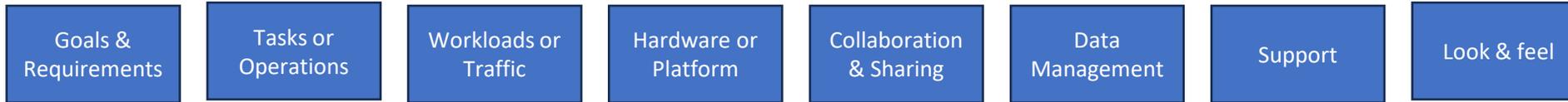




# VISUALSIM TRAINING

---

## Planning



## Modeling



## Simulation and Analysis



## Advanced Features



# Agenda- Part 2: Modeling

---

- Parameters/Variables/DS/Fields
- Data Types
- Regex
- Flow- System, Beh-Arch, Data flow
- Hierarchical/ Class/ DI Blocks
- Traffic
- Results
- Abstraction

# Variable Vs Parameter Vs Data Structure

## Variable

1. 3 types- Block(In Script and Expression List only), Local(One window) and Global (Full Model)
2. Values change during simulation
3. Example: Flag, Statistics

## Common

- All standard data types supports by Parameters, Variables and fields
- Viewable by Blocks

## Parameter

1. Fixed for a simulation and modified for next run
2. Example: clock speed, buffer size
3. Used for multi-core and batch simulation

## Data Structure

1. Similar to Struct in C with Fields and values
2. Contains specific information for that Transaction or packet
3. Dynamically add and destroy fields

# Data Structure

---

Data Structure is similar to “struct” in C

Fields of the Data Structure represent

- Data transmitted through the model
- Represents the Control signal info , frames, packets etc.

Data Structures contain 6 standard header

- Block (Source name) and DS\_Name (Template name)
- TIME (creation time-stamp) and ID (sequence number)
- INDEX (integer scratch pad) & DELTA (double scratch pad)

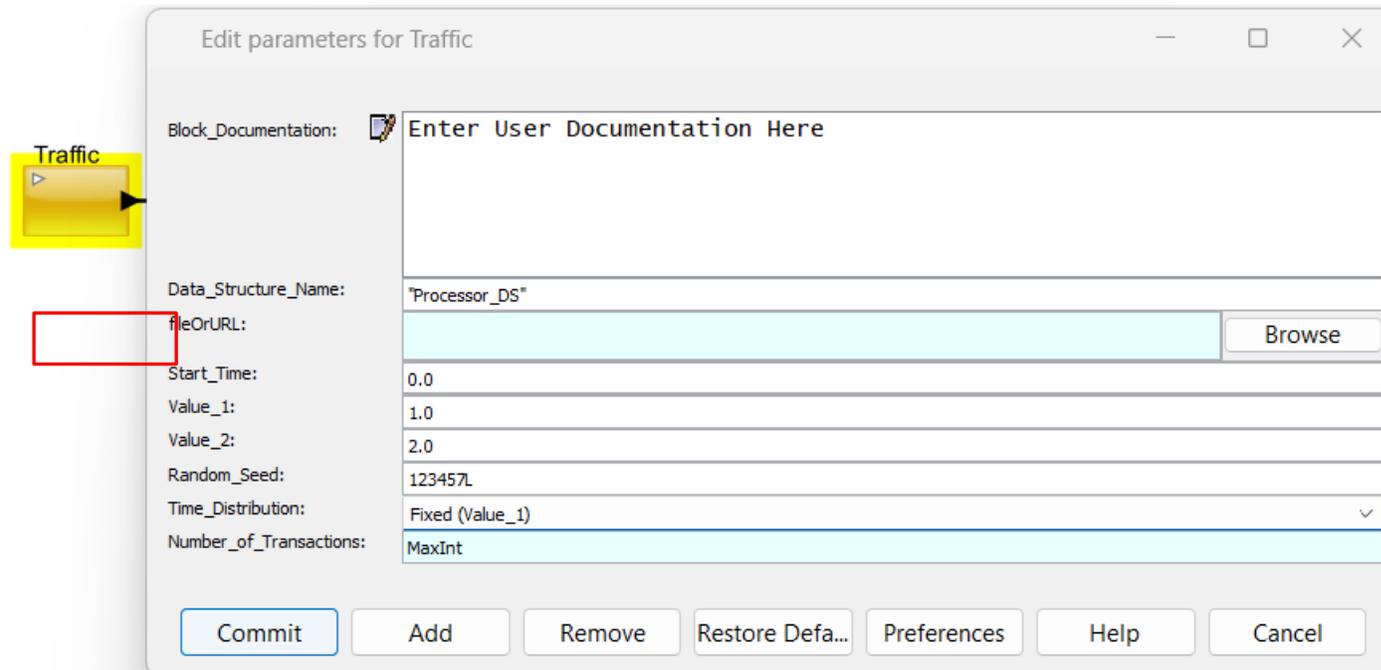
```
DISPLAY AT TIME          ----- 0.0 ps -----
{BLOCK                   = "Traffic",
DELTA                    = 0.0,
DS_NAME                  = "Header_Only",
ID                       = 1,
INDEX                   = 0,
TIME                    = 0.0}
```

# Data Structure Templates

---

- VisualSim Processor, memory -> Processor\_DS
- Network modeling -> Task\_Class
- Non-Processor Hardware -> Hardware\_DS
- Stochastic -> Header

# Processor\_DS - template

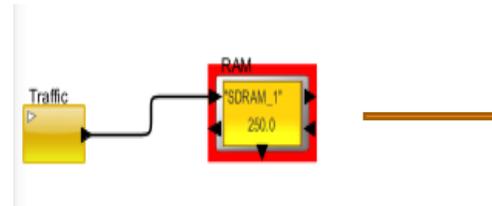


**Replacing the default data structure template which can have 50+ fields with a minimal template (Min DS) (~10 fields) will show significant improvement in Run Time of the Models**

# Necessary Fields in the Data Structure

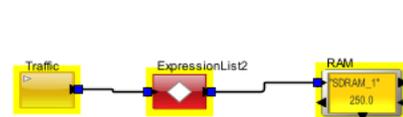
- A\_Source
- A\_Destination
- A\_Command
- A\_Bytes
- A\_Bytes\_Remaining
- A\_Bytes\_Sent
- A\_Task\_Flag

**Note:** Error message will be thrown when one or more required fields are absent





Block : stochastic\_model\_0.RAM  
 Line : Incoming Data Structure does not contain one or more of the necessary fields:  
 {"A\_Source", "A\_Destination"}  
 Error\_Number : DRAM\_001A  
 Explanation : Add the required fields listed.  
 Exception : Incoming Data Structure is missing some fields  
 in .stochastic\_model\_0.RAM



```

Edit parameters for ExpressionList2
Result_C = MyRegExpression_C_or_None /* Exp
/* Add as many RegEx lines are required */

/* Template to enter multiple RegEx lines*/
input.A_Source = "Traffic"
input.A_Destination = "SDRAM_1"
input.A_Bytes = 8
input.A_Bytes_Remaining = 4
input.A_Bytes_Sent = 4
input.A_Command = "Read"
    
```

Adding the necessary fields resolves the issue

# Parameter

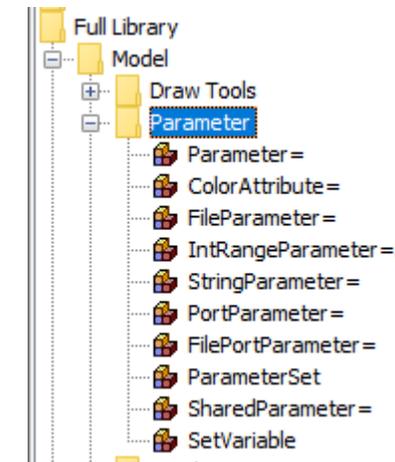
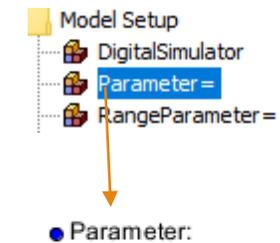
Used to define values which cannot be changed during simulation but can be changed between simulation

## Usage

- Any block in the BDE can access these values
- Export block parameters to link to the BDE parameters

## Types

- Generic parameters - Scalar and String
- Expressions
- File names
- Parameter Set



# Variable

Variable is a lookup list, variable or register to store data during the simulation

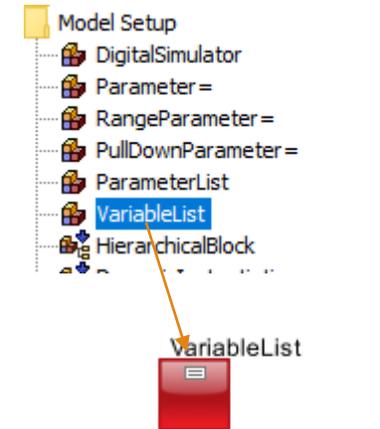
Variable is a named location

- Local- available in the current window only
- Global- available throughout the model
- Block- available within the Script, ExpressionList

Used to communicate between blocks and routing

Defined in VariableList (Global and local), ExpressionList (Block) and Script (Block)

Supports all standard data types



# Data Types supported in VisualSim

## 1. Scalar

- Integer
- Double
- Long

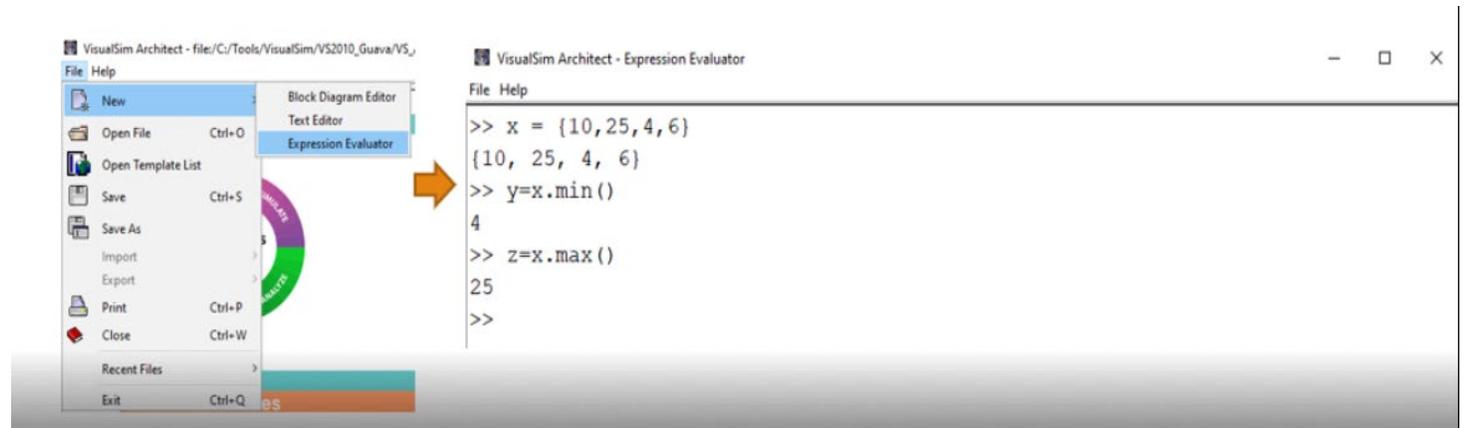
## 2. String

## 3. Array

## 4. Matrix

### Note:

- $1/5$  will be 0
- $1.0/5$  will be 0.2
- $1/5.0$  will be 0.2
- $1.0/5.0$  will be 0.2



# RegEx

- Collection of Mathematical, Logical, Statistical and Algorithm-Specific Functions
- Popular RegEx are used with Array, Queues, Schedulers, Data Structure, Power and Networking
- Usage
  - ✓ Parameters
  - ✓ Processing, ExpressionList, Script and Queue/Server for defining logic and decisions
  - ✓ Can combine parameters, variable and data structure fields

# Regex Examples

## Hardware Architecture Routing table

<b>addToRoutingTable</b> (String Architecture_Name, String Source_Name, String Destination_Name, String Hop_Name, String Source_Port_Name)	Adds a line to the routing table. This is used to add a specific connection.	addToRoutingTable ("Architecture_1", "IO_2","IO_2", "Bus_Name_1", "Bus_Name_2")
<b>getRoutingTable</b> (String Architecture_Name)	Get Routing Table as a Data Structure.	getRoutingTable("Architecture_1")

## Statistics

<b>addStats</b> (String memory_name, double value)	Updates a statistics memory that is defined using the Statistics block. If the memory exists, then add double value and return true, else returns false.
---	--

## Power Regex

<b>powerManager</b> (String power_table_name)	Gets the complete power table. This will get the details for all devices that are associated with this PowerTable	powerManager ("ARM_Table_Manager ")
<b>stateChange</b> (String power_manager_name, String Device Name String Operating State, String State name)	Changes the state of a device dynamically. See device list above.	stateChange("Manager_1", "ARM","Standby","Idle")

## File I/O

<b>readFile</b> (String filename)	Get the string text contained in the specified file.	readFile("File_Name")
<b>writeFile</b> (String filename, java.lang.Object token)	Write the token to the specified file.	writeFile("File_Name",port_token)

# Expression List



Sequence of mathematical expressions

Requires one transaction on all input ports to fire the block

Assign values to fields or variables

- Execution starts when data arrives on all input ports
- Data at each port is identified by the port name
- Queued if multiple values arrive at one port

Usage

- `input.field_name = value`

Output

- Condition can use any expression containing fields, variables and logical operators

## Some Commonly used Expressions in ExpressionList

The screenshot shows the configuration for an ExpressionList block. It includes an 'Expression\_List' field with a code editor containing the following code:

```
/* Template to enter multiple RegEx lines*/  
latency = TNow - input.TIME
```

Below this is another 'Expression\_List' field with the following code:

```
Result_A = (Select_Scenario == 1)?true:false  
input.INDEX = (Result_A == true)?(irand(1,10)):input.INDEX  
input.ID = ID_Counter + 1  
ID_Counter = ID_Counter + 1  
input.TIME = TNow
```

Annotations include a bracket on the left side of the second code block labeled 'Execution is sequential' and a circle around the `input.TIME = TNow` line. On the right side, the text 'Can use conditional statements' is present with an arrow pointing to the `Result_A` assignment line.

At the bottom, the 'Output\_Ports' section is visible with the following settings:

Output_Ports:	output,output1
Output_Values:	input,input
Output_Conditions:	Result_A,Result_A

# Using Regex in ExpressionList – Demo Model

The model uses two server blocks.  
The statistics are generated using  
getBlockStatus from the ExpressionList blocks



# Demo Models – Regex in script block

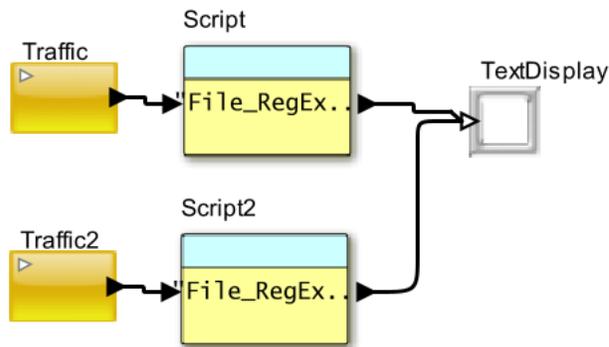
## Regex - File

Parameters:

- Sim\_Time: 0.1

this model takes the file name as the parameter  
Appends the file with the incoming traffic  
and the traffic is written to another file

DigitalSimulator



\$VS/doc/Training\_Material/Tutorial/General/Regex/  
Regex\_File\_Test.xml

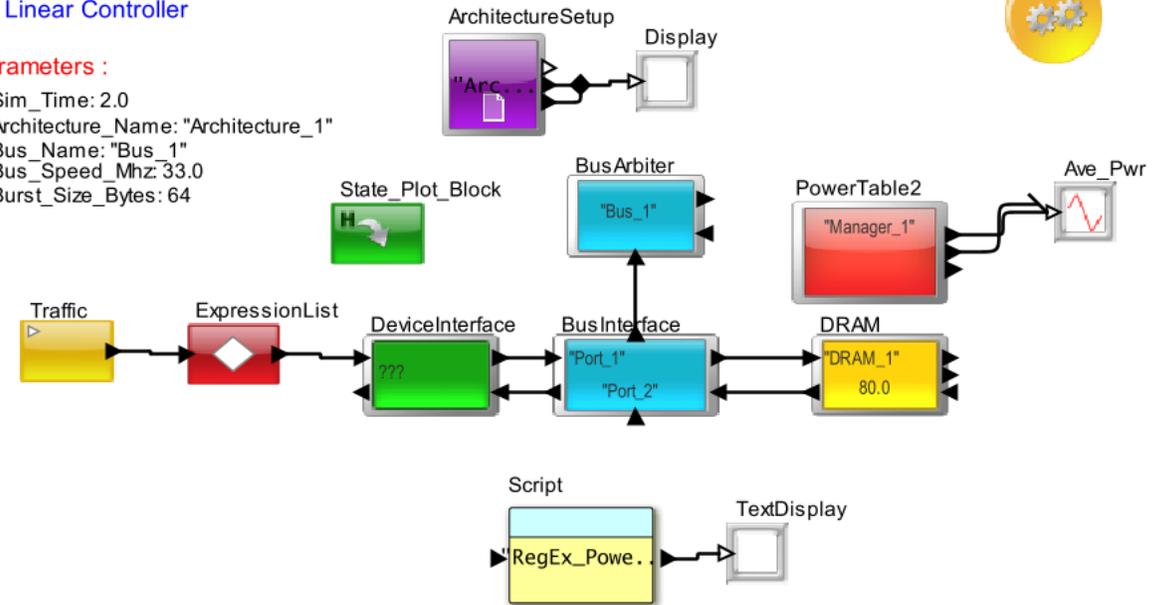
## Linear Controller - Power Calculation

model providing Round Robin Algorithm  
in Linear Controller

Parameters :

- Sim\_Time: 2.0
- Architecture\_Name: "Architecture\_1"
- Bus\_Name: "Bus\_1"
- Bus\_Speed\_Mhz: 33.0
- Burst\_Size\_Bytes: 64

Digital



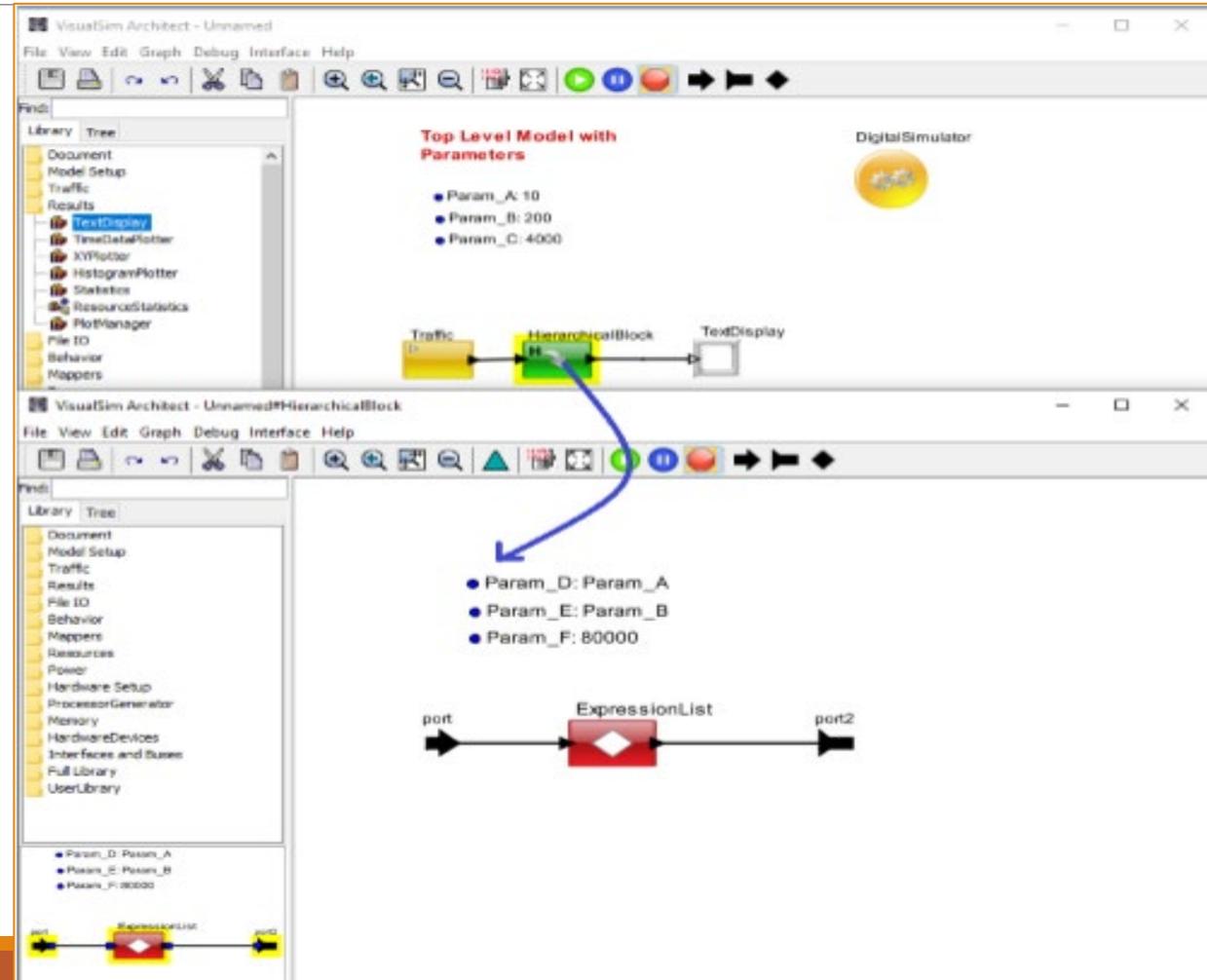
\$VS/doc/Training\_Material/Tutorial/Gener  
al/Regex/Regex\_Power\_Test.xml

# Hierarchical/ Class/ DI Blocks

---

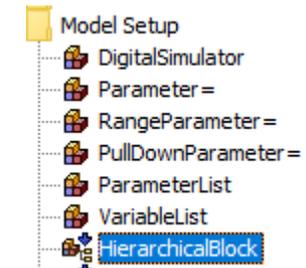
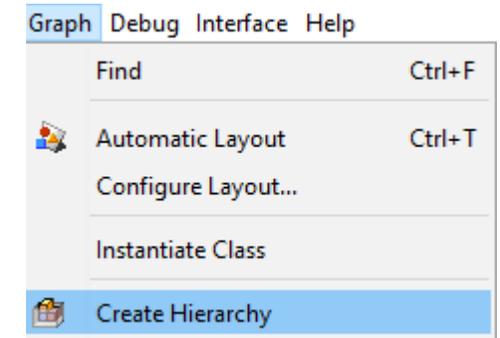
# Hierarchical Blocks

- Grouping a set of functional blocks that combine to define a function or device



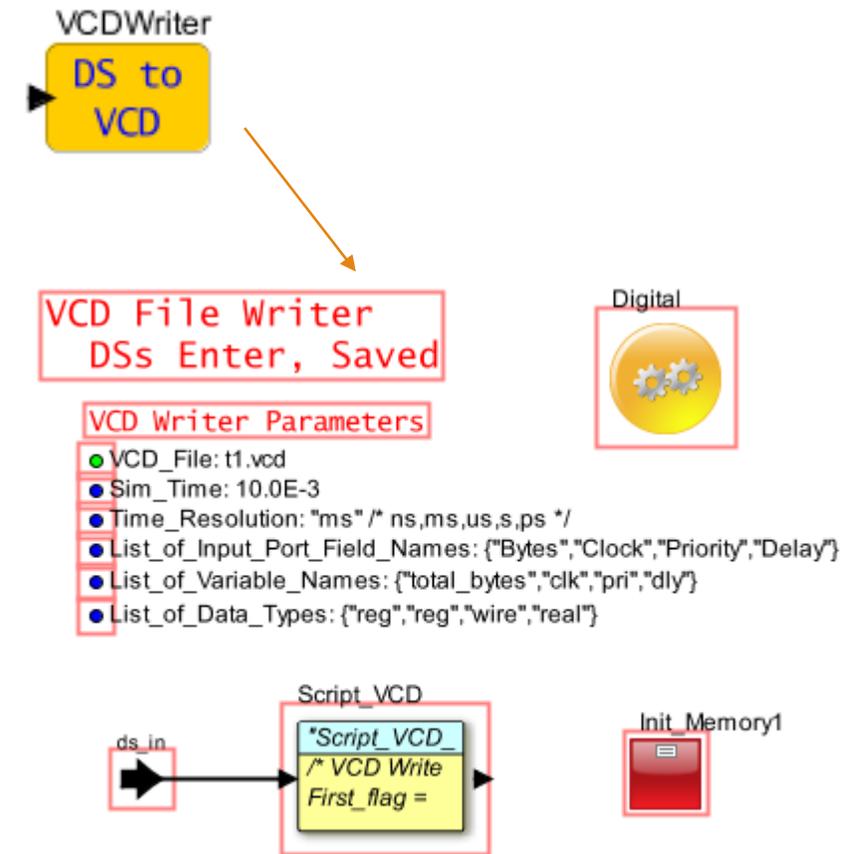
# Construct Hierarchical Block

- Select the blocks to be grouped. Select Edit -> Create Hierarchy
- Drag from Model Setup -> HierarchicalBlock
- Add input and output ports, Parameters
- To view inside the Block, **Right click- > Open Block**



# Class

- A class is a master version of the block.
- Class is an XML sub-model
- Can be instantiated multiple times in the model.
- Changes made to class block are replicated to all linked instances
- All sub-models need a Simulator



# How to construct a Class?

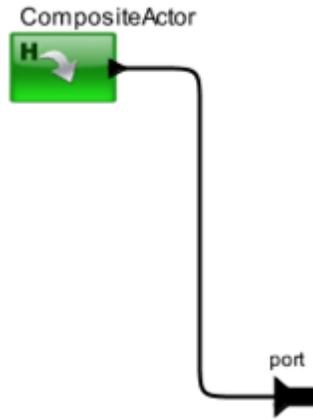
---

- **Assemble the initial block diagram**
  - ✓ Use the library blocks to assemble the model
  - ✓ Create a Hierarchical block of this block diagram
- **Create a class**
  - ✓ Convert the Hierarchical block into a Class
  - ✓ Save as a sub-model
- **Instantiate a new class for use in a model**
  - ✓ To use the Class, you need to instantiate the block in the model using Graph> Instantiate. Entity. Make sure the Class is located within the VS\_Model\_Library directory
- **To test the class**
  - ✓ Test the class by constructing a simple model around it.
- **Save in Library**
  - ✓ Right-click the block and select Save Block in Library.

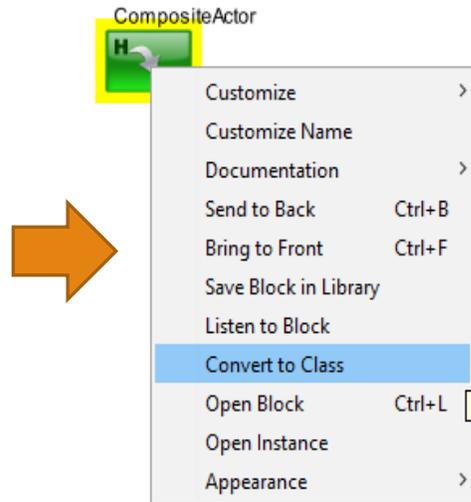


# Constructing a class

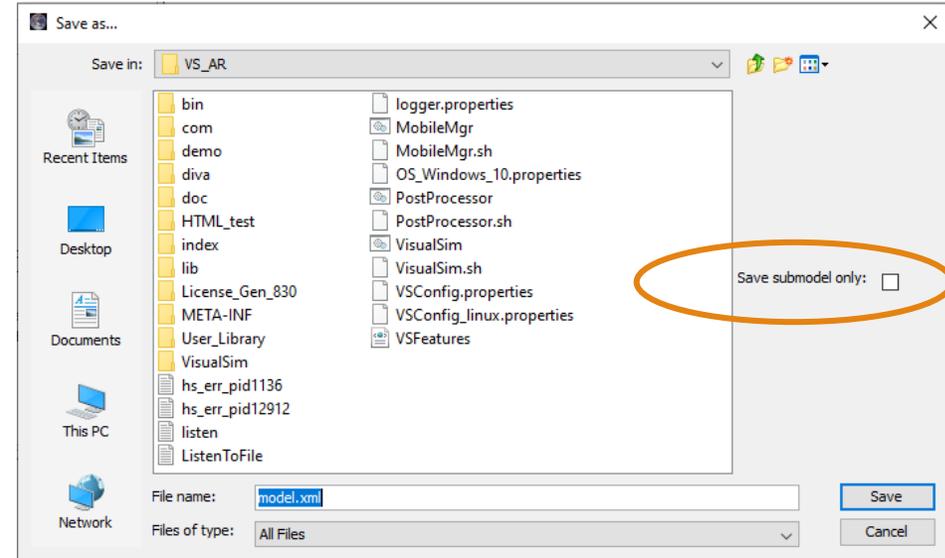
## CREATE HIERARCHY



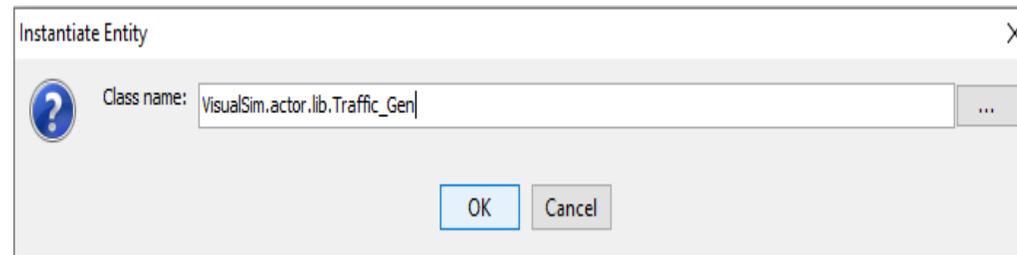
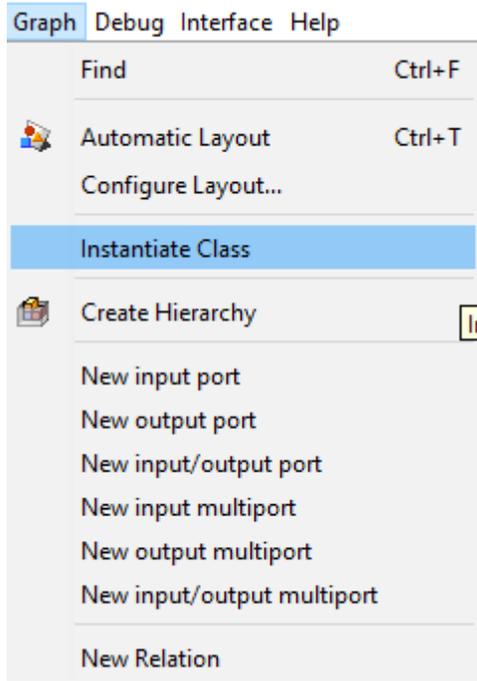
## CONVERT TO CLASS



## OPEN CLASS AND SAVE AS

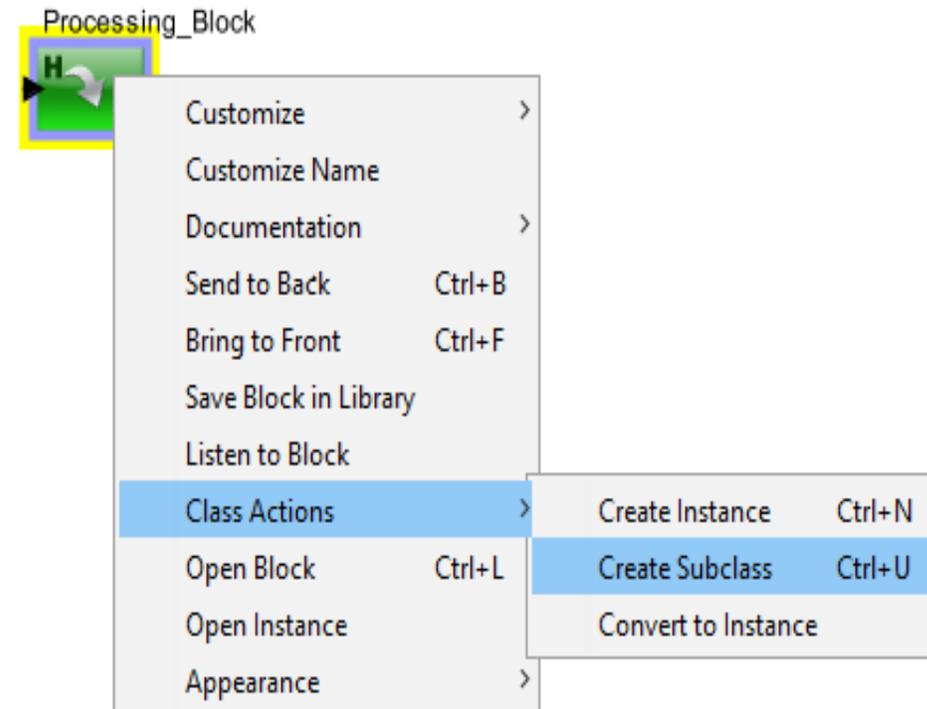


# Instantiating the Class



# Creating Sub Class

- Right click on the Class Block
- Select **Class Actions > Create Subclass**
- Can add additional parameters and Blocks



# Do's and Don't for Class Block

---

Save all class blocks in a common location

Make sure the location is below a Classpath

Organize the Classes in folders under this Classpath

Use the Open Instance for Listen to Block and Listen to Port ONLY

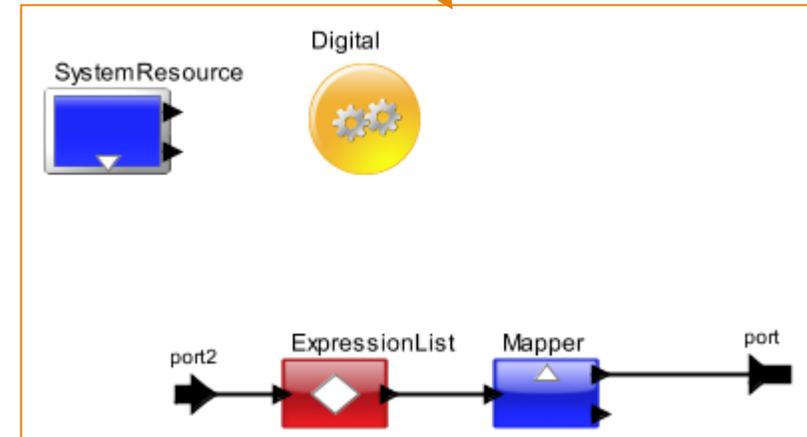
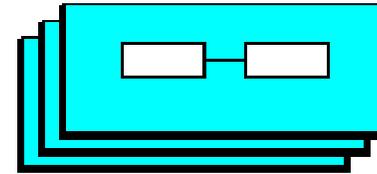
Do not modify any functionality inside the Open Instance.

- This will reflect only in that instance of this Class.
- Any changes to the instance is extremely hard to debug.

# Dynamic Instantiation

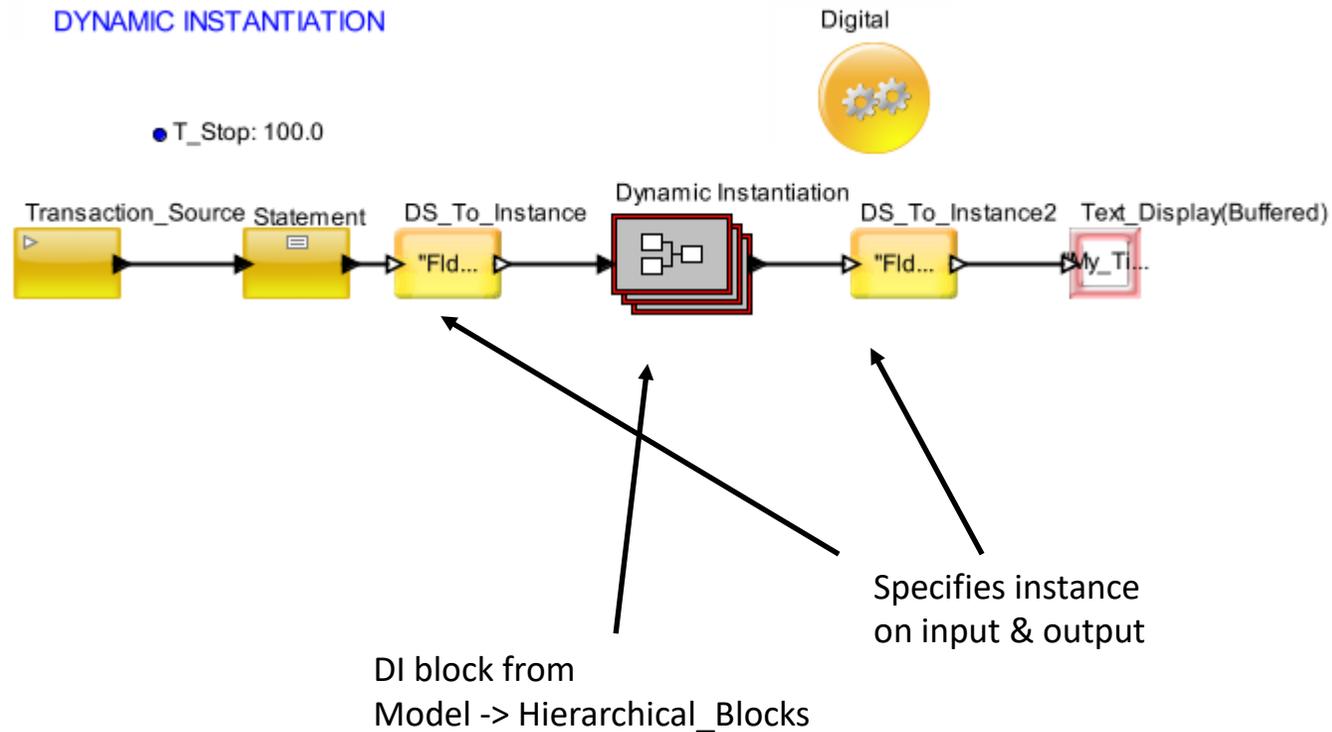
- Creates multiple instances of itself during the preinitialize phase of model execution.
- Each instance of this block behaves exactly like a Hierarchical block.
- Helps significantly in building large designs where the model structure scales.

DynamicInstantiation



# Example of Dynamic Instantiation

## DYNAMIC INSTANTIATION



# Advantages of Dynamic Instantiation

- 
- Dynamic Instances solves modeling problems where objects are arbitrarily required.
  - Examples are:
    - ✓ Mobile units entering, traversing and exiting a coverage area
    - ✓ Creation and deletion of virtual circuits
    - ✓ Peer-to-peer protocol connections
  - Above problems are difficult to model using Static Instantiation
    - ✓ Place many copies of a block in a diagram and use block logic to emulate creation and deletion and to branch data to/from the appropriate blocks.
    - ✓ Create a primitive that does all the work.
  - Static Instantiation requires over-specification and makes model engineering and presentation difficult.

# Differences

---

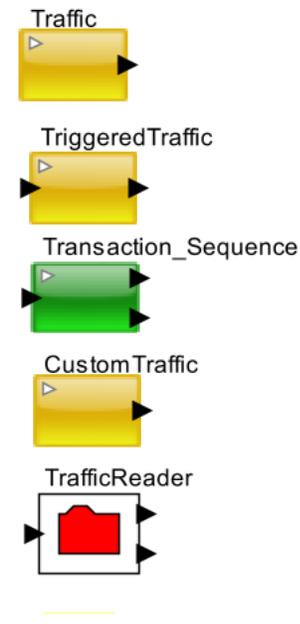
- Modification in one Hierarchical block does not affect another
- Changes in Open Block of Class block will be reflected at all places were used
- Changes in Instance will only be reflected in that one location
- Open Block for Dynamic Instantiation and Hierarchical Block for Listeners
- Open Instance for Classes for Listeners

Traffic

# Data Structure Generation

---

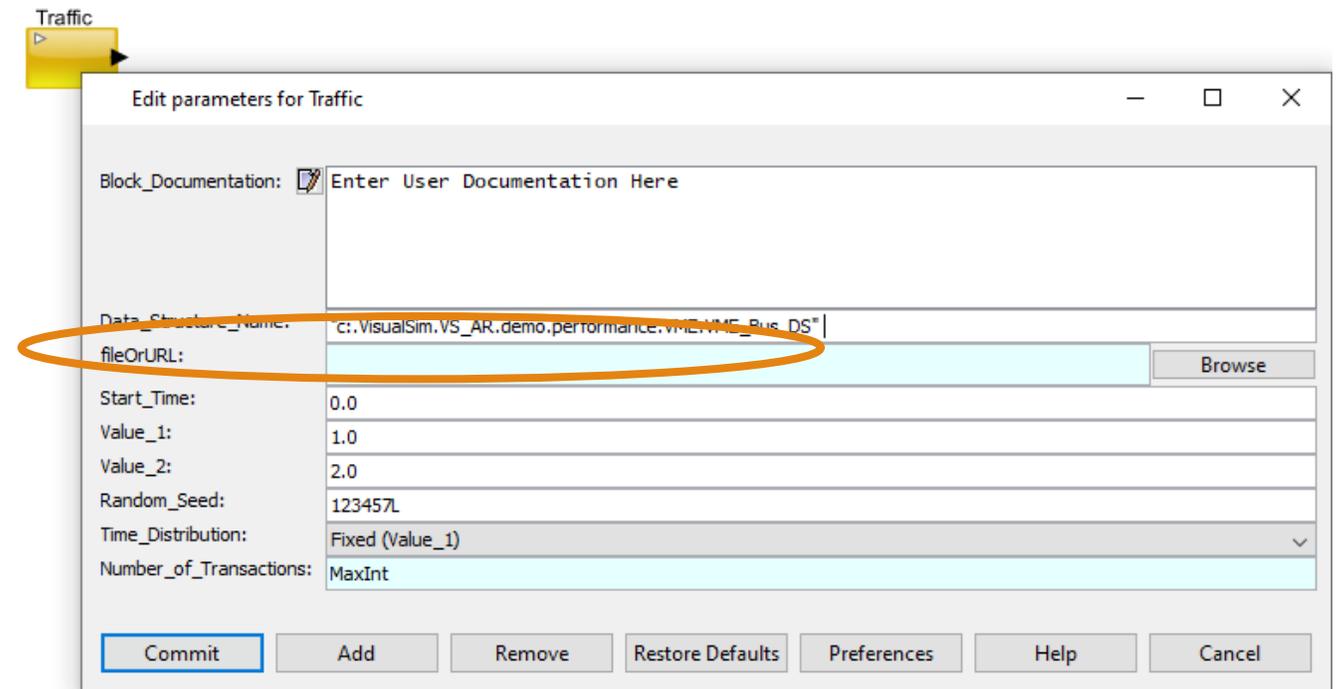
- Traffic
  - ✓ Time distribution
- TriggeredTraffic
  - ✓ Requires an input trigger to generate DS
- Transaction Sequence
  - ✓ Custom list of operations
- Custom Traffic
  - ✓ Periodic distribution
- Trace
  - ✓ Read from a file
- Using RegEX
  - ✓ `newToken(Value)`



# Defining Data Structure in Traffic Blocks

## Data Structures template

- .txt can be located anywhere
- .class located in VS\_AR/VisualSim/data
- Absolute path is required for accessing files located anywhere.
- File name if located in the \$VS/VisualSim/data directory.



# Types

---

- **Statistics Distribution-** Single request, periodic or fixed, uniform within a range, normal, exponential
- **Custom-** Based on a combination of data size and interface speed. Can also be triggered by external event
- **Trace file-** Existing file from hardware bus, network, software thread execution sequence, instruction order
- **Sequence-** Special case to typically debug with a order such as command of “Read, Write, Write, Read , or packet sizes of “128, 1512, 256”

# Traffic



- Double click to configure

Edit parameters for Traffic

Block\_Documentation:  Enter User Documentation Here

Data\_Structure\_Name: \_\_\_\_\_

fileOrURL: C:/VisualSim/Data\_Structure\_Name.txt

Start\_Time: 0.0

Value\_1: 1.0

Value\_2: 2.0

Random\_Seed: 123457L

Time\_Distribution: Single Event

Number\_of\_Transactions: MaxInt

< \_\_\_\_\_ >

This Parameter is an alternate to the Data\_Structure\_Name field above. If the user defines a file name here, the above parameter is not considered.

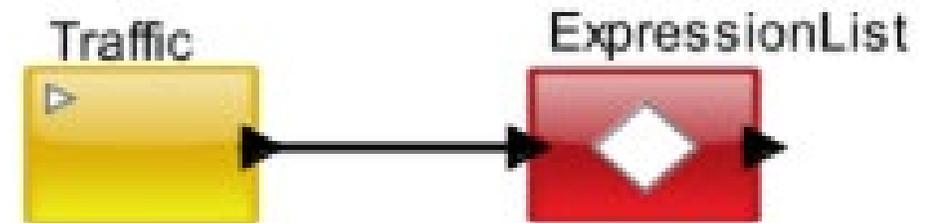
Select the "Time\_Distribution" according to design

Restrict the number of transactions

# Type I - Statistical

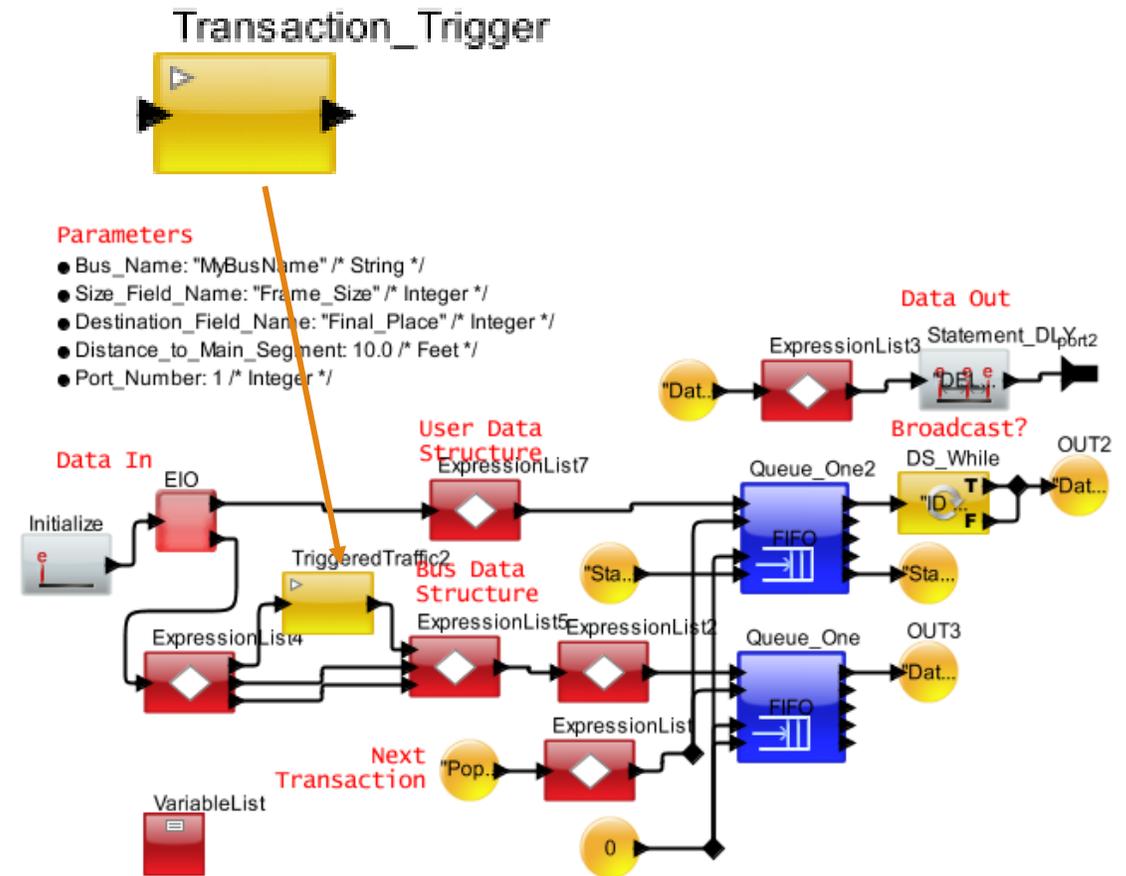
---

- Define a distribution
- Parameters for mean and standard distribution
- Specify values for the Data Structure fields. It can be source, destination, data, priority or bus delay



# Type II - Custom

- If a custom distribution is required or the Data Structure is generated as a function of another activity, or triggered during the flow, use the **Triggered Traffic**.
- Every time the input port is triggered the Triggered Traffic block generates a transaction



# Type III – Transaction Sequence

- Generate transactions or Data Structures in a specific sequence
- Define sequences in the parameter window or specify a file + path
- Time interval between Data Structure is a parameter
- Specify an output processing using the Regular Expression (RegEx) Language

## Traffic Modes.

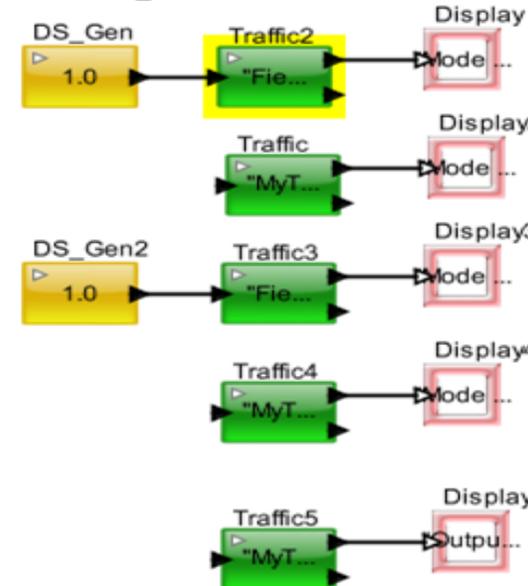
This Model uses 'Data\_Structure\_Text' for traffic.  
 Time is in 'MyTime' field, Probability is in 'MyProb' field.  
 All modes are sending out the full Data Structure, so output=traffic".



```

    • Traffic: "/"
      ID MyStr MyTime MyProb ;
      0 Str_1 1.0 0.15 ; /* DS 1 */
      1 Str_2 2.0 0.25 ; /* DS 2 */
      2 Str_3 3.0 0.17 ; /* DS 3 */
      3 Str_4 4.0 0.23 ; /* DS 4 */
      4 Str_5 5.0 0.2 ; /* DS 5 */
    
```

**Traffic Text Window is the same in each.**



**Trigger Only.**

Each Trigger gets next Data Structure.

**Time Only.**

Time Column gets next Data Structure.

**Trigger Probability (5 Outputs)**

Each Trigger gets a Probability Column Data Structure.

**Time + Probability (2 Outputs)**

Probability Column selects Time to delay Data Structure, last entry ends sequence, so only 2 outputs.

**Output Field Expression Processing**  
 Expression performed on MyTime and the result is placed on the output port

# Type IV – Custom Traffic

- Generate data structure during the T\_Interval period
- Stalls all transmission during the T\_Pause.
- Equally distributes the Number\_Of\_Transactions during the T\_Interval range.



Edit parameters for CustomTraffic

Block\_Documentatio...  Enter User Documentati 

Data\_Structure\_Name: "Header"

fileOrURL:

Start\_Time: 0.0

Time\_Interval:

Time\_Pause:

Number\_of\_Transactions: MaxInt

<  >

Commit Add Remove

# Plotting, Displays and Statistics

# Result

---

- **Statistics**

- ✓ ResourceStatistics
- ✓ Statistics blocks to collect statistics at intermediate points

- **Assertions or tests**

- ✓ High/low value for scalar
- ✓ Conditional model activity
- ✓ Model termination

- **Collect data**

- ✓ Write to screen or to files (Excel, text or XML)

- **Plot data**

- ✓ Bar, Histogram or XY plots
- ✓ Special viewers- Matrix, Image, MPEG and speakers

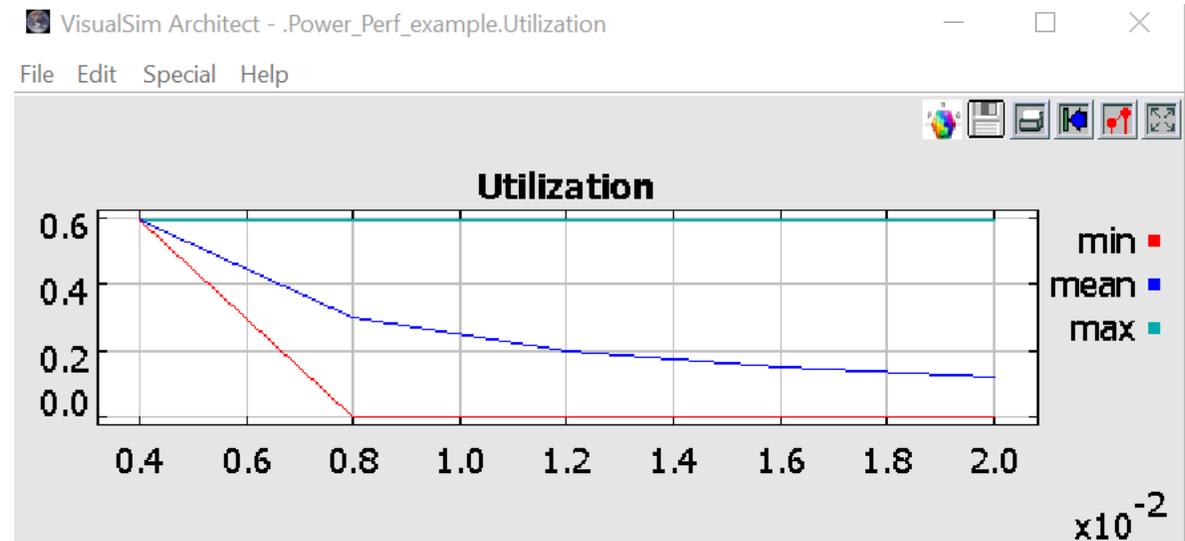
- **3D- Interactive Creation**

- ✓ Create custom animated views that resemble the system

# Using TimeData Plotter



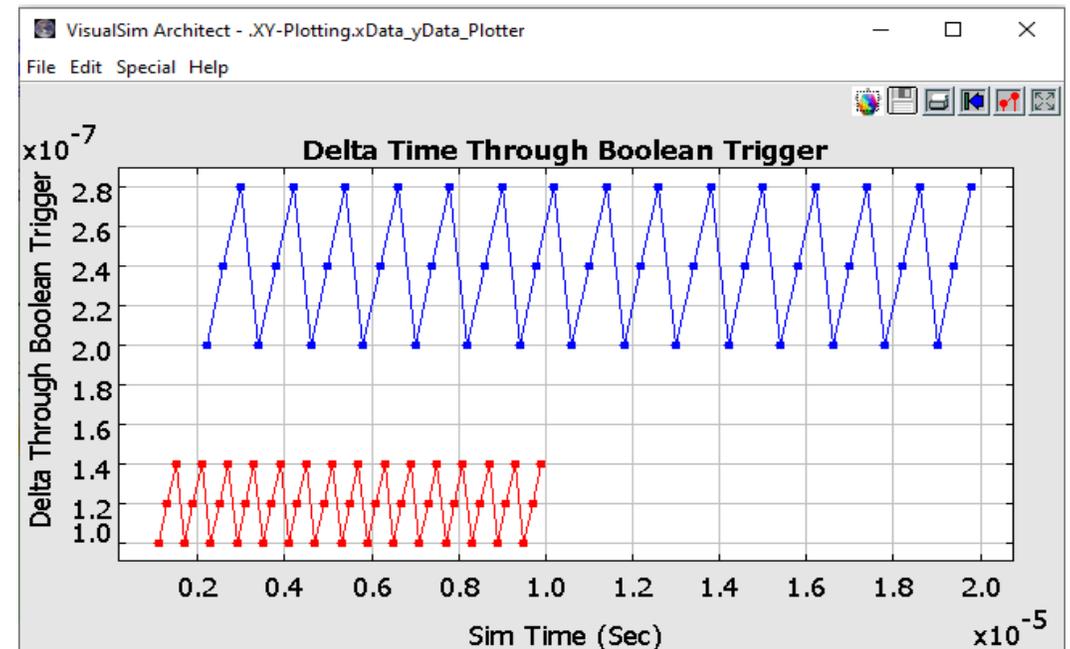
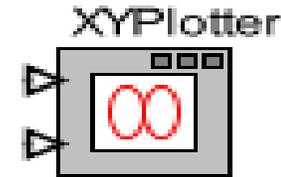
- Plot double values against simulation time
- View or save the results of the simulation in a XY format.
- Used to depict latency, throughput and other variables that vary against time.



X-axis : Sim Time (s), Y-axis : Utilization

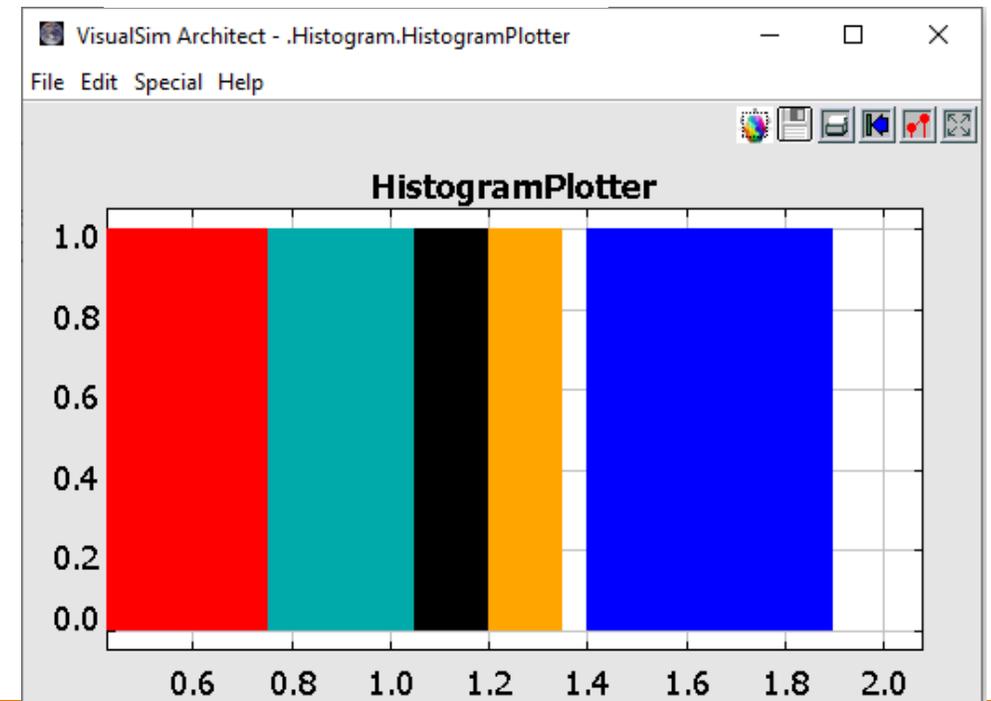
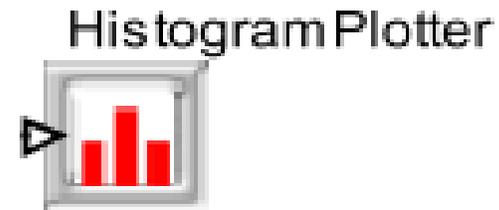
# XY Plotter

- Any scalar value against any other scalar value. Both values must arrive synchronously.
- The X- and Y-axis can have different data values.
- Plots can be Latency vs. Packet Size or Task Delay vs. Processor Speed.
- The parameters of this block match the fields (or RegEx) of the incoming Data Structure to determine the coordinates, color and trace identifier (Dataset).
- Values, color, legend defined in fields of incoming data structure. Plot similar to XYPlotter

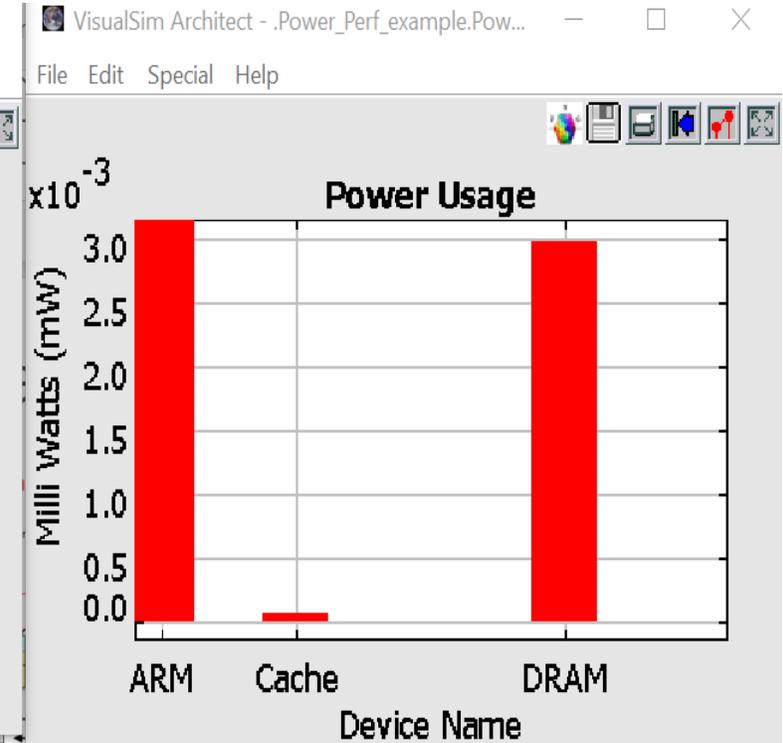
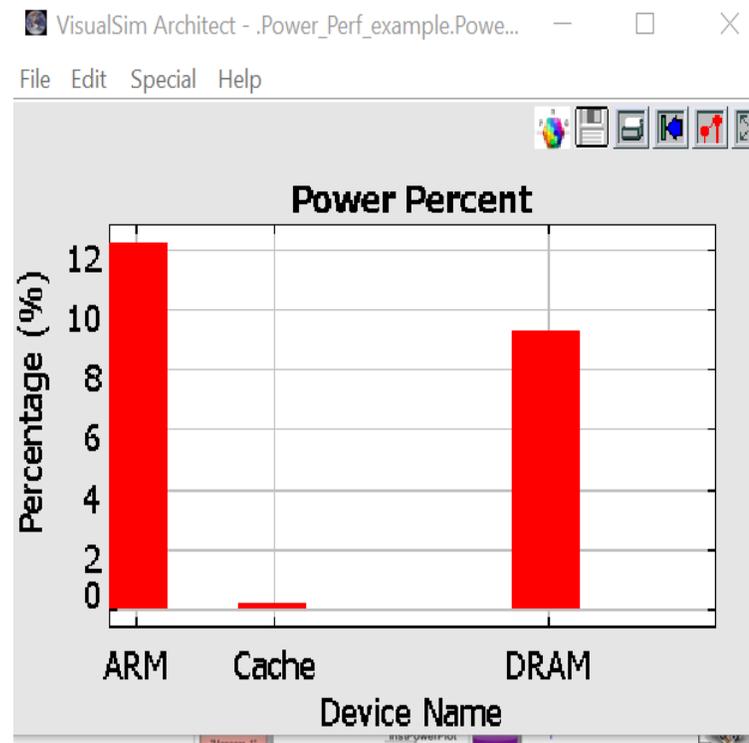
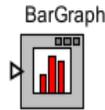
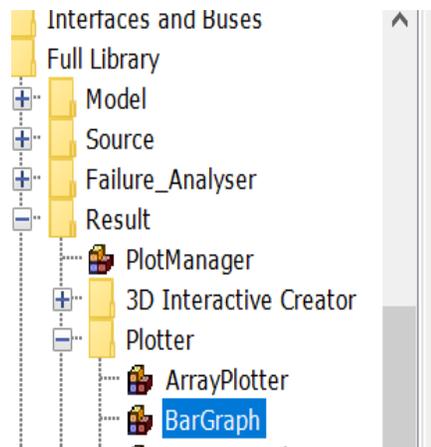


# Histogram

- The plotter accepts data on the input and plots them as a histogram.
- View the plot in real-time or save for future viewing.

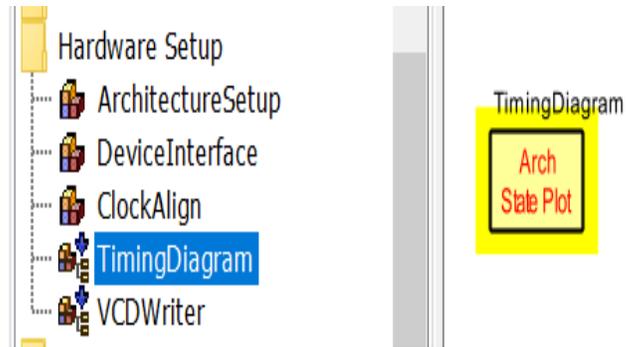


# Using Bar Graph

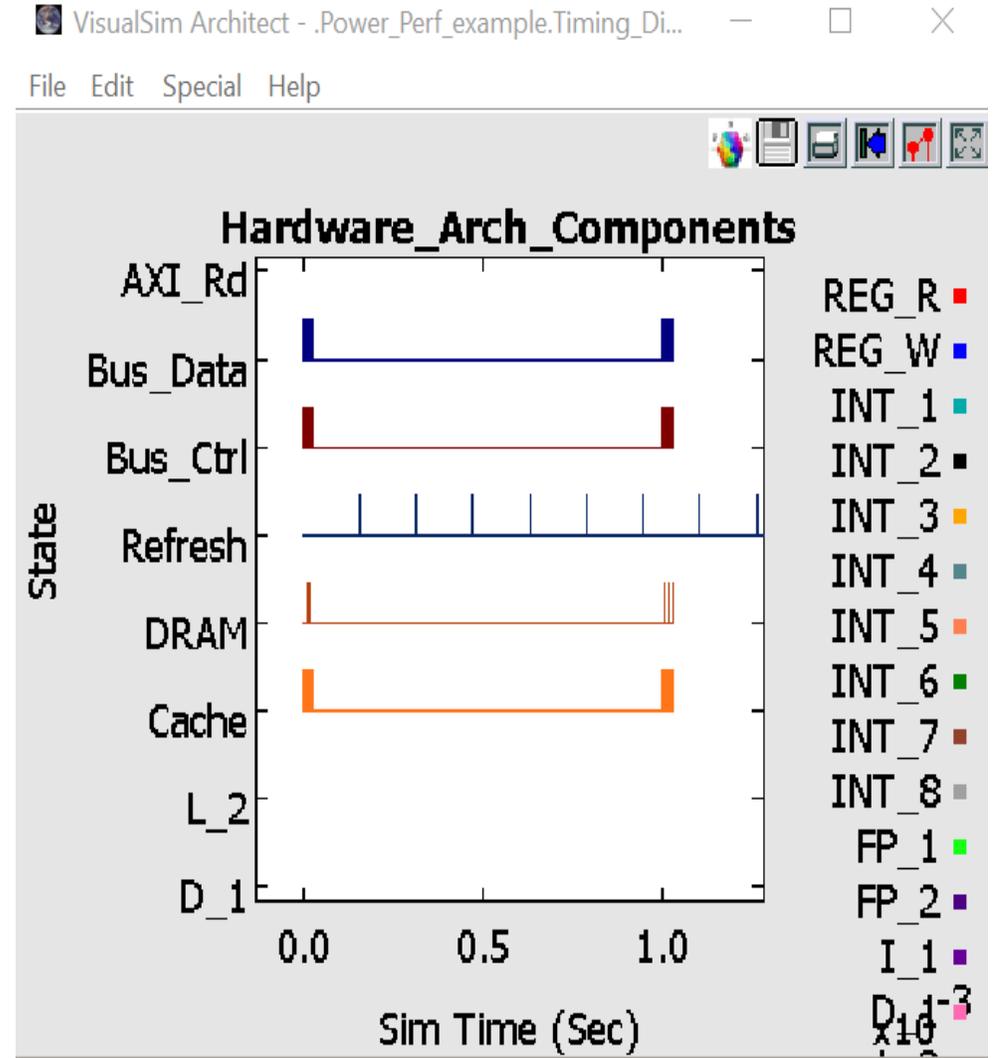


- The bar graph plots series
- The input is an array

# Using Timing Diagram

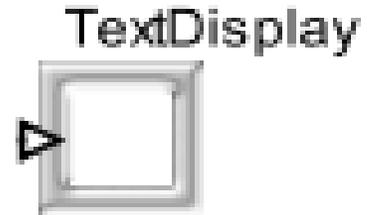


Need to add details



# Text Display

- Output to text the data structure and statistics
- The input type can be of any type.
- Can be set to Save/View from Post Processor
- Cannot be viewed from the Post Processor



```

VisualSim Architect - .Text_Display_Unbuffered.Text_Display(...)
----- 0.00 ns -----
{BLOCK                               = "Transaction_Source",
DELTA                               = 0.0,
DS_NAME                             = "Header_Only",
ID                                   = 1,
INDEX                                = 0,
|TIME                                = 0.0}

```

Edit parameters for TextDisplay

Block\_Documentation:

ViewText:

saveText:

Append\_Time:

fileName:

rowsDisplayed:

columnsDisplayed:

suppressBlankLines:

title:

<        >

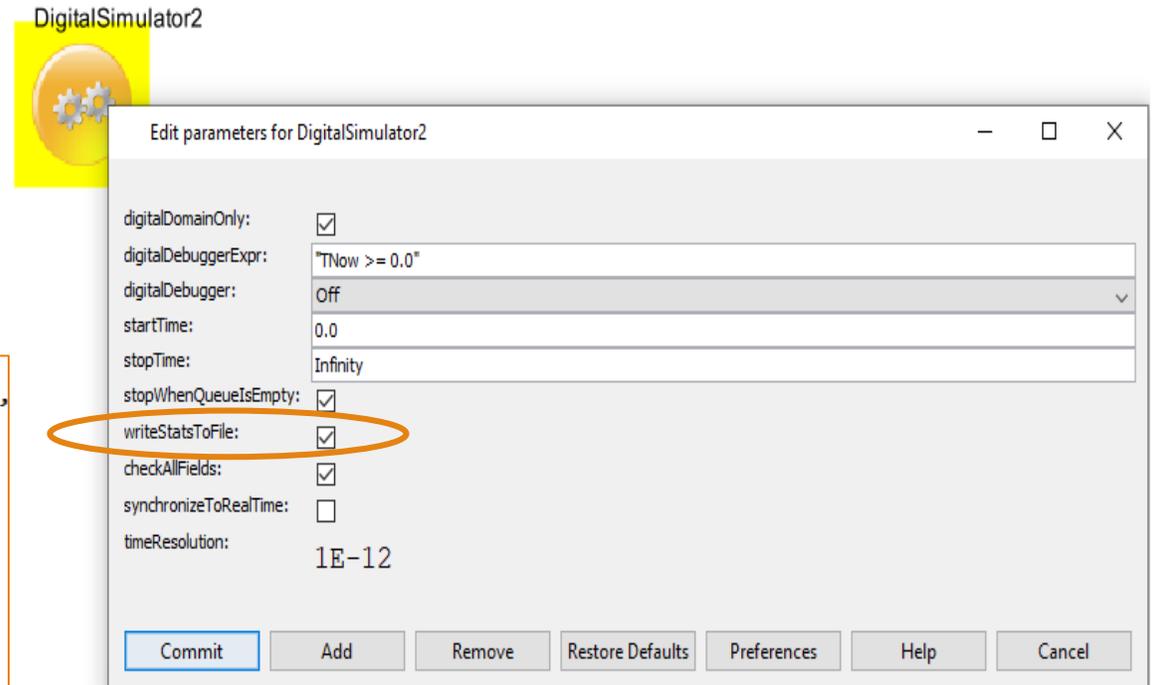


# writeStats To File

- Generates Statistics for all the blocks in the model at the end of simulation
- Writes into a Text File in the model directory

```

Queue_Statistics      6.000000000000 sec
{BLOCK                = "SR_SrExtend_example.SystemResource_Extend",
DELTA                 = 0.0,
DS_NAME               = "Queue_Common_Stats",
ID                    = 1,
INDEX                 = 0,
Number_Entered        = 7,
Number_Exited         = 1,
Number_Rejected       = 0,
Occupancy_Max         = 6.0,
Occupancy_Mean        = 3.77777777777778,
Occupancy_Min         = 1.0,
Occupancy_StDev       = 1.4740554623802,
Queue_Number         = 1,
TIME                  = 6.0,
Total_Delay_Max       = 4.0,
Total_Delay_Mean      = 4.0,
Total_Delay_Min       = 4.0,
Total_Delay_StDev     = 0.0,
Utilization Mean      = 0.0}
    
```



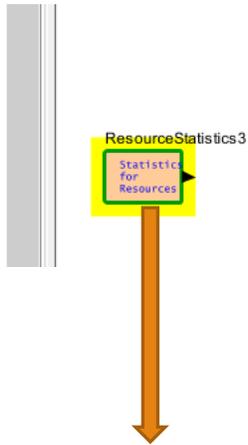
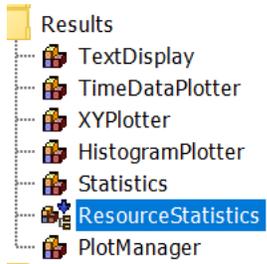
---

# Differences between ArchitectureSetup and Resource Statistics Blocks

# Resource Statistics from Architecture setup – Covers all the hardware Blocks

DISPLAY AT TIME	----- 4.00000000 ms -----	BLOCK	= ".Power_Perf_example."	RAM_Bus_Delay_Mean	= 4.7999999999962E-8,
{AHB_Bus_Delay_Max	= 7.4000000000081E-8,	Cache_Delay_Time_Max	= 4.8192771084337E-8,	RAM_Bus_Delay_Min	= 4.7999999999888E-8,
AHB_Bus_Delay_Mean	= 4.1460937499999E-8,	Cache_Delay_Time_Mean	= 4.6380835843373E-8,	RAM_Bus_Delay_StDev	= 0.0,
AHB_Bus_Delay_Min	= 3.999999999979E-8,	Cache_Delay_Time_Min	= 6.0240963855422E-9,	RAM_Bus_IOS_per_sec_Max	= 3000.0,
AHB_Bus_Delay_StDev	= 6.8947470309723E-9,	Cache_Delay_Time_StDev	= 8.551245431752E-9,	RAM_Bus_IOS_per_sec_Mean	= 3000.0,
AHB_Bus_IOS_per_sec_Max	= 256000.0,	Cache_Hit_Ratio_Max	= 95.5102040816327,	RAM_Bus_IOS_per_sec_Min	= 3000.0,
AHB_Bus_IOS_per_sec_Mean	= 256000.0,	Cache_Hit_Ratio_Mean	= 95.5102040816327,	RAM_Bus_IOS_per_sec_StDev	= 0.0,
AHB_Bus_IOS_per_sec_Min	= 256000.0,	Cache_Hit_Ratio_Min	= 95.5102040816327,	RAM_Bus_Input_Buffer_Occupancy_in_Words_Max	= 4.0,
AHB_Bus_IOS_per_sec_StDev	= 0.0,	Cache_Hit_Ratio_StDev	= 0.0,	RAM_Bus_Input_Buffer_Occupancy_in_Words_Mean	= 1.6216216216216,
AHB_Bus_Input_Buffer_Occupancy_in_Words_Max	= 4.0,	Cache_Memory_Used_By_ARM_MB_Max	= 0.01568,	RAM_Bus_Input_Buffer_Occupancy_in_Words_Min	= 0.0,
AHB_Bus_Input_Buffer_Occupancy_in_Words_Mean	= 0.2324853228963,	Cache_Memory_Used_By_ARM_MB_Mean	= 0.01568,	RAM_Bus_Input_Buffer_Occupancy_in_Words_StDev	= 1.4950612633448,
AHB_Bus_Input_Buffer_Occupancy_in_Words_Min	= 0.0,	Cache_Memory_Used_By_ARM_MB_Min	= 0.01568,	RAM_Bus_Preempt_Buffer_Occupancy_in_Words_Max	= 0.0,
AHB_Bus_Input_Buffer_Occupancy_in_Words_StDev	= 0.5308132512572,	Cache_Memory_Used_By_ARM_MB_StDev	= 0.0,	RAM_Bus_Preempt_Buffer_Occupancy_in_Words_Mean	= 0.0,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_Max	= 0.0,	Cache_Memory_Used_By_RAM_MB_Max	= 7.04E-4,	RAM_Bus_Preempt_Buffer_Occupancy_in_Words_Min	= 0.0,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_Mean	= 0.0,	Cache_Memory_Used_By_RAM_MB_Mean	= 7.04E-4,	RAM_Bus_Preempt_Buffer_Occupancy_in_Words_StDev	= 0.0,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_Min	= 0.0,	Cache_Memory_Used_By_RAM_MB_Min	= 7.04E-4,	RAM_Bus_Throughput_MBs_Max	= 0.048,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_StDev	= 0.0,	Cache_Memory_Used_By_RAM_MB_StDev	= 0.0,	RAM_Bus_Throughput_MBs_Mean	= 0.048,
AHB_Bus_Throughput_MBs_Max	= 3.964,	Cache_Memory_Used_By_Total_MB_Max	= 0.016384,	RAM_Bus_Throughput_MBs_Min	= 0.048,
AHB_Bus_Throughput_MBs_Mean	= 3.964,	Cache_Memory_Used_By_Total_MB_Mean	= 0.016384,	RAM_Bus_Throughput_MBs_StDev	= 0.0,
AHB_Bus_Throughput_MBs_Min	= 3.964,	Cache_Memory_Used_By_Total_MB_Min	= 0.016384,	RAM_Delay_Time_Max	= 3.8461538461538E-8,
AHB_Bus_Throughput_MBs_StDev	= 0.0,	Cache_Memory_Used_By_Total_MB_StDev	= 0.0,	RAM_Delay_Time_Mean	= 2.8846153846154E-8,
ARM_Context_Switch_Time_Pct_Max	= 0.075375,	Cache_Throughput_MBs_Max	= 4.096,	RAM_Delay_Time_Min	= 1.9230769230769E-8,
ARM_Context_Switch_Time_Pct_Mean	= 0.075375,	Cache_Throughput_MBs_Mean	= 4.096,	RAM_Delay_Time_StDev	= 9.6153846153846E-9,
ARM_Context_Switch_Time_Pct_Min	= 0.075375,	Cache_Throughput_MBs_Min	= 4.096,	RAM_Memory_Used_By_Cache_MB_Max	= 1.92E-4,
ARM_Context_Switch_Time_Pct_StDev	= 0.0,	Cache_Throughput_MBs_StDev	= 0.0,	RAM_Memory_Used_By_Cache_MB_Mean	= 1.92E-4,
ARM_D_1_Hit_Ratio_Max	= 0.0,	DELTA	= 0.0,	RAM_Memory_Used_By_Cache_MB_Min	= 1.92E-4,
ARM_D_1_Hit_Ratio_Mean	= 0.0,			RAM_Memory_Used_By_Cache_MB_StDev	= 0.0,
ARM_D_1_Hit_Ratio_Min	= 0.0,			RAM_Memory_Used_By_Total_MB_Max	= 1.92E-4,
ARM_D_1_Hit_Ratio_StDev	= 0.0,			RAM_Memory_Used_By_Total_MB_Mean	= 1.92E-4,
ARM_I_1_Hit_Ratio_Max	= 100.0,			RAM_Memory_Used_By_Total_MB_Min	= 1.92E-4,
ARM_I_1_Hit_Ratio_Mean	= 24.9388753056235,			RAM_Memory_Used_By_Total_MB_StDev	= 0.0,
ARM_I_1_Hit_Ratio_Min	= 0.0,			RAM_Throughput_MBs_Max	= 0.048,
ARM_I_1_Hit_Ratio_StDev	= 37.8307159101048,			RAM_Throughput_MBs_Mean	= 0.048,
ARM_Stall_Time_Pct_Max	= 0.0,			RAM_Throughput_MBs_Min	= 0.048,
ARM_Stall_Time_Pct_Mean	= 0.0,			RAM_Throughput_MBs_StDev	= 0.0,
ARM_Stall_Time_Pct_Min	= 0.0,			ROM_Bus_Delay_Max	= 4.800000000105E-8,
				ROM_Bus_Delay_Mean	= 4.7999999999888E-8,
				ROM_Bus_Delay_Min	= 4.7999999999888E-8,

# Using Resource Statistics – Generates Stats for Stochastic Blocks like queues, servers, system resources



Edit parameters for ResourceStatistics

Resource_List:	{ "Queue", "Server", "SR", "SR_E", "QSR" } /* Names of Resources */
ResourceLength_List:	{ Num_Queues, 1, 1, 1, Num_Queues } /* Length of all Resources in the Resource_List */
Number_of_Samples:	2 /* Number of output or reset in a simulation run */
Statistics:	true /* True to generate; False to reset */
SimTime:	EndTime - 1.0 /* End time of the simulation */

Commit Add Remove Restore Defaults Preferences Help Cancel

## Generating Statistics from Resources

Notes:

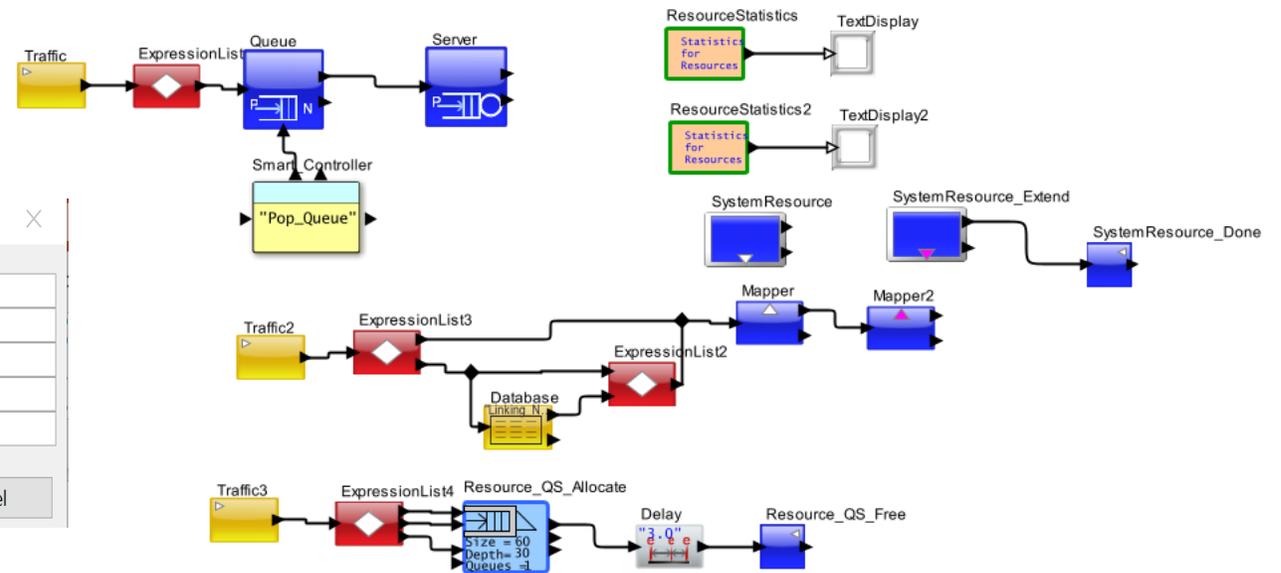
1. Each Resource can be triggered on the stats\_in block to output the current statistics on the stats\_out port.
2. Combine all the statistics request for all Resources in a single location.
3. If there is a single Queue or FIFO for a Resource block, send 1 on the stats\_in to trigger the statistics for the single queue.
4. If there are multiple queues, then the queue is provided to get statistics for a single queue. (Number of Queues) + 1 can be sent to generate statistics for all the queues.

## 2. Parameters

- EndTime: 100.0
- Zone\_Change\_Prob: 0.5
- Num\_of\_Zones: 15
- StartTime: 1.0
- Num\_Queues: 5

## 1. Digital Simulator

DigitalSimulator



## Model Location:

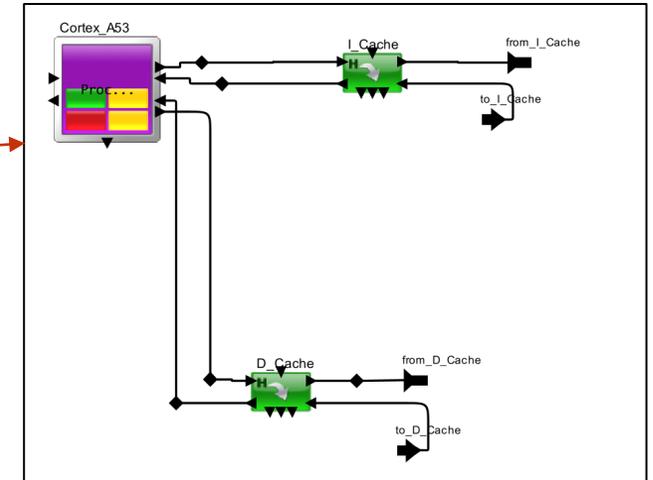
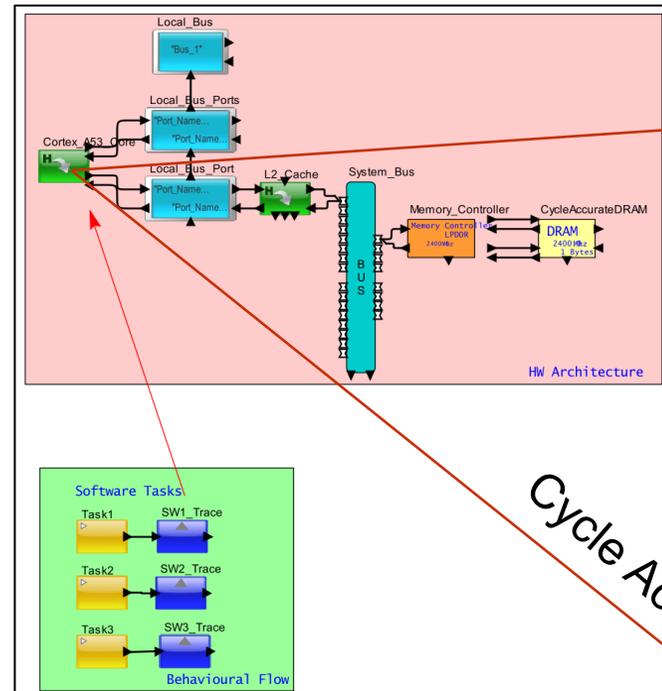
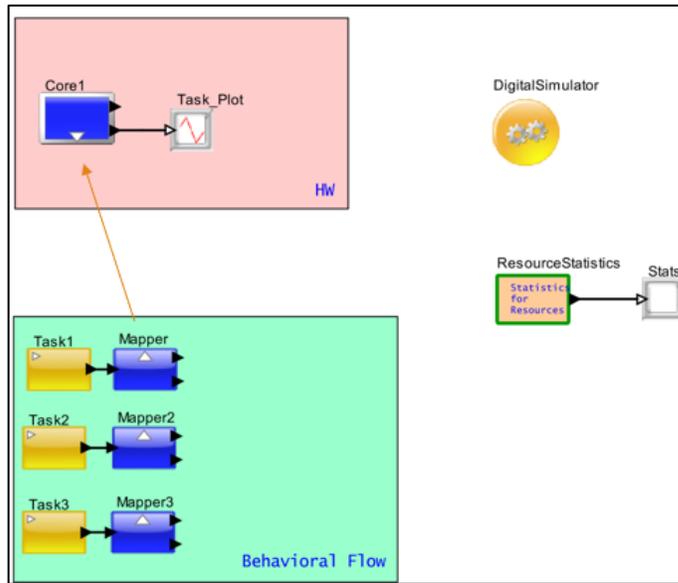
\$VS/doc/Training\_Material/Tutorial/General/Statistics\_Plotting/Resource-Statistics.xml



# Abstraction

# Multiple Levels of Abstraction

## Stochastic vs Hybrid vs Cycle Accurate



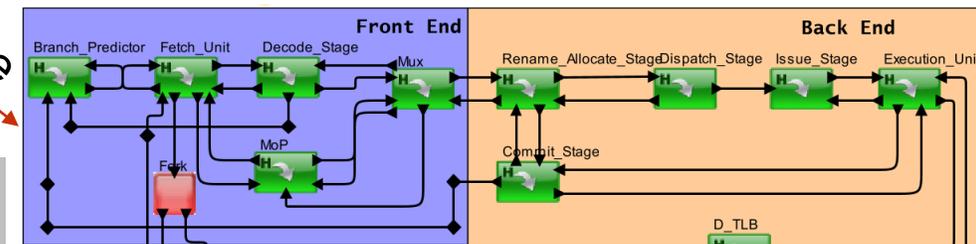
Hybrid

Cycle Accurate

### Stochastic

- Easy to build the target SoC/System (Starting version)
- Stochastic subsystems can be replaced with more accurate blocks as required
- Get an early estimate on Power and Performance metrics

Hybrid – More detailed than stochastic, focus on the application than on the processor micro architecture  
 Cycle Accurate – Very detailed implementation, focus on processor micro architecture



---

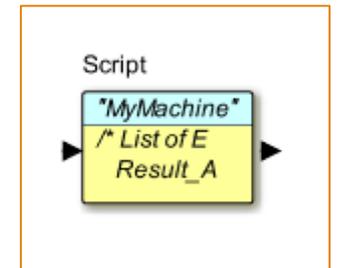
# Algorithms - Adding Custom Block

# Script

---

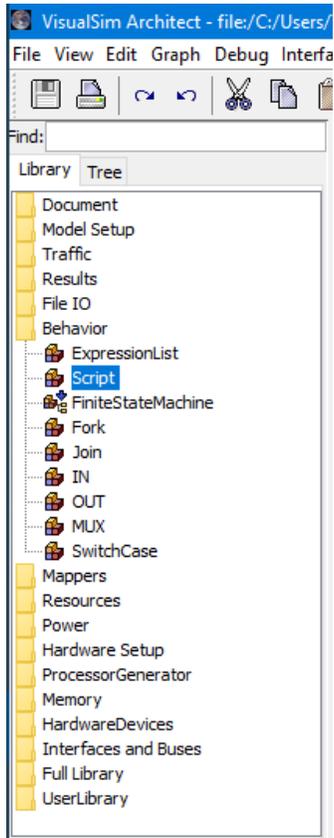
- Used for :
  - Custom logic
  - Scheduling algorithm
  - Arbitration algorithm
  - Statistics and plot generation

A VERY POWERFUL MODULE



- Can be used to define Resources, Cycle-accurate hardware components and algorithmic behaviors
- Can use if, else-if, else, while, for, SWITCH/CASE/BREAK
- Can implement dependencies, Task Graphs etc.

# Getting Started with Script

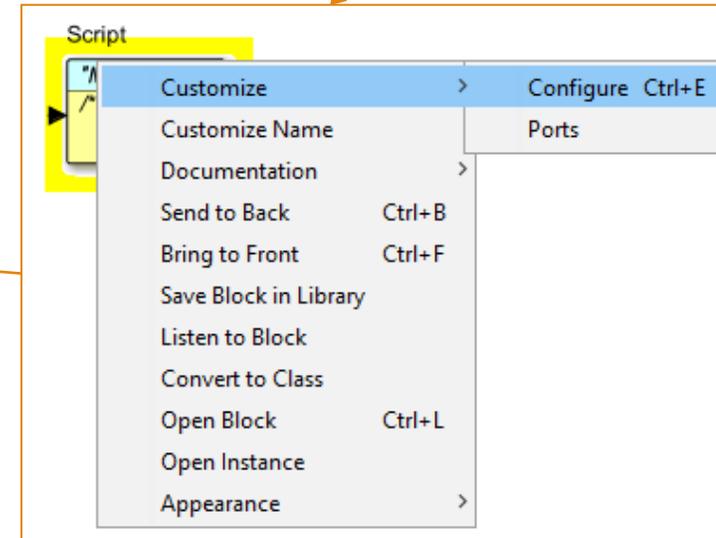
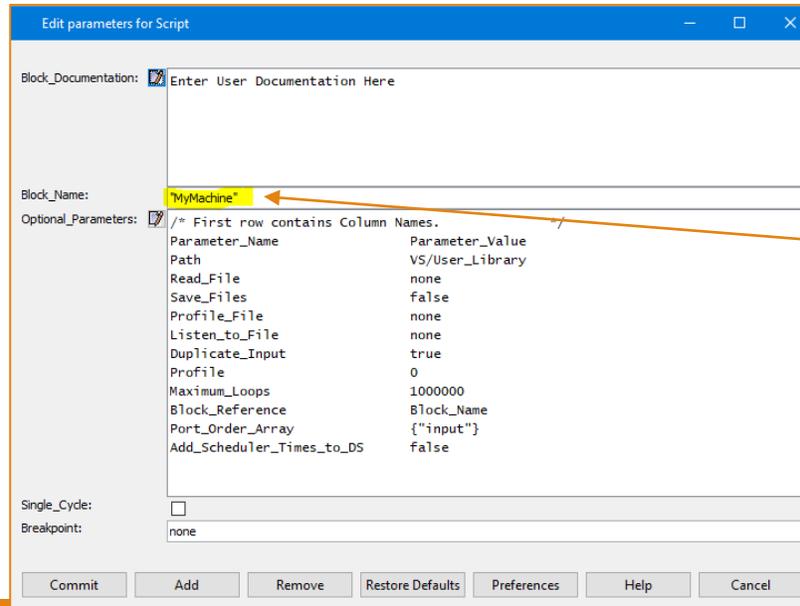


1

Drag and drop the Script block from the library

2

Right Click on script -> Customize -> configure ; give a unique Block Name



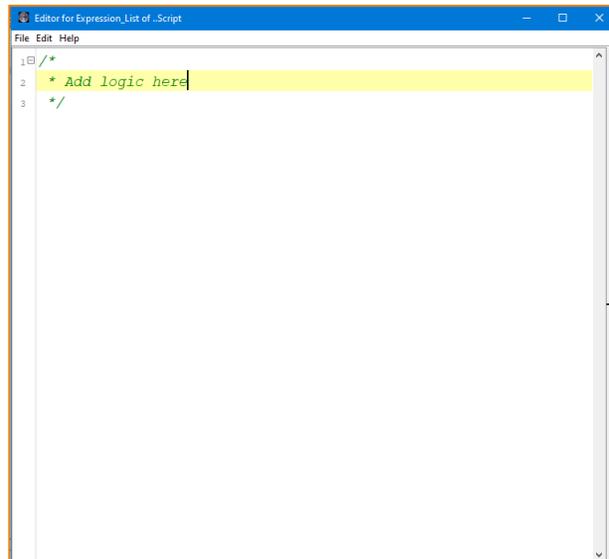
# Getting Started with Script

3

Double click on Script; It already has default contents. Delete them

4

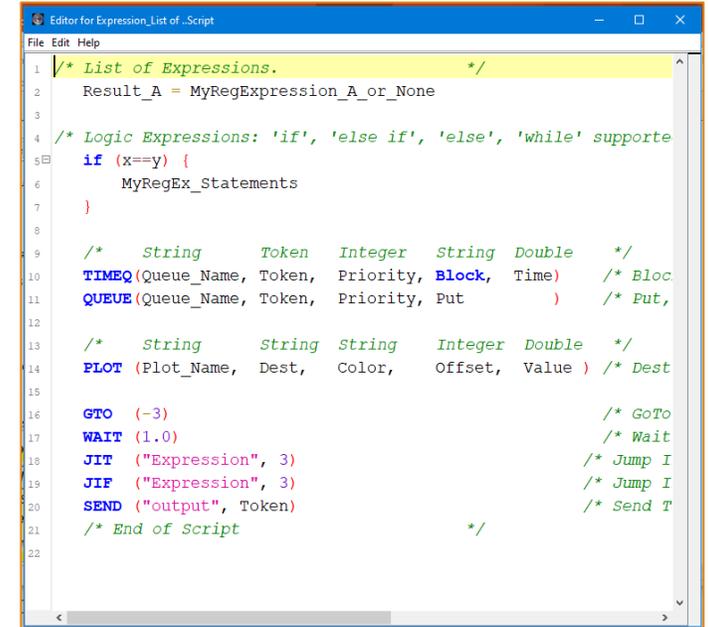
Now we can add our logic here.



```
1 /*  
2  * Add logic here  
3 */
```

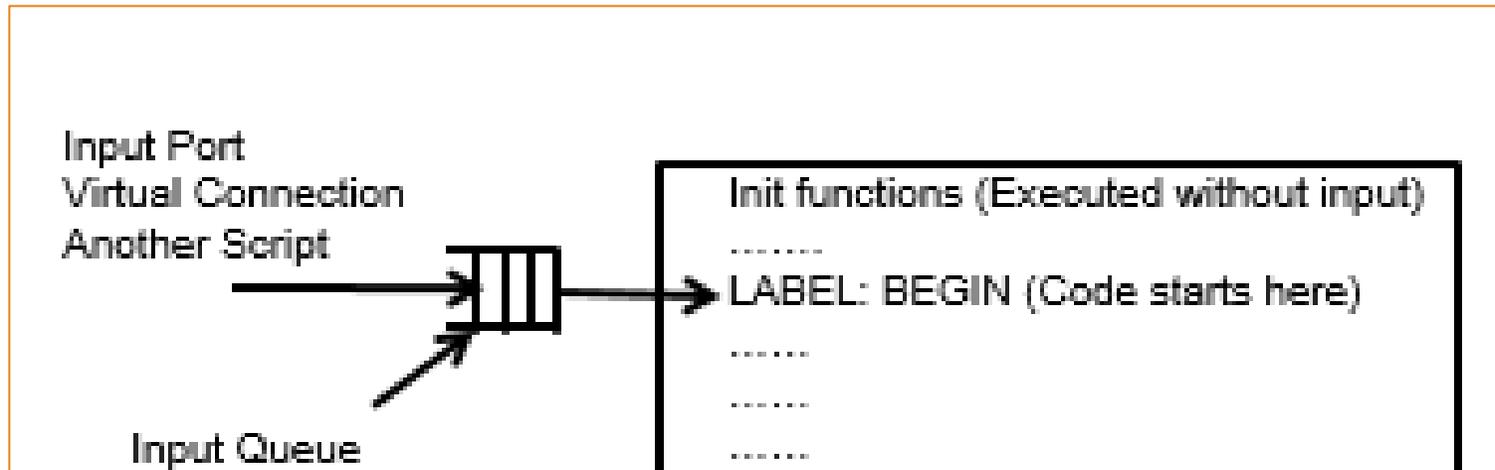
## NOTE

Model has to be saved before running a model which has script



```
1 /* List of Expressions. */  
2 Result_A = MyRegExpression_A_or_None  
3  
4 /* Logic Expressions: 'if', 'else if', 'else', 'while' supported  
5 if (x==y) {  
6     MyRegEx_Statements  
7 }  
8  
9 /* String Token Integer String Double */  
10 TIMEQ(Queue_Name, Token, Priority, Block, Time) /* Block  
11 QUEUE(Queue_Name, Token, Priority, Put ) /* Put,  
12  
13 /* String String String Integer Double */  
14 PLOT (Plot_Name, Dest, Color, Offset, Value ) /* Dest  
15  
16 GTO (-3) /* GoTo  
17 WAIT (1.0) /* Wait  
18 JIF ("Expression", 3) /* Jump I  
19 JIF ("Expression", 3) /* Jump I  
20 SEND ("output", Token) /* Send T  
21 /* End of Script */  
22
```

# How Does Script block work?



Even if the script has multiple input ports, the script doesn't have to wait for all input ports to have data to start execution

- When the packets comes into the script, they will be put into a queue called Input Queue
- If a script was idle and a packet comes, then it will be processed immediately.
- If a script was busy executing a packet and another packet comes into the script, then the latest packet will be in the Input Queue until the current execution finishes/stops.
- until the current execution finishes, it can also start the next one when the block DS stalls at a TIMEQ.

The Input Queue is a Virtual Queue

# Script Overview

---

1. port\_token is the pointer to the currently executing data structure.
2. Do not use the port name to identify the data structure.
3. Script block maintains the currently executing Data Structure at each line.

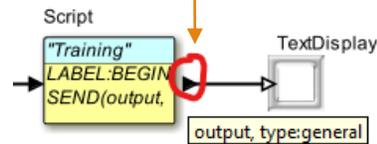
# SEND

FORMAT

**SEND(destination , data)**

```
1 /*
2  * Add logic here
3  */
4 SEND(output, port_token)
```

destination can be a port or a virtual connection like another script

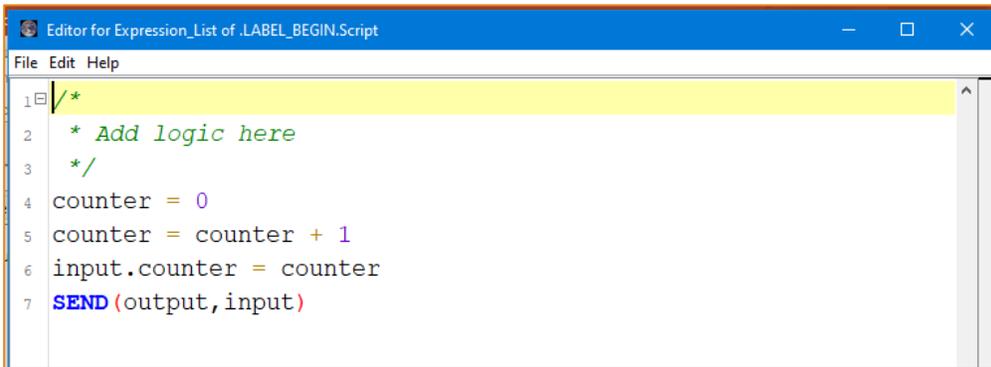


What is happening here?

Here, the data coming to the script is send out through the output port

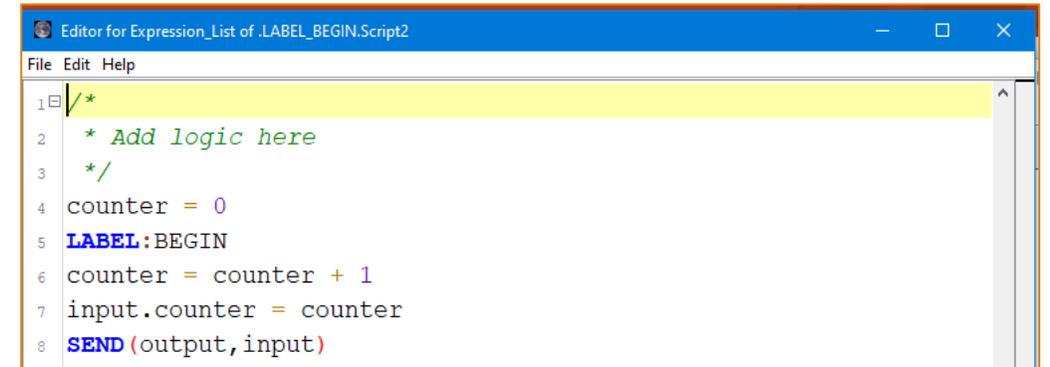
# LABEL:BEGIN

Implementing counter logic: Left or Right



```
1 /*
2  * Add logic here
3  */
4 counter = 0
5 counter = counter + 1
6 input.counter = counter
7 SEND(output,input)
```

Counter = 0 is executed every time script is triggered

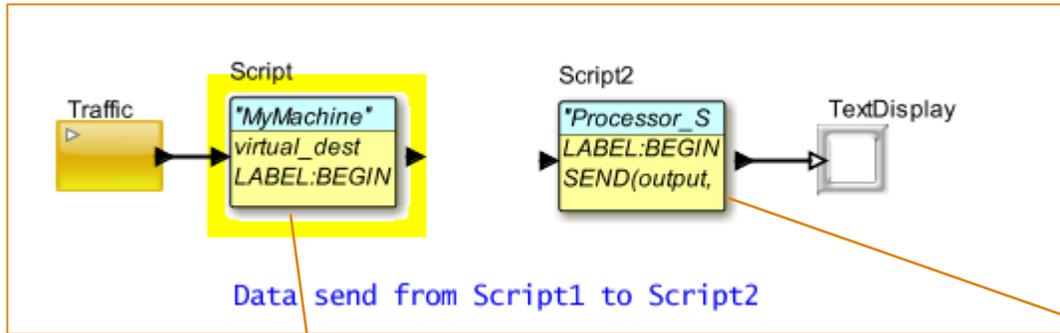


```
1 /*
2  * Add logic here
3  */
4 counter = 0
5 LABEL:BEGIN
6 counter = counter + 1
7 input.counter = counter
8 SEND(output,input)
```

Counter = 0 is only executed once during initialize

All lines of code defined before LABEL:BEGIN is executed at initialize and all lines of code after LABEL:BEGIN are executed when an input comes to script

# Can we use "SEND" to send packet to another Script block?



```
DISPLAY AT TIME          ----- 0.0 ps -----
{BLOCK                   = "Traffic",
DELTA                    = 0.0,
DS_NAME                  = "Header_only",
ID                       = 1,
INDEX                    = 0,
TIME                     = 0.0,
x                         = 250.0}
```

```
Editor for Expression_List of .Virtual_Connection.Script
File Edit Help
1 virtual_dest = "Processor_Script"
2 LABEL:BEGIN
3 port_token.x=100.0*5/6*irand(1,6)
4 SEND(virtual_dest,port_token)
5
```

```
Editor for Expression_List of .Virtual_Connection.Script2
File Edit Help
1 LABEL:BEGIN
2 SEND(output,port_token)
```

Packet comes into the second script through a port\_name called "virtual"

# GTO

```
Editor for Expression_List of .port_token.Script2
File Edit Help
1 /*
2  * Add logic here
3  */
4 counter = 0
5 LABEL:BEGIN
6 counter = counter + 1
7 GTO (END)
8 port_token.counter = counter
9 SEND (output,port_token)
```

GTO -> Go To  
GTO(END) -> finish script execution at this line  
So script execution stops at line 7.

Line 8 and 9 not executed

FORMAT

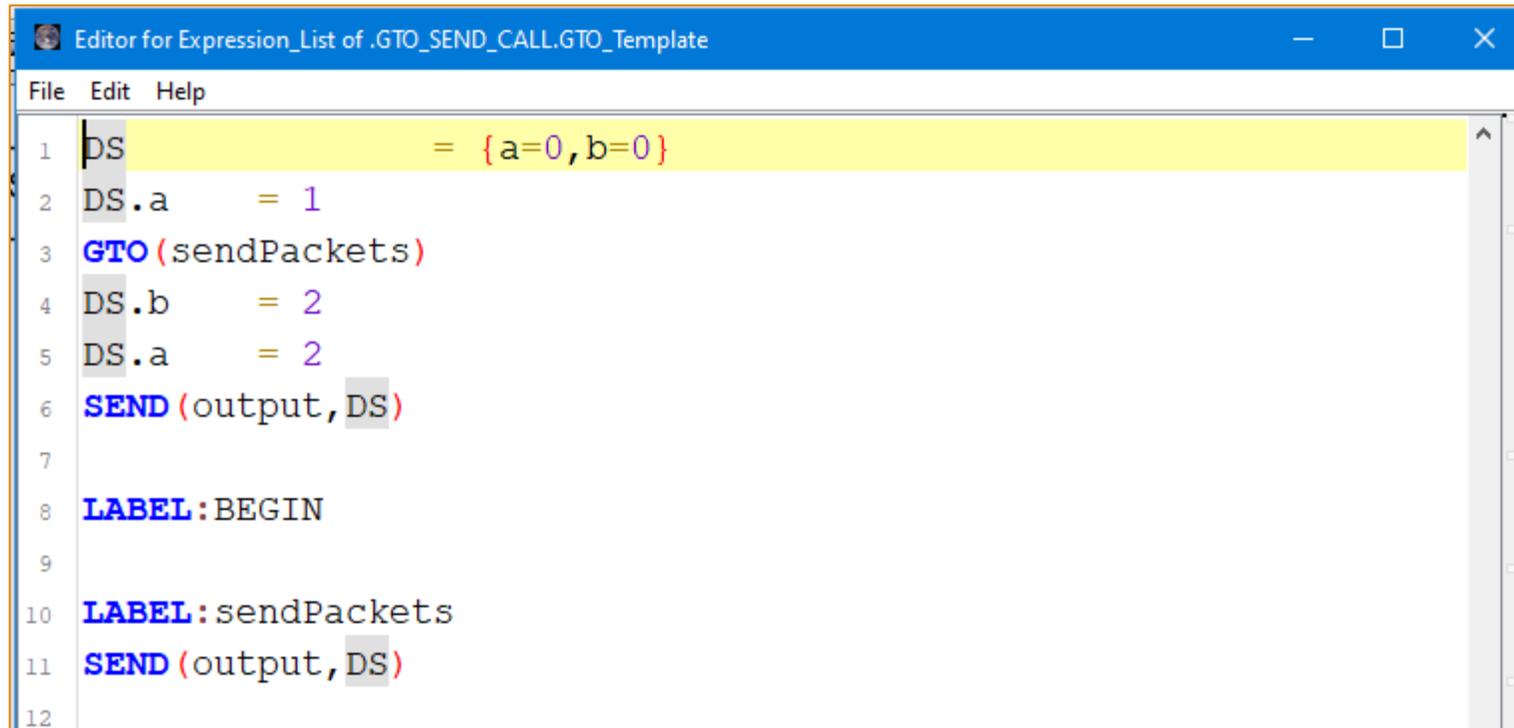
GTO(LABEL NAME)

```
Editor for Expression_List of .GTO.Script2
File Edit Help
1 /*
2  * Add logic here
3  */
4 counter = 0
5 LABEL:BEGIN
6 counter = counter + 1
7 GTO(send_packet)
8 port_token.counter = counter
9 LABEL:send_packet
10 SEND (output,port_token)
```

GTO(LABEL name) -> sends the execution to that LABEL  
Line 7 to Line 9.  
Line 8 wont be executed.

# What happens here?

---



```
Editor for Expression_List of .GTO_SEND_CALL.GTO_Template
File Edit Help
1 DS = {a=0,b=0}
2 DS.a = 1
3 GTO(sendPackets)
4 DS.b = 2
5 DS.a = 2
6 SEND(output, DS)
7
8 LABEL:BEGIN
9
10 LABEL:sendPackets
11 SEND(output, DS)
12
```

# SWITCH

```
Editor for Expression_List of .Flow_Control_RR_DRRSmart_Controller
File Edit Help
1 SWITCH (Scheduling_Algorithm) {
2   CASE:Round_Robin
3     //logic implemented here
4   BREAK
5   CASE:Deficit_Round_Robin
6     //logic implemented here
7   BREAK
8   CASE:DEFAULT
9     throwMyException("Undefined Scheduling_Algorithm (" +Scheduling_Algorithm+"")
10    BREAK
11 }
```

## Edit parameters for Flow\_Control\_RR\_DRR

SimTime:	1.0e-4
Queue_Depth:	100
Xoff_Rate:	610.0e6
Scan_Rate:	1500.0e6
Period:	10.0e-6
Scheduling_Algorithm:	Round_Robin
View_Stats:	Deficit_Round_Robin
In_Rate:	Round_Robin 20.0e6
Ingress_Size:	15

By using a SWITCH CASE logic, we select appropriate operation as per the parameter value.

FORMAT

```
SWITCH(Variable Name) {
CASE:
BREAK
CASE:DEFAULT
}
```

Note: Switch always requires a case (default)

# if – else

## FORMAT

```
if( condition ) {  
    <at least one line of code>  
}
```

Here, when the counter value reaches 10, we stop the script execution. Otherwise, we send the output.



```
Editor for Expression_List of .LABEL_BEGIN.Script2  
File Edit Help  
1 /*  
2  * Add logic here  
3  */  
4 counter = 0  
5 LABEL:BEGIN  
6 counter = counter + 1  
7 if(counter == 10) {  
8     GTO(END)  
9 }  
10 else {  
11     input.counter = counter  
12     SEND(output,input)  
13 }
```



# TNow and TStop

```
Editor for Expression_List of .TStop.Script2
File Edit Help
1 /*
2  * Add logic here
3  */
4 LABEL:BEGIN
5 TextMessage = "Pkt ID = "+port_token.ID+" . Current TIME = "+TNow+" and Simulation stop time = "+TStop
6 SEND(output,TextMessage)
```

TNow -> gives the current time in simulation

Tstop -> gives the simulation stop time

TStop is obtained from the top level Digital Simulator

```
VisualSim Architect - .TStop.TextDisplay2
----- 0.0 ps -----
DISPLAY AT TIME
Pkt ID = 1 . Current TIME = 0.0 and Simulation stop time = 10.0

----- 1.0000000000000000 sec -----
DISPLAY AT TIME
Pkt ID = 2 . Current TIME = 1.0 and Simulation stop time = 10.0

----- 2.0000000000000000 sec -----
DISPLAY AT TIME
Pkt ID = 3 . Current TIME = 2.0 and Simulation stop time = 10.0

----- 3.0000000000000000 sec -----
DISPLAY AT TIME
Pkt ID = 4 . Current TIME = 3.0 and Simulation stop time = 10.0

----- 4.0000000000000000 sec -----
DISPLAY AT TIME
Pkt ID = 5 . Current TIME = 4.0 and Simulation stop time = 10.0
```

# WAIT

```
Editor for Expression_List of .TStop.Script2
File Edit Help
1 /*
2  * Add logic here
3  */
4 WAIT(Tstop)
5 Msg = "Message after WAIT . Current TIME = "+TNow+" and Simulation stop time = "+Tstop
6 SEND(output,Msg)
7 LABEL:BEGIN
8 TextMessage = "Pkt ID = "+port_token.ID+" . Current TIME = "+TNow+" and Simulation stop time = "+Tstop
9 SEND(output,TextMessage)
```

During the delay period, entire script is locked up. No other line or transaction can be executed

WAIT(time) -> Provides delay for the time specified  
Until the delay is over, script cannot execute other transactions

FORMAT

WAIT( time )

```
VisualSim Architect - .TStop.TextDisplay2
DISPLAY AT TIME ----- 10.000000000000000 sec -----
Message after WAIT . Current TIME = 10.0 and Simulation stop time = 10.0

DISPLAY AT TIME ----- 10.000000000000000 sec -----
Pkt ID = 1 . Current TIME = 10.0 and Simulation stop time = 10.0

DISPLAY AT TIME ----- 10.000000000000000 sec -----
Pkt ID = 2 . Current TIME = 10.0 and Simulation stop time = 10.0

DISPLAY AT TIME ----- 10.000000000000000 sec -----
Pkt ID = 3 . Current TIME = 10.0 and Simulation stop time = 10.0

DISPLAY AT TIME ----- 10.000000000000000 sec -----
Pkt ID = 4 . Current TIME = 10.0 and Simulation stop time = 10.0

DISPLAY AT TIME ----- 10.000000000000000 sec -----
Pkt ID = 5 . Current TIME = 10.0 and Simulation stop time = 10.0
```

# TIMEQ

```
Editor for Expression_List of .TStop.Script2
File Edit Help
1 /*
2  * Add logic here
3  */
4 TIMEQ("stat_Queue",port_token,1,TStop)
5 Msg = "Message after TIMEQ . Current TIME = "+TNow+" and Simulation stop time = "+TStop
6 SEND(output,Msg)
7 LABEL:BEGIN
8 TextMessage = "Pkt ID = "+port_token.ID+" . Current TIME = "+TNow+" and Simulation stop time = "+TStop
9 SEND(output,TextMessage)
```

With TIMEQ, the current port\_token is stored in the Queue and a new one can be read from the input queue to start executing from LABEL:BEGIN

## FORMAT

**TIMEQ(Queue Name, data\_structure, 1, delay expression )**

With TIMEQ, we get line number 5 to execute at simulation stop time while other Data Structures (DS) will be executed

```
DISPLAY AT TIME          ----- 5.0000000000000 sec -----
Pkt ID = 6 . Current TIME = 5.0 and Simulation stop time = 10.0

DISPLAY AT TIME          ----- 6.0000000000000 sec -----
Pkt ID = 7 . Current TIME = 6.0 and Simulation stop time = 10.0

DISPLAY AT TIME          ----- 7.0000000000000 sec -----
Pkt ID = 8 . Current TIME = 7.0 and Simulation stop time = 10.0

DISPLAY AT TIME          ----- 8.0000000000000 sec -----
Pkt ID = 9 . Current TIME = 8.0 and Simulation stop time = 10.0

DISPLAY AT TIME          ----- 9.0000000000000 sec -----
Pkt ID = 10 . Current TIME = 9.0 and Simulation stop time = 10.0

DISPLAY AT TIME          ----- 10.0000000000000 sec -----
Message after TIMEQ . Current TIME = 10.0 and Simulation stop time = 10.0
```

# getBlockStatus

---

```
while (Queue_Num      <= Ingress_Size) {  
    Buffer_occupancy  = getBlockStatus(Queue_Name, "length", Queue_Num)  
    if (Buffer occupancy > 0 ) {
```

## FORMAT

**getBlockStatus(Queue Name, <keyword>,Queue Number)**

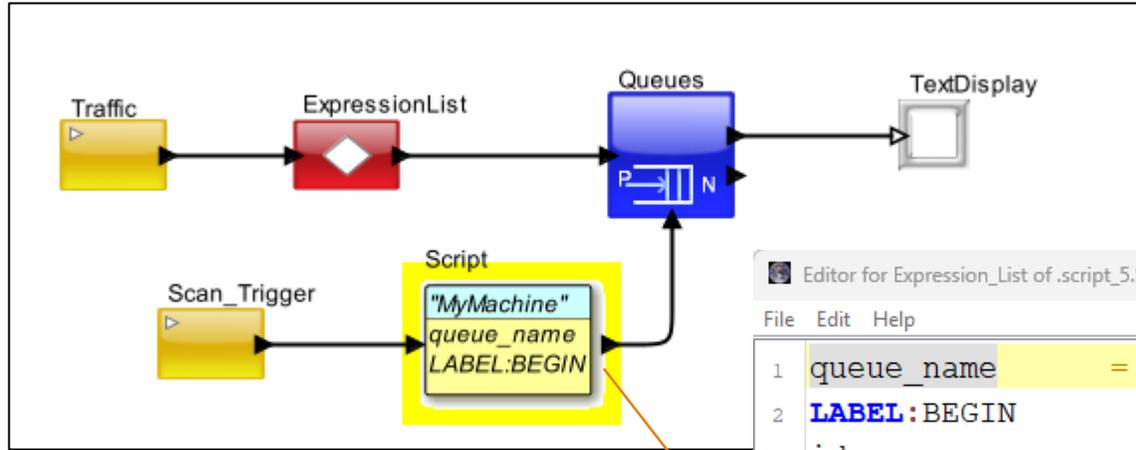
Keyword can be any of the following:

“length” -> gives us the current buffer occupancy ( no need to specify Queue Position )

“copy” -> gives us the copy of the packet present in the specified Queue

“pop” -> pop the Specified Queue virtually

# Arbitration algorithm : Queue + Script



```
Editor for Expression_List of .script_5.Script
File Edit Help
1 queue_name = "Ingress_Queue"
2 LABEL:BEGIN
3 idx = 1
4 while(idx <= Number_Of_Queues) {
5     qLen = getBlockStatus(queue_name, "length", idx)
6     if(qLen > 0) {
7         SEND(output, idx)
8         GTO(END)
9     }
10    idx = idx + 1
11 }
12
```

# Using regex to send data to script Virtually

Virtual (<script\_block\_name>, <data\_to\_be\_sent>)

The image shows a screenshot of the VisualSim Architect software interface. On the left, there are two 'Edit parameters' windows. The top window, titled 'Edit parameters for latency\_calc', shows a 'Block\_Documentation' field with a multi-line regex template and an 'Expression\_List' field containing the code: `virtual(Cluster_Name+"_Latency",input)`. The bottom window, titled 'Edit parameters for printToCmdLine', shows a similar 'Block\_Documentation' field and an 'Expression\_List' field containing: `virtual(Cluster_Name+"_MIPS",input)`. Both instances of the `virtual` function call are circled in orange. On the right, the main VisualSim Architect window is open, displaying a block diagram with two script blocks: 'Script' and 'Script2'. Each script block is connected to a core component, 'Core\_Latency' and 'Core\_MIPS' respectively. The 'Script' block has a 'coreArr=' parameter set to 'LABEL:BEGIN'. A library tree on the left side of the main window lists various components like Document, Model Setup, Traffic, Results, File IO, Behavior, Mappers, Resources, Power, Hardware Setup, ProcessorGenerator, Cycle\_Accurate\_Processor, Memory, HardwareDevices, Interfaces and Buses, Full Library, and UserLibrary. At the bottom of the main window, there are buttons for 'Commit', 'Add', 'Remove', 'Restore Defaults', 'Preferences', and 'Help'.

---

# Task Graph Modeling



# Blocks Required To Model a Basic Task Graph in VisualSim

---

Traffic Block – Explained in the Section Traffic

Traffic Generator – Explained in the next section

Trace Mapper – Explained in the next section

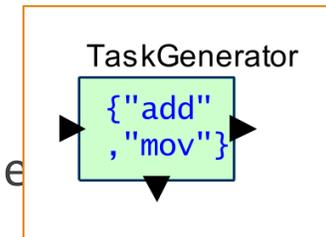
Mapper Blocks – Explained in the Library Blocks Power Point

System Resource or Processor to represent the hardware on which the task graph will be executed -  
Explained in the Library Blocks Power Point

# Task Generator Module

---

- Custom Task generator – number of instructions, type of instructions, order of tasks (loop, random) can be set
- More dynamic and distributed traffic profile can be generated
- “n” number of Software tasks can be defined
- In case software development hasn’t started yet, we can use this module to generate instruction traces



# Task Generator – Config File (Instruction Mix Table)

Task\_Name

Call\_Fn\_Low  
 Call\_Fn\_High  
 Sensor  
 Read\_Frame  
 Decode\_Frame  
 Video\_Post\_Proc  
 Render\_Frame  
 Format\_Conversion  
 Rotate\_Frame  
 Display

Relative\_Time(double)\_or  
 Number\_Instructions(int)

276  
 573  
 150  
 100  
 160  
 200  
 500  
 100  
 1000  
 100

Type	Pct	Type	Pct	Type	Pct	Type	Pct	Type	Pct	Type	Pct	Type	Pct	Type	Pct	Type	Pct	*/
IFU	25.0	BU	5.72	SU	10.0	LU	15.0	PU	0.43	LDU	25.0	STU	15.0	ETC	3.85			;
IFU	40.0	BU	11.36	SU	3.8	LU	15.1	PU	0.85	LDU	12.2	STU	6.1	ETC	10.59			;
IFU	39.7	BU	10.20	SU	7.2	LU	0.0	PU	0.76	LDU	23.6	STU	10.9	ETC	7.64			;
IFU	57.0	BU	10.22	SU	0.0	LU	0.0	PU	0.76	LDU	14.4	STU	0.2	ETC	17.42			;
IFU	20.6	BU	11.5	SU	16.9	LU	0.0	PU	0.86	LDU	16.2	STU	7.9	ETC	26.04			;
IFU	20.6	BU	11.5	SU	16.9	LU	0.0	PU	0.86	LDU	16.2	STU	7.9	ETC	26.04			;
IFU	33.9	BU	9.65	SU	2.6	LU	10.5	PU	0.72	LDU	21.7	STU	18.1	ETC	2.83			;
IFU	35.8	BU	18.68	SU	11.4	LU	8.6	PU	1.40	LDU	16.1	STU	4.9	ETC	3.12			;
IFU	45.1	BU	6.23	SU	13.0	LU	20.5	PU	0.46	LDU	7.1	STU	6.7	ETC	0.91			;
IFU	2.00	BU	6.00	SU	12.0	LU	10.0	PU	0.00	LDU	35.0	STU	35.0	ETC	0.0			;

Software tasks

Number of instructions per task

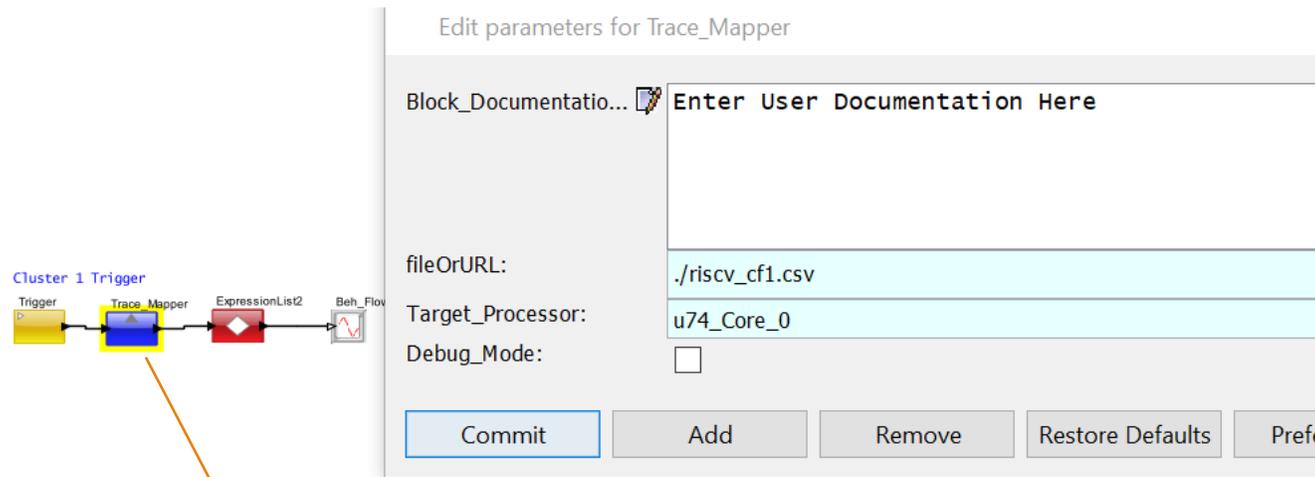
The Total Number of instructions are made up of instructions of different types. The percentages of each type of instruction is specified here.

# Task Generator - Config File (Instruction Mix Table)

Description	Type (used below)	Mnemonic List */
Int_Float	IFU	add sub sbcs ccmn eon eor cmps andhi addgt dmb;
Branch_Unit	BU	b br beq bcc bge tbnz;
Shift_Unit	SU	movs movw mvn asrv;
Logic_Unit	LU	and add orn bics csel ands subne bichi ;
Mispredict_Unit	PU	*b ;
Load_Unit	LDU	ldr ldrsw prfm ldur ldurb ldurh ldursb ;
Store_Unit	STU	stur sturb sturh str strb ;
Others	ETC	casa stxr uqadd16 shsub8 smulls umaal sdiv udiv ;

This type descriptor is used in the previous slide. User can specify the percentage of each type of instruction for each software operation

# Trace Mapper and structure of CSV File containing traces



**Note:** File name to be specified in CSV format only

```
1 I_Cache_Address,A_Instruction,D_Cache_Address
2 array,array,array
3 {"0x00000000","0x0000005c",...},{"ldr","mrc",...},{"0x00000020","0x00",...}
```

# Trace Generation using GEM5 or other simulators compatible with VisualSim

Source Code

Compile

Run on Spike

Generate log

Parse log – remove pk overhead

Gen. data in csv format

Load into the model

```
13 #include <stdlib>
14 #include <string>
15 #include <iostream>
16
17 #include "Parallel.hh"
18 #include "InputFile.hh"
19 #include "Driver.hh"
20
21 using namespace std;
22
23 int main(const int argc, const char** argv)
24 {
25     Parallel::Init();
26
27     if (argc != 2) {
28         if (Parallel::ispp == 0)
29             cerr << "Usage: pennant <filename> << endl;
30         exit(1);
31     }
32
33     const char* filename = argv[1];
34     InputFile inp(filename);
35
36     string probname(filename);
37     // strip .opt suffix from filename
38     int len = probname.length();
39     if (probname.substr(len - 4, 4) == ".opt")
40         probname = probname.substr(0, len - 4);
41
42     Driver drv(&inp, probname);
43
44     drv.run();
45 }
```

riscv64-unknown-elf-gcc  
./Source\_Code/main.c

./riscv-isa-sim/build/spike -l --log-commits --isa=RV64IMAC ./riscv-pk/build/pk a.out 2>gcc\_out.log

```
2467900 core: 0: 0xffffffff00001eb2 (0x0e053e03) ld t5, 224(a0)
2467909 core: 0: 1 0xffffffff00001eb2 (0x0e053e03) x28 0x0000000000000000 mem 0xffffffff00041bf0
2467910 core: 0: 0xffffffff00001eb2 (0x0e053e03) ld t4, 232(a0)
2467911 core: 0: 1 0xffffffff00001eb2 (0x0e053e03) s29 0x0000000000000000 mem 0xffffffff00041bf0
2467912 core: 0: 0xffffffff00001eba (0x0f053f03) ld t5, 240(a0)
2467913 core: 0: 1 0xffffffff00001eba (0x0f053f03) x30 0x0000000000000000 mem 0xffffffff00041bf0
2467914 core: 0: 0xffffffff00001eba (0x0f053f03) ld t6, 248(a0)
2467915 core: 0: 1 0xffffffff00001eba (0x0f053f03) x31 0x0000000000000000 mem 0xffffffff00041bf0
2467916 core: 0: 0xffffffff00001ec2 (0x00006928) c.ld a0, 80(a0)
2467917 core: 0: 1 0xffffffff00001ec2 (0x06928) x10 0x0000000000000000 mem 0xffffffff00041bf0
2467918 core: 0: 0xffffffff00001ec4 (0x1020073) s.ret
2467919 core: 0: 1 0xffffffff00001ec4 (0x1020073) c7eb_mstatus 0x8000000a00066a2
2467920 core: 0: 0x00000000000011010 (0x02e008e4) jal pc + 0x2e
2467921 core: 0: 0 0x00000000000011010 (0x02e008e4) x1 0x00000000000011014
2467922 core: 0: 0x0000000000001103e (0x0013f197) auipc gp, 0x13f
2467923 core: 0: 0 0x0000000000001103e (0x0013f197) x3 0x0000000000015003e
2467924 core: 0: 0x00000000000011042 (0x0b18193) addi gp, gp, 582
2467925 core: 0: 0 0x00000000000011042 (0x0b18193) x3 0x000000000014fd8
2467926 core: 0: 0x00000000000011046 (0x0000802) ret
2467927 core: 0: 0 0x00000000000011046 (0x00802)
2467928 core: 0: 0x00000000000011014 (0x000087aa) c.mv a5, a0
2467929 core: 0: 0 0x00000000000011014 (0x87aa) x15 0x0000000000000000
2467930 core: 0: 0x00000000000011016 (0x00000517) auipc a0, 0xc0
2467931 core: 0: 0 0x00000000000011016 (0x00000517) x10 0x00000000000011016
2467932 core: 0: 0x0000000000001101a (0xa8650513) addi a0, a0, -1402
2467933 core: 0: 0 0x0000000000001101a (0xa8650513) x10 0x0000000000010a9e
```

```
81 print(trace name)
82 with open(input_file_name, "w") as arm_trace_file, open(input_file_path+output_file_name, "w") as arm_trace_file:
83     trace_writer = Writer(cache_address, instruction_cache_address, memory_array)
84     instr_line_parsed = False
85     d_addr_trace_text = ""
86     instr_trace_text = ""
87     d_addr_trace_text = ""
88     check_for_new_line = True
89     for lines in arm_trace_file:
90         line = arm_trace_file.readline()
91         if (<<>> not in lines and (" in lines):
92             sub_line = lines.split("\n")
93             sub_line = sub_line[0]
94             sub_line = sub_line.strip()
95             non_whitespace_sep = sub_line.strip().split(" ")
96             if check_for_new_line and starting == 0:
97                 check_for_new_line = False
98                 d_count = d_count + 1
99                 if "mem" in lines:
100                     mem_arr_lines = split("mem")
101                     non_whitespace_sep = non_whitespace_sep[1].strip().split(" ")
102                     d_addr_trace_text = d_addr_trace_text + " " + non_whitespace_sep[0]
103                     else:
104                         d_addr_trace_text = d_addr_trace_text + " " + "0x" + line
105             continue
106             addr = non_whitespace_sep[0]
107             instr = non_whitespace_sep[1]
108             if hex_to_decimal(addr) != hex_to_decimal(starting_instr_addr):
109                 start_instr = True
110                 start_instr = False
111             continue
112             if start_instr:
113                 if hex_to_decimal(addr) != hex_to_decimal("0xffffffffffffffff"):
114                     out_of_range_flag = True
115                     continue
116                 elif out_of_range_flag and "0x" not in addr:
117                     continue
118                 else:
119                     out_of_range_flag = False
120             if instr_count == num_of_instr_per_line:
121                 if !is_contained_count:
122                     raise ValueError("Array size mismatch")
123                 d_addr_trace_text = d_addr_trace_text + "\n" + d_addr_trace_text[1:]
124                 instr_trace_text = instr_trace_text + "\n" + instr_trace_text[1:]
125                 d_addr_trace_text = d_addr_trace_text[0:len(d_addr_trace_text)-1]
126                 instr_trace_text = instr_trace_text[0:len(instr_trace_text)-1]
```

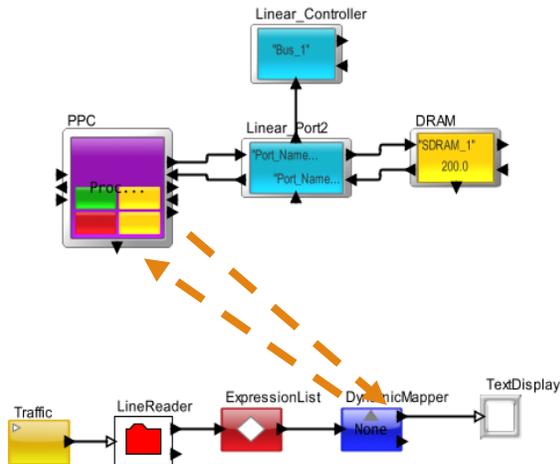


VisualSim Architect

# Dynamic Mapper

## Mapping of tasks on

- Target processor,
- SystemResource
- SystemResource\_Extend



Edit parameters for DynamicMapper

Block\_Documentatio...

Block_Name:	SW_Mapper
Database_Lookup:	None
Task_Name:	A_Task_Name
Target_Resource:	Board_Name + ".PPC_7410_1"
Task_Instruction:	A_Instruction
Task_Plot_ID:	1
Task_Number:	A_Task_ID
Task_Priority:	A_Priority
Task_Time:	A_Time
Database_Expression:	None /* Advanced Feature: can use for any DB_Fld_Name below with database name */

Commit Add Remove Restore Defaults Preferences Help

# Model Showcasing the usage of TraceMapper and Task generator

