

Mirabilis Design Inc. Quick Reference Card

What is VisualSim

- System-level modeling and simulation software.
- Systems architecture exploration.
- For electronics, software, network and semiconductors.
- Analyze timing, power, and functionality.
- Validate specification for non-functional need- requirements, cost, resources etc

Examples of Usage

- Aerospace - avionics, flight control, inertial navigation, communication, radar
- Communication- Software-Defined Radio, DSP-based communication systems, Antenna, 5G
- Networking - Custom protocols, arbitration, flow control
- Automotive - Networks, security, safety, ECU
- Compute - SoC, processor, FPGA, TPU, HPC, media servers, switch
- Storage - SSD, HDD, NAS, DDRs, LPDDRs, HBM

What can you do in VisualSim?

- Resource sizing and capacity planning
 - Size to support use cases and workloads
 - Combination of components, topology, speeds, width, bandwidth, buffers, cache, memory and power states
 - Verify that the statistics meets the requirements.
- Failure analysis
 - Timing and throughput when processing/ storage/ network failure, change in software scheduling, network intrusion, reduced power and modify memory tables.
- Abstract design to cycle accurate design.
 - Build models in different levels of abstraction to focus on specific part of the system or function.
 - Hybrid processor with cycle accurate memory architecture.
 - Model behavioural components as resources to analyse timing and power.
- Power management, sizing and generation
 - Experiment with different sets of power states and power management based on capacity and utilization Extend the battery life.
 - Size the power generator and battery.
 - Metrics for instant, total, average and individual device power.
- Hardware-software partitioning and Task graph analysis
 - Sequence the threads and processes across multiple

processors and network nodes.

Partitioning across AI tiles, Tensors, Deep Learning Units, CPU, GPU, DSP etc

- Select the functions that need acceleration. Generate document, test bench and UPF power table for implementation and communication
- Create custom and classified libraries and share it with others by encrypting the library.
- Software modeling for the hardware components with simplified methods without affecting the accuracy.
- Run Complex models with different combination of parameters as a batch run and compare results from different runs.
- Debug and verify the design using diagnostic feature. Validate whether it meets the requirement or not.

Working with VisualSim

1. Draw a block diagram of your proposed system on paper.
2. Open a new Block Diagram Editor.
3. Drag blocks from the library, set parameters, and connect to create an architecture.
4. Define the Task Graph of the system behavior. The sensors and input interface must be defined in the architecture
5. Map each task to an element or master in the architecture
6. Simulate the system model for various configurations, topologies, workload, and use cases.
7. Define the requirements in the Diagnostic format. Require Diagnostic block in model
8. Open the model in the Post Processor after adding the PlotManager in the model. Define the list of parameters and the ranges of values to simulate. Select the plots and display to show. An Index file with the list of all possible runs and a few other configuration files are created.
9. Run the Thread Sim from the command-line to distribute the selected simulation runs across all available cores.
10. Review the generated statistics text files, recommendations on the requirements and compare plots using Post Processor
11. Auto-Save is a setting in the Config.Properties that determines how often the currently modified file is being saved to <User Home>/.VisualSim/ directory. The file name is name+time_and_date.xml. Note: The file is never deleted.
12. Save and restarts is an option to save the state of the simulation at distinct points during a simulation

and restart from that point onwards. The sim times at which to save are set in PauseSimulationAt.txt which is located in the same directory as the model.

VisualSim Cloud

- Online version of the tool, that user can run without installing the VisualSim in the local system.
- Requires OpenWebstart and Java 17 or higher.
- User can share the model from installed tool by export to web option and the user with VisualSim cloud can use that file to run and analyse the design

Parser and External tool support

- In built parser to convert files from external software or hardware to VisualSim.
- ARXML, AXI, GEM5, GCC and Spike**
- External tool to VisualSim, Support files from other software to load in VisualSim models.

Batch Run and Multi Core Processing

- **Batch Run**
 - Inbuilt Post Processor allows user to create batch files for a particular model to run multiple combinations of parameters.
 - Validate the system stability.
 - Enables recursive analysis and comparison with different runs.
 - Combine plots from different runs in a single window.
- **Simulation on Multi core**
 - Post processor creates files to run the batch simulation on the local system by assigning different runs to different cores of the system.
 - Increases parallelism and fast simulation.

GUI Concepts

- **Block**
 - Defines basic functionality such as traffic, FIFO, arbitration, processor, storage, and statistics generation.
 - Contains input/output ports, parameters.
 - Initializes block based on the parameters and functionality.
 - Logic executes when inputs are received.
 - Inputs can be read from files
 - Contains optional timing and power.

- Blocks are color coded (Yellow – Traffic Generation / Traffic Reader/ Database, Red –Power Management, variable list, ExpressionList, Blue – Behavior Mapping, Queuing/System/Server resources).
- Block Parameters
 - Can reference a parameter or data structure field.
- Ports
 - Input, Output or multiport (input-output).
 - Polymorphic –scalar, string or data structures.
- Block Ports
 - If port has specific data type, then data coming through the connected port must be the same type.
 - Direction can be modified (Customize Ports).
 - Configure > Ports – Can add in/out ports to ExpressionList and Script blocks.

• Wires

- Represent a connection between ports.
- Does not introduce any delays.
- To connect port-to-port, press and hold at a port, then drag, and release when connected to the other port.
- If connected, line is thick, else thin.

• Relations

- Connect multiple input and Output ports.
- To connect, hold “Ctrl” key, and drag from relation to port.
- Can connect relation to relation to arrange wiring.

• Annotation

- Represent comments and documentation about the model.
- Multiple annotation can be added and placed anywhere in the model to describe each block and flow.

Block folders

- Document – Describe the model
- Model Setup – Basic model requirements
- Traffic – Generate transactions
- Results – Output the simulation results
- File IO – Read or write data from or into files and database
- Behavior – Implement the logic for the model
- Mapper – Map the task to the resource
- Resource – Consume time, quantity, and power
- Power – Generate, store, consume, manage, and

Mirabilis Design Inc. Quick Reference Card

<p>report</p> <ul style="list-style-type: none"> • Hardware setup –Device modeling and overall system setup • ProcessorGenerator – New processor template and models of commercial processors. • Cycle_Accurate_Processor – Pipeline stages for microarchitecture design. • Memory – Model statistical and cycle-accurate Memory and cache • Hardware Device – DMA, Buses and peripherals • Interfaces and Buses – Standards-based buses, interconnects, and networks 	<ul style="list-style-type: none"> • Listen to block: Right-click on the block and select Listen to Block. Shows the sequence of execution in block. Verify if operation is correct. Listen to block can be enabled for every block except hierarchical blocks. • Script Tracer: captures the execution of multiple scripts and write the content in one file. Trace file will be saved in (User folder)/.VisualSim. Helps to check if dependent blocks have deadlock or error. • Listen to port: Right-click on the port and select. Check if the field values are correct. • Animate Execution: Debug -> Animate Execution. Shows order of execution. Identify if blocks are not executing, executing too many times or executing in the wrong order. • Block Breakpoints: Right-click on block and select Breakpoint. Stops model at a specific time in the block. Do a Listen to Block or Port at any block • Pause and resume: Stop and step through each event in the model PauseSimulationAt.txt and store in same directory as model. Simulation stops at each timestamp to save current state of variables, queues, and Script line. Simulation can be started from any of the specified time points. • Digital Debugger-> Pause / Summary – Only / Run: Pause-stop simulation and restart. Summary –consolidated value of time/number of executions of all blocks. Run-block execution sequence. • Diagnostic: Add requirements and constraints in a text file and link it to the Diagnostic block. Simulation will generate statistics and recommendation files, user can check if the requirements are met. • Additional: Text Display, Plotter, statistics, and fileWriter to trace intermediate activity. 	<p>is for the whole model.</p> <ul style="list-style-type: none"> ▪ Block variables are defined and used in Script and ExpressionList blocks. ▪ Example: Buffer_Usage in Egress Queues or device available flag. 	<p>(a) Block Name for Resources, Script, Architecture and Application-specific blocks must be unique in entire model</p> <p>(b) Variables:</p> <ol style="list-style-type: none"> Global must be unique in the entire model Local and Global names cannot overlap Local and Block variable can overlap if they are not in the same window or BDE <p>(c) Parameters</p> <ol style="list-style-type: none"> Link from block in lower-level hierarchy to the parameter in the next-level above. Never jump from a lower-level block or hierarchy to top-level parameter SharedParameter are stopTime and tResolution <p>(d) Instantiating simulator</p> <ol style="list-style-type: none"> Required in top-level of Model, DynamicInstantiation and Class
<p>User Documentation</p> <p>(a)Annotation: Provides textual information in the Block Diagram Window. (b)Block Documentation: Available in all blocks. Details of the block activity and internal expressions can be added. Generated with Export to Web. (c)Comments (/ *....*/): Comments can be in parameter, variable list, and any expression or script.</p>		<p>• Transaction or Data Structure</p> <ul style="list-style-type: none"> ▪ Container with fields of any data type. ▪ Represent data details, control, and simulator details. ▪ Example: Data, Priority, Command, Source, Destination, Time Stamp <ul style="list-style-type: none"> • SampleDS = {Source_Name = "Sensor1", Destination_Name = "DRAM", Priority = 1, DataSize = 64 , CRC = "F822", Length = 80} 	<p>ii. Note: All parameters of the lower-level Simulators are ignored</p> <p>(e) Ports</p> <ol style="list-style-type: none"> Input connect to output, except between block output and window output <ol style="list-style-type: none"> Multiport and In-Out port allows input and output of a device to be connected <p>(f) Zero-delay loop</p> <ol style="list-style-type: none"> When the wires between a set of ports are considered a loop and there is no delay interrupting this loop, a zero-delay loop error will be generated To resolve this, add a Delay block with a value of 0.0 <p>(g) Virtual Connection</p> <ol style="list-style-type: none"> IN, OUT, Script to Virtual Connection and scripts. Target names can be local (same window) or global (entire model). Multiple Sources can go to a single Script or OUT block. Node block- connection-less. No wiring required. Connections attribute are captured in the database associated with the Routing Table. <p>iii. Mapper-to-SystemResource: Names must match</p> <p>(h) Random Number Generators</p> <ol style="list-style-type: none"> No seed- random() and Gaussian. This means that results from every run are different. It is not deterministic. Seed- rand, irand. Define model_seed as a parameter with a value of seed(Long Value). For every seed value, the outputs will be deterministic
<p>Library Management</p> <ul style="list-style-type: none"> • Creating Reusable Blocks or Class <ol style="list-style-type: none"> 1. Select blocks and parameters by left click and drag. 2. Click Graph > Create Hierarchy, Disconnect all connections to the created Hierarchical Block. 3. Right-click the block and select 'convert to class'. 4. Right-click the block and select Open Block. 5. File > Save As (Directory must be under CLASSPATH – VS_AR) and select "Save Submodel Only". 6. save the file with .xml extension. • Import Custom Blocks in Model <ol style="list-style-type: none"> 1. Graph > Instantiate Class. 2. Browse and select .xml file. Commit. • Adding Reusable Blocks to UserLibrary <ol style="list-style-type: none"> 1. Add block to model. 2. Right-click block and select "Save Block in Library". 3. UserLibrary.xml is opened and select File > Save. 4. Block available in Folder > UserLibrary. 	<p>Methodology Concepts</p> <ul style="list-style-type: none"> • Parameters <ul style="list-style-type: none"> ▪ Constants during simulation. ▪ Model Setup -> Parameter / Range / Pulldown. ▪ Example: Processor_Clock = 250.0, Traffic_Interval = 1.0e-6 • Variables <ul style="list-style-type: none"> ▪ Similar to programming variables or hardware registers. ▪ Defined in VariableList (Types – local and global). ▪ Script block can define a global variable which can be used in other blocks with the help of RegEx commands ▪ Local is accessed only in current window and global 	<p>• RegEx (Regular Expressions)</p> <ul style="list-style-type: none"> ▪ Used in Parameter (Processed once at startup), ExpressionList (Every entry), and Script (Every entry or loop). ▪ Collection of Math, Logic, Statistics, power access, resource internal access, array processing, and programming option for popular GUI functions. ▪ Example: Buffer=getBlockStatus("Egress"); a = x.cos(90) ▪ Can include parameters, variables, and fields in the expression. ▪ Parameters can be used in LHS (not in ExpressionLists or Script blocks) <p>• Script</p> <ul style="list-style-type: none"> ▪ Programming language for modeling ▪ Used to define algorithm and common data processing operations. ▪ Supports RegEx and common C functions. ▪ Blocking and non-blocking timing. ▪ Supports Virtual Connection ▪ Can be modelled to implement any algorithm or logic ▪ Can define Queue within a script ▪ Can access and remove content of a Queue block virtually. ▪ Power calculations can be made for the application or algorithm implemented in the script. 	
<p>Debugging</p> <ul style="list-style-type: none"> • Block highlighting: When error occurs, the block with the error is highlighted in red. Narrows down which block has the error. • Error Messages: Provide block location in design hierarchy and error details. 		<p>Scope</p> <p>The scope of names plays a major role in VisualSim. Here are the important items.</p>	

Mirabilis Design Inc. Quick Reference Card

across any number of runs	<ul style="list-style-type: none"> Failure Analysis <ul style="list-style-type: none"> Inject fault or error into the system to observe the stability. Can be a data error, unsupported size, variable packet arrival rate, link failure, resource failure, etc. Blocks: Traffic, Expression list, script 	<p>Connect Behavior and Architecture</p> <ul style="list-style-type: none"> Examples: Image read, decode, rotate, and transpose on ARM. Concept: (a) Used to add timing and power for each function. (b) Mapper block (Behavior flow) sends transaction to SystemResource or Processor (Actual Hardware Unit). (c) Can be a static (fixed mapping) or dynamic (resource selection is based on task type, first available resource or prior executions). Block: (a) Behaviour: Mappers, SoftwareMapper, DynamicMapper. (b) Architecture: SystemResource, SystemResource_Extend, SystemResource_Done Operation: (a) Required fields for the SystemResource are A_Time field has execution time, A_Priority field has the task Priority, and A_Task_ID field has task identifier. (b) Mapping name must match a SystemResource. (c) Mapper (behavior control block) sends the transaction to the SystemResource or SystemResource_Extend (followed by Done block) for processing. 	<ul style="list-style-type: none"> Random Early Detection
<p>Modeling Methodology</p> <ul style="list-style-type: none"> Select the methodology for your system. Criteria for selection – Familiarity with modeling method, shared resources across flows or hardware or software only. In-Line Flow Descriptions <ul style="list-style-type: none"> Typical for stochastic studies, protocol, and Ingress-Egress models. Shared resources accessed by individual flow in a central location. Main blocks: Traffic, packet attributes (ExpressionList), timing (Queue, Server), and Script for arbitration Separation of Behavior and Architecture <ul style="list-style-type: none"> Y-chart design approach Behavior flow is the sequence of functional operation in concurrent threads. No algorithm details or delays considered. Architecture describes the platform – electronics, RTOS, and middleware. For each task in the flow, use mapper block to link to SystemResource or Processor. Main fields: Timing (A_Delay), Priority (A_Priority), Task ID (A_Task_Address), and Task Name (A_Task_Name). Hardware-only flow <ul style="list-style-type: none"> Architecture is built using cycle-accurate hardware blocks. Traffic triggers the Processor, DMA or Device Interface. Traffic can be data arriving at an interface, Instruction sequence to a processor and Read/Write to Memory. Main blocks: Processor, Memory, Cache, DMA, Bus, Switch, Bridge. Define Accelerator: DeviceInterface + SystemResource + Bus Link for resource access. Software and hardware <ul style="list-style-type: none"> Instruction sequence and processing elements Define software functionality using the Instruction set and Instruction mix table. Generate tasks based on the instruction mix table and allocate task to specific resource or processing element based on the allocation algorithm. Main Blocks: Processor, System Resource, Mapper, Dynamic Mapper, Task Generator, Instruction Set 	<p>Modeling Details Workload</p> <ul style="list-style-type: none"> Examples: Traffic to represent trigger behavior flow, network traffic, instruction sequence, Read/Write request, Customers arriving at a bank teller. Description: Data Structure sent out to emulate the packet/command/customer. Blocks: Two main types – Distribution based (Traffic block), Trace file based (TrafficReader). Use Script block to generate custom traffic. <p>Workload Attributes</p> <ul style="list-style-type: none"> Examples: Defines the workload such as priority, data size, data value, type, address. Concept: (a) Update existing or new fields of incoming transaction. (b) Use RegEx, parameters, fields and variables to provide computed values or to select options. Blocks: (a) ExpressionList and Script. (b) Support blocks are Boolean switch, fork, and join. <p>In-line behavior and architecture</p> <ul style="list-style-type: none"> Examples: Multi-port network switch, DSP functions in an FPGA, manufacturing flow. Concept: Define the sequence of activity with queues, arbitration, flow control, and processing times. Blocks: Traffic, ExpressionList, Script, Queue, Server <p>Behavior Flow, Process, Thread and Tasks</p> <ul style="list-style-type: none"> Examples: Imaging pipeline, Customer consumption of resources at a restaurant. Concept: (a) List the sequence of functional processing on the data. (b) without the functional description. (c) Can be hardware or software implementation. Block: ExpressionList, Mapper, Script, FSM, Fork, and Join. Operation: (a) Behavior flow is modeled with Expression List and Mapper blocks. (b) Each function in the sequence can update field values. (c) and decision trees and map to the SystemResource. (d) Use Fork and Join for visual appeal. 	<p>Scheduler Block with added functionalities</p> <ul style="list-style-type: none"> Provides more scheduling algorithms for the user to choose from. The following are the different Queue rejection mechanisms supported : <ul style="list-style-type: none"> Incoming Token Rejected/ Tail drop Lowest Priority Token Rejected Random Drop on full Drop front on full 	<ul style="list-style-type: none"> The following are the different Scheduling algorithms supported : <ul style="list-style-type: none"> FIFO FCFS Round Robin Weighted Round Robin Deficit Round Robin Weighted Fair Queuing Strict Priority Round Robin Priority If the user selects Deficit round robin or Weighted fair queueing, a field called Task_Size is necessary. <p>Queuing Resources with pre-determined processing time</p> <ul style="list-style-type: none"> Examples: (a) Bus or network delay, processing with no pre-emption.(b) Cross a traffic intersection. (c) All hardware blocks that consume time. Model time – and partition-based scheduling. Concept: (a) Combines the queue and the processing in a single block. (b) Has multiple concurrent flows represented. (c) Slot based schedule provides different processing time for each queue in order. Block: Server Input and Output: Transactions Key Fields: Queue_Number_Field (A_Task_Address), Priority (A_Priority), Max_Queue_Length (Can be a Parameter), Number_of_Queues (Can be a Parameter), Time_Field (A_Time) <p>Access a shared System Resource from multiple concurrent flows</p> <ul style="list-style-type: none"> Examples: (a) Model the Y-chart concept of separating the behavior and architecture. (b) Processor, shared robot on a shop-floor. Concept: (a) Modeling of Hardware Resources such as Processor, Cache, Bus, DRAM, and Accelerators. (b) Consumes time period as defined in the behavior flow. (c) Select scheduling from FCFS, Round-Robin, and pre-emption. (d) Simplify the flows without having multiple flows connected with wires to a single Server. (e) Support preemption across request from multiple flows. Blocks: SystemResource Input and Output: Block receives and sends completed transactions virtually from or to the Mapper blocks.

Mirabilis Design Inc. Quick Reference Card

- Operation: (a) Arriving transaction are stored in the Queue. (b) Head of the queue gets delayed by the A_Time. (c) Return to Mapper. (d) Time_Type is set to “Relative_Time”, then A_Time is seconds. Time_Type is set to Number_Clocks, then the A_Time is number of clocks and is multiple by Clock_Speed.
- Key Parameters
 - SchedulerName: Name referenced by Mappers
 - Clock_Rate_Mhz (Can be a Model Parameter)
 - Time_Type (Relative Time or Number_Clocks)
 - Max_Scheduler_Length (Can be a model parameter)

Extend SystemResource timing with more architecture details

- Examples: (a) Cache and memory hierarchy. (b) Separate the behavior flow or data flow from the architecture definition. Use as a processing resource when processing time is unknown and the processing has dependency on some other processing. For example, usage can be a processor: as processor requests memory devices for accessing the data, time taken for memory device supply data is unknown.
- Concept: SystemResource is the Processor and Bus/Cache/Memory can be attached to the output.
- Blocks: SystemResource_Extend and SystemResource_Done (terminate a task).
- Input and Output: Block receives transaction virtually from Mappers, delays internally, and then sends to output port. SystemResource_Done receives the transaction and returns to the Mapper to complete the architecture operation.
- Operation: (a) Same operation as SystemResource. (b) Difference is that the delayed transaction is sent to output port for executing additional blocks. Return to Mapper when encounters Done.
- Key Parameters: Same as SystemResource.
- Choose nonblocking_FCFs type if none of the packets have to wait for it's chance
- For plotting power consumed by this block, use the following format in power table:
Scheduler_(SystemResource_Extend_Name)

Power

- Examples: Generators, Battery, hardware systems that consume power, power management algorithms

- Concept: (a) Used to understand all the parts of the power system for non-functional analysis. (b) The focus is on consumption, sizing the devices, losses, battery life, and so on. This does not focus on the change in Voltage or current. (c) LDO Efficiency is a variable that modifies the amount of power consumed by a device in a state. (d) Determine minimum power needed to run the system. (e) Provide input to mechanical team for cooling.
- Blocks: PowerTable, Battery, and Energy Harvesters.
- Input and Output:
 - PowerTable comprises the following three output ports: (a) Instant power output sends out the instantaneous power consumption of all devices at this level and the levels below in the tree. (b) Average power output sends the average power consumed by the devices over time. (c) State change in comma-separated list of time, hierarchy name, device name, the new state and the power consumed.
- Battery comprises the following ports: (a) From_powerTable – Receives the current load from the PowerTable. (b) Battery_Charge_Input – Receives the charge from the charging source. (c) Battery_External_Aging – Percentage of battery loss as a double. Represents external factors – shock and dropping battery. Reduces battery life by percentage. (d) Available_Energy_Capacity(watt-Hr) provides energy available in the battery. (e) Battery_Life_Remaining(%) – Output battery charge remaining. (f) stats – outputs initial and final states, and warnings.
- Key parameters
 - PowerTable
 - Power_Manager_Table: Define the list of states for all devices, special states (Existing, On and Off), and transition time between states. Define any parameter that is required – can reference a variable. State – Can be any list. Active state and standby states must be defined in the On and Off state columns. Existing- Initial state.
 - Delay_to_Change_State – Time in a state before transition to lower power state.

- ExpressionList – Create variables that can be used for the state power and transition.
 - Battery: (a) Battery type (b) Charging mechanism – standard, threshold before charging starts, turbo charging.
- Generator: Define attributes of the charger such as Sun activity, motor rotations, and so on.
- Temperature and Heat: Plot the heat dissipation and temperature of the entire system while each device consumes power. Also provides the max and average of temperature and heat of each device in the system.
 - Input and output:
 - Power_input: power consumption from the power table's instant_pwr_out port
 - State_Change_input: state change of each device from Power table's state_change port
 - Temp_output: temperature value
 - Heat_output: heat dissipation value
 - Device_output: max and mean values of each device at the end of simulation.

Statistics

- Two types: (a) Statistics block: Input values are aggregated to generate statistics on demand. (b) Reports with list of pre-defined statistics for Resources and hardware.
- List of Outputs Available: (a) Utilizations (Min, Max, Mean, and Standard Deviation). (b) Occupancy (Min, Max, Mean, and Standard Deviation). (c) Number of Transactions Entered, Exited, and Rejected. (d) Instantaneous and Average Power consumption for system, subsystem, and components. (e) Battery Life in Percentage, Available Charge, and Discharge.
- Custom Output: Compute latency and throughput in ExpressionList and send to Plotter.
- Saving Statistics: Statistics can be saved as text, csv, and .plt (Plot format) files located in any directory. The Digital simulator supports to store the statistics in a file by enabling writeStatsToFile parameter. The statistics from Architecture Setup will be saved in a file and placed in the Result folder which will be in the model location.
- Interpretation: (a) Statistics like utilization, Queue length vs throughput, (b) Data size vs latency, instantaneous power consumption, and so on are used to decide the efficiency, resource usage,

- and scalability. (c) Example utilization of a resource above 90% suggests that the resource is being over utilized and the user must increase the processing speed or reduce the workload arrival time. (d) Buffer occupancy is the minimum queue depth required to prevent data loss.
- Bottleneck or Problem: (a) Bottlenecks are identified by continuous increase in latency. (b) Must check buffer usage and utilization for all resources in the flow to determine actual bottleneck.

Glossary

- Modeling Concept: Type of modeling approach.
- Simulator: Model of computation that determines the interaction between blocks.
- Discrete-event or Digital: Sequence of synchronous (same-time) events executing between asynchronous (different but not periodic) time.
- Behavior: Sequence of tasks that describe a process or application. Focused on the flows and the delay to complete a task. Does not include the functional details like the math and algorithms.
- Architecture: Platform to execute the application. Can contain hardware, network, middleware and RTOS.
- Timing: Measured in seconds.
- Data: Size is always in bytes.
- Power: Consumption of energy by a device.
- Functionality: Behavior of a certain sequence of tasks.
- Stochastic: Using random values to define the time and other attributes.
- Process: Sequence of tasks. Typically for a behavior flow. Can be implemented in hardware or software
- Task: Part of a behavior flow. Defines a specific action.
- Thread: A behavior flow that executes on a single processor.
- Traffic: Start of an activity.
- Traffic attributes: Details of the traffic.
- Workload: Alternate term of traffic.
- Statistics: Recording of the model activities and interpreting them in a decision form.
- Plotting: Display the statistics.
 - Latency can be end to end latency of the system or latency across a particular block.
 - Throughput can be extracted by the specific field of

Mirabilis Design Inc. Quick Reference Card

<p>the data structure coming out of the block.</p> <p>Key Blocks</p> <p>Traffic: Outputs a new Data Structure (DS) at time intervals specified by the "Time_Distribution" setting. Input/Output: Output is a transaction. Can also be called a packet or data request. Key Traffic block Parameters: (a)Data_Structure_Name: Set to "Header" or "Processor_DS". (b)Start_Time: Offset for the first transaction. (c)Value_1 and Value_2 are used in the distributions. (d) Distributions: single transaction, transactions at fixed interval (Value_1), exponential distribution, uniform between Value1 and Value2, and normal with mean Value_1 and standard deviation of Value_2. (e) FileOrURL: If we want to read the traffic from file , then provide the address of the file here. (f) Random_Seed: the value defined by default is 123457L. Used to determine the beginning random sequence of data structures generated. (g) Number_of_Transactions: Specifies the max number allowed and by default it is MaxInt whose value is 2147483647</p>	<p>Input/Output: (a) Can add as many ports as required for connection by right-clicking and selecting "Configure Ports". Note that Type need not be defined. (b) Ports receive Data Structure or a value of any data type (Polymorphic). (c) With multiple input port, block executes the expression after all ports receive data. (d) In the expression, the data on a port is identified by port name + "." + field name. (e) Number and order of items in Output_Values, Output_Ports, and Output_Condition must match. (f) All output ports must be listed. (g) Output_Condition is a Boolean that determines if data is to be sent on a particular port.</p> <p>Fork and Join: (a) Fork: Outputs a single transaction into two output transactions. (b) Join: Combines two incoming transactions into one flow. Input/Output: Transactions. Note: There is no delay between input and output, and between output ports. Fork output is first the top port and then the lower. Join sends out the transaction in the order received.</p>	<p>received at Pop. (c) Used when the processing delay is not known in advance. Input/Output: (a) Input: Transactions (b) Pop_Input: Integer or array (Queue, position) (c) Output: Transactions, (d) reject_output: Transactions Key Data Structure fields</p> <ul style="list-style-type: none"> • Priority_Field, Queue_Number_Field <p>Key Queue block Parameters</p> <ul style="list-style-type: none"> • Max_Queue_Length: The maximum queue length for each queue (can be a parameter). • Number_of_Queue: Number of independent queues (can be a parameter). • Queue_Reject_Mechanism : Incoming Token rejected or • Queue_Type: choose between FIFO or LIFO • Initial_Queue_State: Refers to the first transaction when queue is empty – output if none are waiting or wait until a pop has been received. <p>Reports: Number_Entered, Number_Exited, Number_Rejected, Occupancy, Utilization, Delay</p>	<p>block. Used by Mappers, RegEx function, and other SystemResource block to call this block.</p> <ul style="list-style-type: none"> • Scheduler_Type: FCFS, Preemptive or Round-Robin. • Maximum_Scheduler_Length: Queue length <p>Reports: Number_Entered, Number_Exited, Number_Rejected, Occupancy, Utilization, Delay</p> <hr/> <p>SystemResource_Extend</p> <ul style="list-style-type: none"> • Same as SystemResource. • No support for Pre-emption. • Cannot call another SystemResource. • Can add more processing activity at the output port. • Return to Mapper when output flow encounters a SystemResource_Done. <p>Input/Output:</p> <ul style="list-style-type: none"> • Transactions, Task_Plot <p>Key SystemResource block Parameters:</p> <ul style="list-style-type: none"> • Resource_Name: Name of this SystemResource block. Used by Mappers, RegEx function, and other SystemResource block to call this block. • Scheduler_Type: FCFS or Round-Robin or non-blocking_FCFS • Maximum_Scheduler_Length: Queue length <p>Reports: Number_Entered, Number_Exited, Number_Rejected, Occupancy, Utilization, Delay</p>
<p>TrafficReader: Import trace files from network sniffer or hardware capture. Input/Output: Output the next row as a data structure for every input.</p> <p>Database: Lookup table containing rows and columns. Each row is a data structure and the field names are the columns. Input/Output: Transactions Key Database block Parameters</p> <ul style="list-style-type: none"> • Linking_Name: Name to link multiple Database blocks using the same table. • fileOrURL: file name containing the table (.txt, .csv, .xml). • Data_Structure_Text: Table values or 'extern' - to reference another database block. • Input_Fields: List of incoming data structure fields for lookup. • Lookup_Fields: List of column names to match with Input_Fields in same order. • Output_Expression: Specifies the match type and the value to be placed on the output port. • Mode: Search (Read), Update (Write), or delete (Remove). 	<p>SoftwareMapper and Dynamic Mapper: (a) Sends the transaction to the Processor (DynamicMapper only), SystemResource or SystemResource_Extend. (b) Mapper block immediately sends the transaction to the Resource. Input/Output: Transactions Key Data Structure fields: (a) A_Time, A_Priority, A_Task_ID for SystemResource, (b) Task_Instruction, Task_Priority, Task_ID, Task_Name, Task_Destination for Processor Key Mapper block Parameters</p> <ul style="list-style-type: none"> • Target_Resource: Identifies the SystemResource. • Task_Number: The Task_Number is the position on the Y-axis. • Task_Priority: Used to reorder the input queue and for preempting the current task. • Task_Time: Processing time at the SystemResource. • Mutual Exclusion enabled (SoftwareMapper): Transaction cannot be pre-empted. • QueueTaskNow: SoftwareMapper can be queued locally until Resource or available or sent immediately. • Database_Lookup (DynamicMapper): Database Name that contains the attribute values. Matched by the Task_Name. 	<p>Server: (a) Combines a queue and a processing time delay. (b) Used when the processing time is known in advance. If the transaction can be preempted, use SystemResource. Input/Output: Transactions Key Data Structure fields Queue_Number_Field (A_Task_Address), Priority (A_Priority), Max_Queue_Length (Can be a Parameter), Number_of_Queues (Can be a Parameter), Time_Field (A_Time) Key Server block Parameters</p> <ul style="list-style-type: none"> • Number_of_Queues: Number of independent queues in this block. • Queue_Type: choose between FIFO, LIFO or SLOT • Max_Queue_Length: Maximum queue length for each Queue. • Time_Field: Predetermined delay time, can be a field or double value. <p>Reports: Number_Entered, Number_Exited, Number_Rejected, Occupancy, Utilization, Delay</p>	<hr/> <p>PowerTable: Study and model the power infrastructure, Determine the consumption by operations on resources, and Design the best power management algorithm. Input/Output: Output: Instantaneous Power output, Average Power output, Device State change. Key PowerTable block Parameters</p> <ul style="list-style-type: none"> • Manager_Setup: Contains the list of states and associated power levels, reference to the Active (On) and Standby (Off), transition time from one state to another and parameters. Delay_to_Change_State: Time in a state before changing to another state. • Expression_List: Computes expression for variables that are used in the Manager Setup. (a) stateChange- RegEx function to change the state in ExpressionList or Script. (b) AsyncStateChange can be define externally using an FSM, or Verilog/C/Script blocks.
<p>ExpressionList: Execute a sequence of expression in-order. Can be used to update field, compute statistics, and make decisions.</p>	<p>Queue: (a) Defines a buffer or a FIFO/LIFO in the flow. (b) Removes the first transaction when trigger is</p>	<p>SystemResource: (a) Used when multiple flows need to access a single resource. (b) Used when pre-emption is required. (c) Model stop execution if queue overflows (error will be thrown). Input/Output: Transactions, Task_Plot Key SystemResource block Parameters:</p> <ul style="list-style-type: none"> • Resource_Name: Name of this SystemResource 	<p>Reports: Cumulative power consumption per device, Cumulative power consumption per device at a particular state, Cumulative power consumption of all devices,</p>

Mirabilis Design Inc. Quick Reference Card

Average power consumption of all devices, State change details of a device.

Script: (a) Implements the VisualSim Script language. This language combines standard programming constructs with the RegEx functions and is fully integrated with the graphical editor. (b) Supports blocking and non-blocking wait statements, ability to create or wait for events, and has multiple threads defined and call between threads and modules. (c) Supports the following Methods: (a)WAIT(<time> or <event> or <clock rate Mhz>): A delay that holds the Script from operating on any data structure until the time has expired. (b)SEND(<port name> or <label name> or <block name>): Send to a port, virtual connection, another Script or to a LABEL. (c)QUEUE(<queue name>, <token>, <priority>, <queue operation>): Stores and removes the token in the Queue in a FIFO method. The queue is reordered based on the priority of the incoming token.(d)TIMEQ (<queue name>, <token>, <priority>, <queue operation>, <delay expression> or <clock_hertz>): Used to define, put, pop time, and do processing in a Timed Queue. (e)EVENT(<Event Name>,) or newEvent(<event name>) or <event name>.event(): Creates a new Event. (f) PLOT(<plot_name>,<destination>,<plot_color>, <plot_offset>,<plot_value 1.0 or 0.0>): Plots timing diagrams or latency or resource activity. (g) CLOCK("MyEvent") : This is an added variation of EVENT that selectively fires a WAIT ("MyEvent") or TIMEQ ("MyEvent"). This acts as a virtual clock.Input/Output: Transactions , values of any data type
Key Script block Parameters: SelfStart - Create a new parameter called SelfStart and set the value to true to start execution of script without a need for trigger.
Reports: Statistics for internal queue, generate plot.

Key Plotters

TextDisplay

- Display the values arriving on the input port in a text display dialog
- Ports – input – Default input port
- TimeDataPlotter
- Depicts latency, throughput, and other variables that vary against time
- Ports – input – accepts multiple datasets
- HistogramPlotter
- Plots a histogram using the input data

- Ports – input – A multiport which means that each port or relation can be connected directly to this port and is treated as a unique dataset.
- Statistics
- Compiles the statistics for a sequence of scalar values- integer, double or long.
- Ports
 - stats_data - An input port for the data samples.
 - stats_trigger - An input port that triggers the current statistics to be placed on the output port.
 - stats_reset – An input port that resets all internal statistics.
 - histo_output - The values can be plotted on a histogram Plotter block in VisualSim.
 - output - Outputs the current Statistics when trigger on the stats_trigger port.
- ResourceStatistics
- Generates and resets statistics for one or many Schedulers and Smart Resources in the model.
- Ports - Stats_Out - Output a data structure for a single queue statistics.
- Resource_List : Specifies the Names of Resources
- ResourceLength_List : Length of all resources in the Resource List
- Number_of_Samples : defines how many stats output has to be done within the simulation time
- Statistics : Boolean field which when enabled generates the statistics and resets when disabled

Key Hardware Blocks

Note:

- Architecture setup is required for all hardware modelling.
- Must use Processor_DS as the Data Structure.
- All hardware blocks have the data structure as the input and output.
- Listen to block on Architecture Setup by selecting the listen to option. This provides details about the block activity.
- Customizable hardware blocks have ports at the bottom for debug messages and statistics.
- Standard block statistics are output at Architecture Setup.
- Buses use multiport. So, input and output of device, bridge and other buses must be connected to the same port.
- CycleAccurateCache, CycleAccurateDRAM, Interfaces and Buses have debug and statistics ports.

- Basic required fields common to all blocks - A_Source (starting device), A_Destination (final point such as a memory or display or HDMI), A_Command (action to be taken- Read, Write, Erase, Prefetch, and so on).
- Each hardware block must have a unique name.

ArchitectureSetup: Handles all the address mapping, routing, debug messages, plotting and statistics generation. Ports:

- plots_out - Generates the values for each Statistics_to_Plot name as a separate dataset.
- internal_stats_out - Generates all the general statistics for the number of times specified in the Number_of_Samples.
- util_stats_out – Generates all the utilization statistics for the number of times specified in the Number_of_Samples.

Error: Invalid architecture block name used, routing table conflicts, field name mapping conflict.

Demo:VS_AR\doc\Training_Material\Architecture\Setup\Arch_Setup.xml

DeviceInterface: Enables users to connect custom blocks to the bus, not needed for standard hw blocks.

Ports:

- input, output – Connected to device or logic blocks
- fm_bus, to_bus – Input connected to the bus port, Output connected to the bus

TimingDiagram: Generates timing diagram for key arch blocks. Connects output port to timed plotter.

Processor: Models commercial and proprietary processors. Ports:

- instr_in, instr_out – Connects to any VisualSim library block
- bus_in, bus_out – Bus input and output ports
- bus_in2, bus_out2 – Bus input and output ports
- reject_out – When the instruction queue is full, the incoming data structure is not executed and placed on this port

Key Data Structure fields:

A_Source, A_Destination, A_Variables, A_Hop, A_Instruction and A_Priority

Dependencies: Instruction_Set

Error: Missing Instruction_Set or an instruction, processor clock speed less than cache speed, pipeline stalled for than 5000 cycles for response from external data request

Demo:VS_AR\doc\Training_Material\Architecture\Setup\Timing_Diagram.xml

Instruction_Set: Lists the instruction for each execution unit of the processor.

Error: Missing semicolon at the end of a line, columns are missing, instruction not found
Demo:VS_AR\doc\Training_Material\Architecture\Setup\Instr_Set.xml

Integrated_Cache:

The cache can be used as a stochastic cache block or cycle accurate cache block by a parameter that the user can select in drop-down menu(Stochastic_or_Address_Based). Ports:

- fm_cache, to_cache – Connects to bus or lower-level cache closer to processor
 - miss_in, miss_out – Connects to bus or next level cache that is closer to memory
 - stats – Sends out debug messages and statistics
- Key Data Structure fields: A_Source, A_Command, A_Bytes, A_Address
Error: Invalid higher memory name

Coherence Cache: Enables multi-core architecture to access shared memory with data consistency. Support MESI coherence protocol in both snooping and directory-based architecture.

Ports:

- input, output – Connects to processor or bus which connects lower level caches.
- To_Bus, Fm_Bus – Connects to bus or interconnect that allows data transfer to higher level cache or main memory.

Key Data Structure fields: A_Source, A_Command, A_Bytes, A_Address

Snooping protocol connects to coherent buses such as ACE. Directory based protocol connects to NoC.

TaskGenerator: Generates profile based sequence of instructions to emulate the software task on the processor.

Ports:

- port- receives data structure with A_Task_Name containing the name of the task
- port2 – outputs ds with updated instruction field
- debug – Outputs status and debug information

Key Data Structure fields:

A_TaskName, A_Instruction

Key Parameters:

Block_Name, My_Path, mode of operation and Instruction_Mix_File

Mirabilis Design Inc. Quick Reference Card

<p>RAM: Combines the memory controller and memory array to model SRAM, DRAM, Flash, and ROM</p> <p>Ports:</p> <ul style="list-style-type: none"> input, output – Connects to a bus input2, output2 – Connects to a bus output3- ds with write operation and A_Task_Field is false is sent here <p>Key Data Structure fields: A_Source, A_Destination, A_Command, A_Bytes, A_Bytes_Remaining and A_Bytes_Sent</p> <p>Error: Invalid parameters, invalid access time entry.</p>	<ul style="list-style-type: none"> port - outputs the debug messages <p>Key Data Structure fields: A_Address_Col, A_Address_Row , A_Command, A_Bytes</p> <p>Dependency: Connected to Memory_Controller</p>	<p>Bridge: Connects two buses</p> <p>Ports:</p> <ul style="list-style-type: none"> port, port2 – Connects to one bus port3, port4 – Connects to another bus status – receives message about bridge operation <p>Key Data Structure fields: A_Source, A_Destination</p> <p>Key Parameters: Bridge_Speed_in_Mhz, Bridge_Width_in_Bytes, Overhead_Cycles</p>	<ul style="list-style-type: none"> stats_out – Connects to a text display to observe the statistics of TileLink plot – connects to a Time Data Plotter to display activity diagram <p>Key Data Structure fields: A_Command, A_Source, A_Destination, A_Bytes</p> <p>Key Parameters: Speed, Bus_Width</p> <p>Dependency: TileLink Client, TileLink Manager</p>
<p>Memory_Controller: Emulates the memory controller at cycle accurate level.</p> <p>Standard: EDEC/JESD Standard</p> <p>Ports:</p> <ul style="list-style-type: none"> rd_wr_data_fm_bus, rd_wr_data_resp_to_bus - receives the read request or writes data from the bus port ,sends the read data out and the acknowledgement out for write through this port to the bus rd_data_fm_mem, rd_wr_data_to_mem - receives the read data returned from the DRAM block and sends out the transaction to the DRAM ctrl_fm_mem, ctrl_to_mem - sends and receives control signals such as tRAS, tRP, tRCD to memory or from the memory status - outputs the debugging messages when the DEBUG parameter is true <p>Key Data Structure fields: A_Address_Col, A_Address_Row, A_Bytes_Total, A_Mem_ID</p> <p>Key Parameters: DRAM_Type, Controller_Speed_Mhz, Mfg_Suggest_Timing, Extra_Timing, Memory_Width_Bytes , Command_Buffer_Length, Burst_Length</p> <p>Error: invalid DRAM type</p>	<p>BusArbiter: Provides arbitration for bus and combines with multiple instances of the BusInterface to create a Shared Bus topology.</p> <p>Ports:</p> <ul style="list-style-type: none"> input - Connected to the top port of the BusInterface input1, output1 - Defines the custom logic when Arbitration Mode is set to Custom <p>Key Data Structure fields: A_Source, A_Destination, A_Command, A_Task_Flag, A_Bytes, A_Bytes_Remaining</p> <p>Key Parameters: Bus_Speed_Mhz, Bus_Size_Bytes, Width_Bytes, Arbiter_Mode</p> <p>Dependency: BusInterface</p>	<p>AFDXSwitch:</p> <p>Standard: ARINC Specification 664 Part 7, a profiled version of an IEEE 802.3 network per parts 1 & 2</p> <p>Ports: Debug</p> <p>Key Data Structure fields: Task_Source, Task_Destination, Task_Type, Task_Size</p> <p>Dependencies: AFDXNode, AFDXTraffic, and AFDXConfig blocks</p>	<p>NoC: It is a scalable interconnect for an SoC to enable data transfer between multiple devices and the memory.</p> <p>Ports:</p> <ul style="list-style-type: none"> Master NIU: Connects master devices to the router <ul style="list-style-type: none"> Device_In, Device_Out – Receives and Sends transactions to Device. NW_Out, NW_In – Send and receive packets from router. Slave NIU: Connects Slave devices to the router <ul style="list-style-type: none"> Device_In, Device_Out – Receives and Sends transactions to Device. NW_Out, NW_In – Send and receive packets from router. Router: transfer data to other routers or devices. <ul style="list-style-type: none"> Device_Port_In, Device_Port_Out – connects to master or slave NIU. Device2_Port_In, Device2_Port_Out – connects to master or slave NIU. North_In, North_Out, South_In, South_Out, East_In, East_Out, West_In, West_Out – directional ports to connect an adjacent router. Wire: Connects two router ports. <ul style="list-style-type: none"> Delay_In, Delay_Out – forward the data from input to output. <p>Key Data Structure fields: A_Command, A_Destination, A_Bytes</p> <p>Key Parameters: Flit_Size_Bytes, Buffer Size, QoS</p> <p>Dependency: Master NIU, Slave NIU, Wire, Router</p>
<p>CycleAccurateDRAM: Captures the functionality and accurate timing of many variations of DRAM.</p> <p>Ports:</p> <ul style="list-style-type: none"> port_1, port_2 - receives the read request or write data from the memory controller and sends out data to memory controller fm_ctrl, to_ctrl - receives the RAS, RP, RCD commands from the memory controller and sends the RAS, RP, RCD response commands back to the memory controller port_4, port_3 - connected to external logic for reading data from an address location and making a read request or write data to an address location 	<p>BusInterface: Connects devices to the bus</p> <p>Ports:</p> <ul style="list-style-type: none"> input1, output1: Connects to a device/bus - can be Master or slave input2, output2: Connects to a device/bus- can be master or slave child_in, child_out – Provides the connection to other businterfaces and the BusArbiter. <p>Key Data Structure fields: A_Source, A_Hop, A_Destination, A_Bytes, A_Bytes_Remaining, A_Bytes_Sent, A_Command</p> <p>Dependency: BusArbiter</p> <p>Error: invalid bus name, similar bus port names</p>	<p>AMBA_AHB: High performance buses for interconnecting peripheral IP to any independent processor/memory subsystems within a SoC.</p> <p>Ports:</p> <p>Port1,port3,port5,port7- connects to master Port2,port4,port6,port8- connects to slave</p> <p>Key Data Structure fields: A_Command, A_Source, A_Destination, A_Hop, A_Bytes, A_Bytes_Sent, A_Bytes_Remaining, A_Task_Flag, A_Prefetch, A_Interrupt</p>	<p>AVB: Designs a completely new AVB-based network to integrate all the equipment, upgrade existing networks, and design the electronics that are used in such networks.</p> <p>Standard: IEEE 802.1BA</p> <ul style="list-style-type: none"> StreamRP- <ul style="list-style-type: none"> TG_in, TG_rtn – Receives traffic, Sends out transactions strm_rtn, strm_out – Receives returned transactions, Sends out traffic AVB_Node
	<p>DMA:</p> <p>Ports:</p> <ul style="list-style-type: none"> Req, Dout – Receives transactions, Sends out transactions, Ack, Din – Sends dma requests to the device, Connects to bus. reject – Sends out transactions when the channel buffer overflows <p>Key Data Structure fields: (with database dependency) A_Task_Name, A_Instruction</p> <p>Key Data Structure fields: (without database) A_DMA_Command, A_DMA_Destination, A_Priority, A_DMA_Bytes, A_DMA_Channel, A_DMA_Burst_Bytes.</p>	<p>AMBA_AXI: Advanced high performance on chip interconnect protocol to enable data transfer between SoC devices.</p> <p>Ports:</p> <ul style="list-style-type: none"> All input ports (left side) are connected to masters All output ports (right side) are connected to slaves When using AXI bus, make sure to give the value for the parameter Threshold_Trans_T_Bytes_F as true When using AXI bus, generate traffics with an offset stats_out – Outputs the statistics <p>Key Data Structure fields: A_Command, A_Source, A_Destination, A_Bytes, A_Priority</p>	
		<p>TileLink: It is a chip-scale cache coherent interconnect standard. It enables coherent memory access in an SoC which contains multi core processor, accelerators, DMA and IO devices.</p> <p>Ports:</p> <ul style="list-style-type: none"> All Multi ports (left side) are connected to TileLink Client All Multi ports (Right side) are connected to TileLink Manager 	

Mirabilis Design Inc. Quick Reference Card

<ul style="list-style-type: none"> o port_in, port_out – Receives transactions, Sends transactions • AVB_Traffic <ul style="list-style-type: none"> o net_tg_in, net_tg_out – Receives traffic, Sends out generated traffic o data_out – Sends out latency of the transaction • AVB_Stats <ul style="list-style-type: none"> o stats_in – Receives transactions <p>Dependencies : AVB Traffic, AVB SRP, AVB Node, AVB Bridge. Network Setup, Config table, AVB Statistics</p> <p>CAN: Based on the Bosch specification and the international standard defined in the ISO 11898-1 and combines both the standards in a single block.</p> <p>Standard: ISO 11898-1</p> <p>Ports:</p> <ul style="list-style-type: none"> • CAN_Node <ul style="list-style-type: none"> o node_in, node_out – Receives messages, sends messages o rx,tx – Connects to CAN_Bus wire port • CAN_Bus <ul style="list-style-type: none"> o wire – Helps in connections 	<ul style="list-style-type: none"> • stats_msg_out – Sends out Statistics (Data Structure) information, Debugging (String) Information. <p>Key Parameters: Number_of_Lanes, Max_Payload_Size, Max_Payload_Req_Size</p> <p>UCle: It is a die to die interconnect to integrate various chiplets. It provides high bandwidth with low power and low latency data transfer between chiplets.</p> <p>Ports:</p> <ul style="list-style-type: none"> • multi-ports (input and output) connected to a chiplet. • Debug_port – sends debug messages about UCle processing. <p>Key Data Structure fields: A_Command, A_Source, A_Destination, A_Bytes</p> <p>Key Parameters: Max_Link_Speed_GTPs, Max_Read_Req_Size_Bytes, Buffer_Size_Bytes</p> <p>PCI_RAD: It is an arbitration based bus which is used for interconnecting peripheral chips to any independent processor/memory subsystems</p> <p>Key features:</p> <ul style="list-style-type: none"> (a)Implements arbitration algorithm according to which masters are given access to the bus (b)Terminate/Wait: It is possible to terminate the current access by a master or make that master wait for a specified cycles (e)Provides block debug feature 	<p>the standards which define mechanisms for the time-sensitive transmission of data over Ethernet network. Has multiple gateways through which sensors, Ethernet traffic etc. are connected</p> <p>Key features:</p> <ul style="list-style-type: none"> (a)IEEE 802.1Qbv (b)IEEE 802.1Qbu (c)IEEE 802.3br (d)IEEE 802.1Qca (e)IEEE 802.1Qcc (f)IEEE 802.1Qci (g)IEEE 802.1Qch (h)IEEE 802.1AS 	<p>node (existing), adding a link, removing a node, or removing a link.</p> <ul style="list-style-type: none"> o stats_input, stats_output - Generates statistics reports for the referenced routing table, driven by the type of 'stats_input' <ul style="list-style-type: none"> • Routing_Table <ul style="list-style-type: none"> o No ports • Multicast <ul style="list-style-type: none"> o fr_layer, to_layer - From the Application layer, To the Application layer o fr_Node, to_Node – From the Node block, To the Node block • Switch_Four_X_One <ul style="list-style-type: none"> o zero_input - input port Zero for data tokens o one_input - input port One for data tokens o two_input - input port Two for data tokens o three_input - input port Three for data tokens o control - Input port for control tokens, which selects the output based on the matching with the relative switch address. o no_output - If the control port value is out of the Switch_Address range, then the incoming token is placed on this port. o switch_output - If the control port value is out of the Switch_Address range, then the incoming token is placed on this port.
<p>Fibre_Channel:</p> <p>Ports:</p> <ul style="list-style-type: none"> • FC_N_Node: <ul style="list-style-type: none"> o Device_In, Device_Out – Receives and Sends transactions to Device o frm_switch, to_switch – Receives and Sends transactions to switch o Debug – Enable debug and connect text display to capture debug information • FC_96_Nodes • FC_Link • FC_Switch • FC_Config • FC_Traffic <ul style="list-style-type: none"> o net_tg_in, net_tg_out o data_out • ASM_Traffic <ul style="list-style-type: none"> o net_tg_in, net_tg_out o data_out <p>Dependencies: ConfigTable, DeviceInterface block</p>	<p>TimeTriggeredEthernet: Provides the capability for deterministic, synchronous, and congestion-free communication, unaffected by any asynchronous Ethernet traffic load.</p> <p>Standard: IEEE 802.3</p> <p>Ports:</p> <ul style="list-style-type: none"> • TTE_Traffic <ul style="list-style-type: none"> o net_tg_in, net_tg_out – Receives and sends transactions o data_out – Outputs the traffic streams • TTE_Node <ul style="list-style-type: none"> o port_in, port_out – Input and output interfaces to the upper layers of the protocol stack • TTE_Stats <ul style="list-style-type: none"> o stats_in – Input from the Node or TrafficGenerator o Error: Multiple RiO_Node, Serial switch blocks with similar name, error due to missing architecture setup 	<p>Networking:</p> <p>Ports:</p> <ul style="list-style-type: none"> • Ethernet_Traffic <ul style="list-style-type: none"> o net_tg_in, net_tg_out - Returns the AVB stream status to the block, all the stream traffic will be output at this port o data_out - outputs the traffic streams that treat this block as the Listener • Layer_Protocol <ul style="list-style-type: none"> o ds_up_input, ds_up_output - Input port for connection from upper Layer_Protocol, Output port for connection from this block to the upper Layer_Protocol o ds_dn_input, ds_dn_output - Input port for connection from upper Layer_Protocol, Output port for connection to lower Layer_Protocol or Node. o up_ext_output, dn_ext_output - Output port for external processing in the Up direction, Output port for external processing in the Down direction. • Layer_Complete <ul style="list-style-type: none"> o input - Input port to Layer_Complete from Protocol_Layer block. • Layer_Table <ul style="list-style-type: none"> o No ports • NODE <ul style="list-style-type: none"> o route_input, route_output – Receives and sends packets o node_input, node_output – Receives and sends packets • NODE_Master <ul style="list-style-type: none"> o request_input - updating the referenced routing table in terms of recalculating the routing table, adding a 	
<p>PCle_Bus: Provides a scalable, high-speed, serial I/O bus that maintains backward compatibility with PCI applications and drivers.</p> <p>Ports:</p> <ul style="list-style-type: none"> • All input ports (left side) are connected to masters • All output ports (right side) are connected to slaves 	<p>TSN: Implements IEEE standard called IEEE 802.1Q Time Sensitive Networking</p> <p>TSN is a set of standards defined by Time-Sensitive Networking task group of the IEEE 802.1 working group,</p>		