/Write

o Cache miss activity to the next level of memory
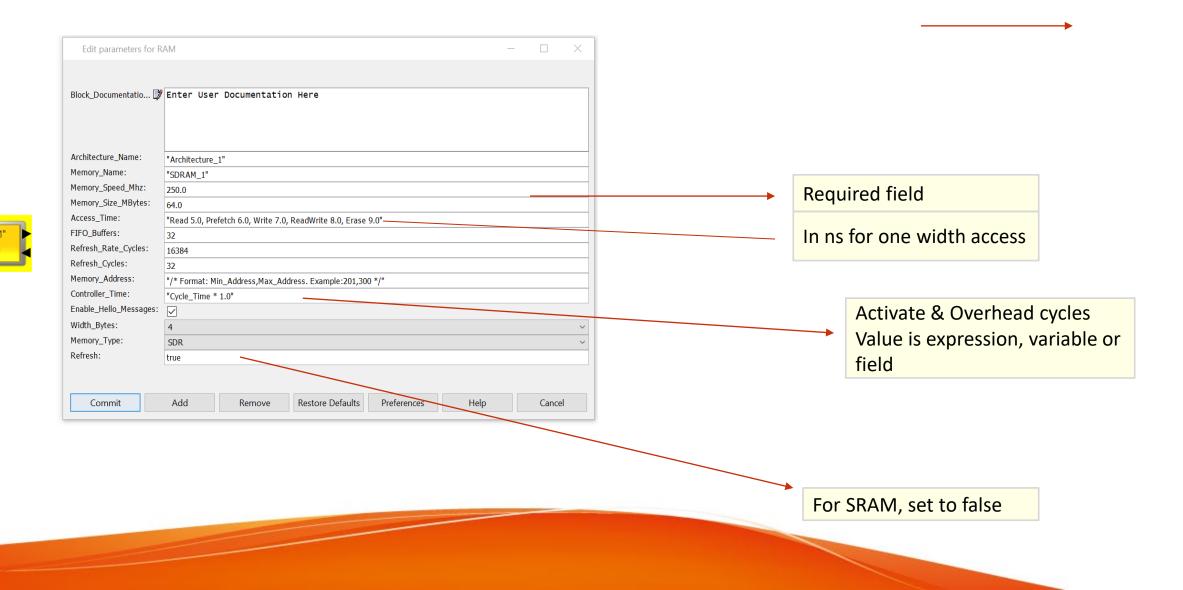
Configuration

Cache_Speed
is necessary

Route the requests to the next level of memory to access when a cache miss occurs or a prefetch is requested

Number of outstanding requests that need to be processed

Expression for the cache hit using RegEx language.
- *true*, then the task had a cache hit
- else a miss occurred.

STOCHASTIC MEMORY AND IT'S USAGE

Stochastic Memory

RAM

"SDRAM_1"

250.0

Edit parameters for RAM

Block_Documentatio... Enter User Documentation Here

Architecture_Name: "Architecture_1"
Memory_Name: "SDRAM_1"
Memory_Speed_Mhz: 250.0
Memory_Size_MBytes: 64.0
Access_Time: "Read 5.0, Prefetch 6.0, Write 7.0, ReadWrite 8.0, Erase 9.0"
FIFO_Buffers: 32
Refresh_Rate_Cycles: 16384
Refresh_Cycles: 32
Memory_Address: "/* Format: Min_Address,Max_Address. Example:201,300 */"
Controller_Time: "Cycle_Time * 1.0"
Enable_Hello_Messages: ☑
Width_Bytes: 4
Memory_Type: SDR
Refresh: true

Commit   Add   Remove   Restore Defaults   Preferences   Help   Cancel

Required field

In ns for one width access

Activate & Overhead cycles
Value is expression, variable or field

For SRAM, set to false

RAM

1. Models the memory controller and memory array

2. Handles
- Read
- Write
- Refresh
- Erase

3. Applications
- ROM, RAM, SRAM, DRAM or SDRAM
- DDR, DDR2, DDR3
- SDR, QDR
- VRAM, Direct Rambus, PSRAM, SGRAM
- NAND and NOR flash

Important Concepts

1. Controller Time – Scheduling + Activation

- Cycle Time = 1/Memory Speed Mhz

2. Access Time
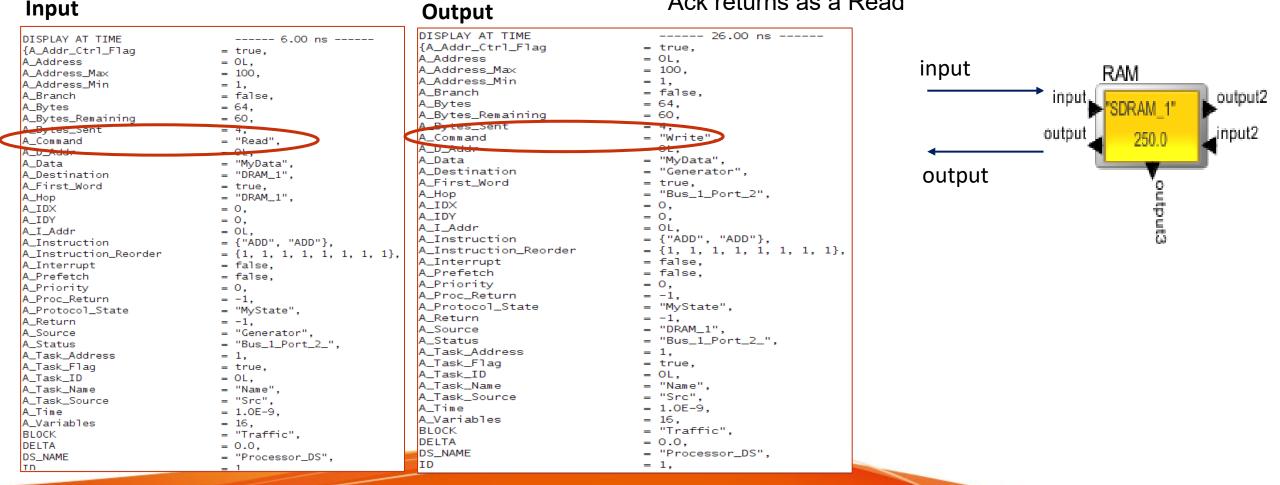
- Access time for Read, Write, Prefetch and Erase is in nanoseconds
  Example : Read 1000.0/Memory_Speed_Mhz
- Default value
  - Read 5.0
  - Prefetch 6.0
  - Write 7.0
  - ReadWrite 8.0
  - Erase 9.0

Read Operation

Request goes as a Read,
Response returns as a Write

Request goes as a Write,
Ack returns as a Read

Input

```
DISPLAY AT TIME                ------ 6.00 ns ------
{A_Addr_Ctrl_Flag              = true,
A_Address                      = OL,
A_Address_Max                  = 100,
A_Address_Min                  = 1,
A_Branch                       = false,
A_Bytes                        = 64,
A_Bytes_Remaining              = 60,
A_Bytes_Sent                   = 4,
A_Command                      = "Read",
A_D_Addr                       = OL,
A_Data                         = "MyData",
A_Destination                  = "DRAM_1",
A_First_Word                   = true,
A_Hop                          = "DRAM_1",
A_IDX                          = 0,
A_IDY                          = 0,
A_I_Addr                       = OL,
A_Instruction                  = {"ADD", "ADD"},
A_Instruction_Reorder          = {1, 1, 1, 1, 1, 1, 1, 1},
A_Interrupt                    = false,
A_Prefetch                     = false,
A_Priority                     = 0,
A_Proc_Return                  = -1,
A_Protocol_State               = "MyState",
A_Return                       = -1,
A_Source                       = "Generator",
A_Status                       = "Bus_1_Port_2_",
A_Task_Address                 = 1,
A_Task_Flag                    = true,
A_Task_ID                      = OL,
A_Task_Name                    = "Name",
A_Task_Source                  = "Src",
A_Time                         = 1.0E-9,
A_Variables                    = 16,
BLOCK                          = "Traffic",
DELTA                          = 0.0,
DS_NAME                        = "Processor_DS",
ID                             = 1
```
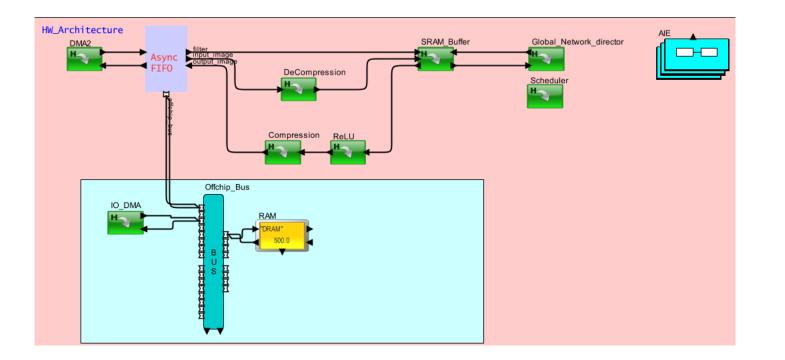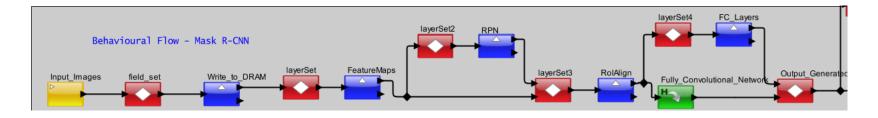
Output

```
DISPLAY AT TIME                ------ 26.00 ns ------
{A_Addr_Ctrl_Flag              = true,
A_Address                      = OL,
A_Address_Max                  = 100,
A_Address_Min                  = 1,
A_Branch                       = false,
A_Bytes                        = 64,
A_Bytes_Remaining              = 60,
A_Bytes_Sent                   = 4,
A_Command                      = "Write"
A_D_Addr                       = OL,
A_Data                         = "MyData",
A_Destination                  = "Generator",
A_First_Word                   = true,
A_Hop                          = "Bus_1_Port_2",
A_IDX                          = 0,
A_IDY                          = 0,
A_I_Addr                       = OL,
A_Instruction                  = {"ADD", "ADD"},
A_Instruction_Reorder          = {1, 1, 1, 1, 1, 1, 1, 1},
A_Interrupt                    = false,
A_Prefetch                     = false,
A_Priority                     = 0,
A_Proc_Return                  = -1,
A_Protocol_State               = "MyState",
A_Return                       = -1,
A_Source                       = "DRAM_1",
A_Status                       = "Bus_1_Port_2_",
A_Task_Address                 = 1,
A_Task_Flag                    = true,
A_Task_ID                      = OL,
A_Task_Name                    = "Name",
A_Task_Source                  = "Src",
A_Time                         = 1.0E-9,
A_Variables                    = 16,
BLOCK                          = "Traffic",
DELTA                          = 0.0,
DS_NAME                        = "Processor_DS",
ID                             = 1,
```

input

output

RAM
"SDRAM_1"
250.0
input output2
output input2
output3

DNN (MASK-R-CNN) USAGE

PORTS AND HOW TO CONNECT

- The On chip network (pink surface) is designed as per the Mask R-CNN specification
- The Task Graph in the bottom defines the behavioral flow for software and trigger the hardware devices.
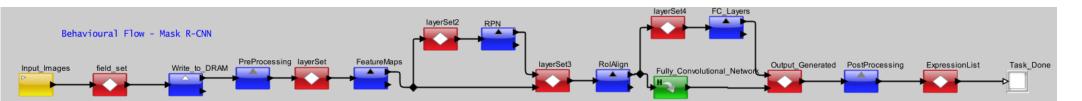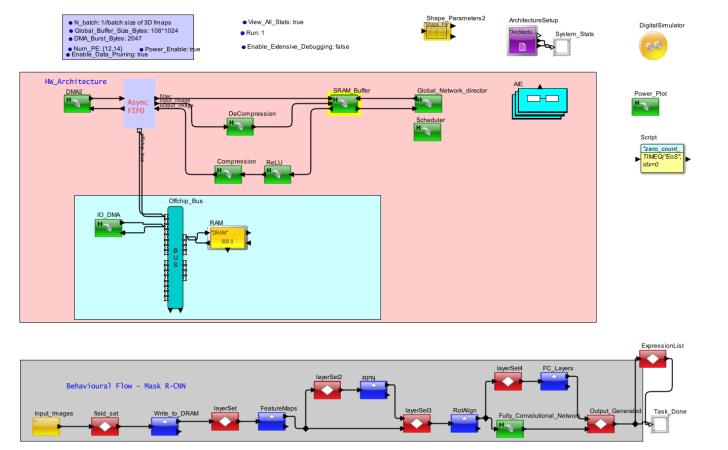
KEY CONFIGURATION

- Each of the layers are defined as different tasks in the task graph and the dependency between them is modeled.
- A database is used to list the layers/functions and the parameters associated with them.
  - These will be used to determine the number of Multiply Accumulate (MAC) operations corresponding to each layer/function

DEMO MODEL

Model Location : $VS/demo/DNN/DNN_Model_Mask_R_CNN.xml

RESULTS FOR THE PREVIOUS MODEL