Getting Started

MIRABILIS DESIGN

1159 Sonora Ct,

Suite 116, Sunnyvale,

CA 94086, USA

Mirabilis Design Inc.
1159 Sonora Ct, Suite 116
Sunnyvale, CA 94086

Revision History

The following table shows the revision history for this document

Date	Version	Revision
March 27, 2013	13	1
Feb 01, 2014	14	1
June 16 2016	16	2
February 10, 2017	17	1
October 31 2019	19	1

Contents

1. About Mirabilis Design and VisualSim

Mirabilis Design is a leading provider of System-Level Architecture Exploration software for designing electronics and real-time software. Using VisualSim, designers can architect the "right" product, i.e. one which minimizes product failures and has not been over- or under- designed.

Mirabilis Design accelerates Concept Engineering by drastically reducing typical model development from months to days and overall project time by 25-30%. VisualSim enables users to design, analyze and validate new system level concepts; and generate a matching system specification. VisualSim is used for performance, functional and power exploration of network of systems, large systems, subsystems, components (IC, SoC, FPGA and boards) and real-time software. Benefits from the solution are a visual executable specification; easier creation of optimized and differentiated products and; corporate infrastructure enabling extremely fast design trade-offs for price, performance and power.

Model constructed in VisualSim can help you make better design specification decisions. Evaluations of system specification using VisualSim performance, power and architectural models can help eliminate clearly inferior choices, point out major problem areas, and evaluate a variety of cost, performance and partitioning trade-offs. Simulation is cheaper and faster than building hardware prototypes and can also help with software development, debugging, testing, documentation, and maintenance. Furthermore, early partnership with customers using visual prototypes improves feedback on design decisions, reducing time to market and increasing the likelihood of product success.

Modeling and Analysis

Users assemble a model of their proposed or existing system using a series of icons or blocks in a graphical schematic capture environment. These icons or blocks are parameterized modeling components that reduce the learning curve, accelerate model development and enable accurate design optimization. User can import Third-Party IP and custom development in various script and programming languages. Multiple levels of abstraction supported in a single model can include statistical, transaction-level (TLM 2.0) and cycle-accurate. You can develop a detailed specification; and generate the test benches and assertions for verification. The models are simulated using a highly optimized simulator to conduct performance, power and functional trade-offs. Designers and architects can conduct trade-off studies by varying parameter values, running different stimulus and modifying the system configuration including the architecture.

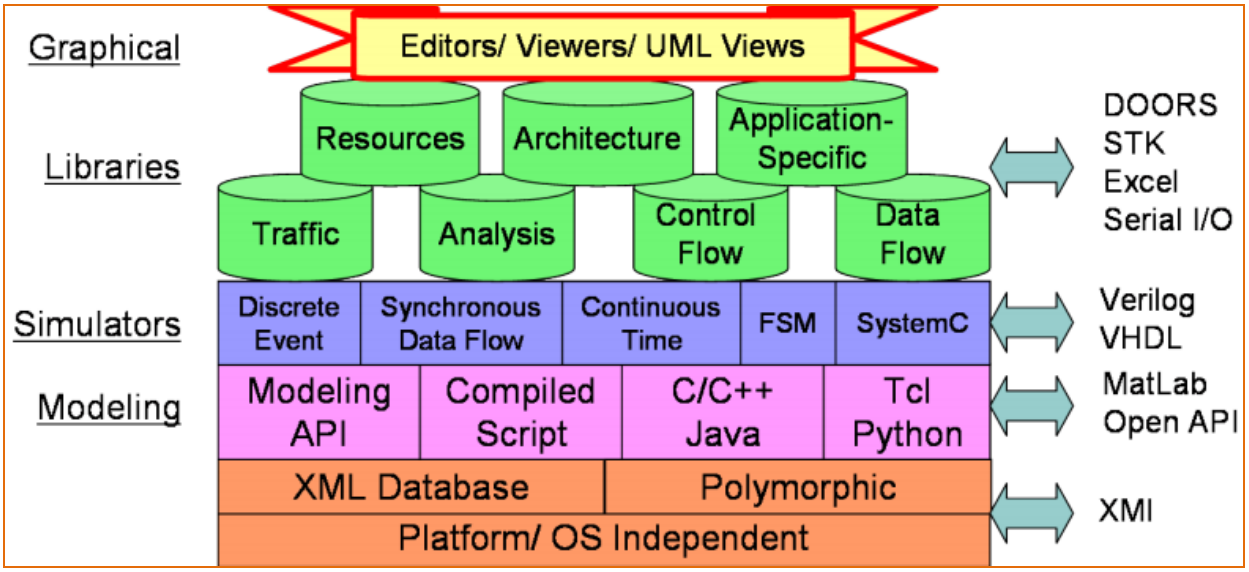


Figure 1: Overview

VisualSim Product Organization

VisualSim provides pre-built components that are graphically instantiated to describe system, environments and output profiles. The traffic is described using transaction generators and the results are created using statistics report writers and plotters. The applications and use cases are described as UML-like block diagrams. The models can be simulated as a traffic flow with delays, or generated by a software execution on target hardware architectures. Traffic profiles, task characteristics and hardware profiles can be varied to cover a large experimentation space. Statistics and reports can be viewed in real-time or written to file for future analysis.

VisualSim is made up of five major products:

- VisualSim License Manager governs the usage of VisualSim products. This is a proprietary network- and client-based license manager. In addition, VisualSim supports FlexLM Server on Linux.
- VisualSim Architect is the model development, simulation and analysis environment. This platform contains simulator, large collection of libraries for model construction, verification and analysis. The parameterized libraries include traffic generators, statistics viewers, behavior definition language, performance resources, accurate hardware components and application-specific algorithms.
- VisualSim Post Processor is a standalone application to conduct port-simulation analysis. This environment can take saved results from multiple runs and combine

them for comparative analysis. This is a separate product and is used for offline review.

- VisualSim Batch Mode Simulation is for execution of the simulation from the command-line without any graphical viewers.
- VisualSim Explorer is a private Web Server that presents the models within Web Browsers for viewing, running simulation and analyzing the results from any accessible computer. The Server is maintained within the company firewall.

VisualSim Architect is composed of the following:

- Core: Simulator, Block Diagram Editor, Basic modeling libraries, XML database and standard interfaces (MatLab, Datagram's, Serial I/O, C/C++/Java, Python and Tcl)
- Add-on Libraries: Modeling IP are provided for a number of tasks and applications. The modeling IP are organized into toolkits or libraries. The use of the libraries requires Architect.
- Utilities: Utilities are tools provided to generate specific analysis. These include model optimizers and power analysis tools.
- Interfaces: Interfaces are provided to run VisualSim in co-simulation with Verilog, SystemC and Satellite Toolkit. The Interfaces require the licenses for the relevant tools that are co-simulated with VisualSim.

VisualSim Post Processor is platform and OS-independent software application, enabling users to focus on the analysis without involving themselves in the modeling details. In addition, a mobile provision enables the Post Processor to be used offline. This enables flexible usage during travel, design review and remote presentations. VisualSim Post Processor can be used to graphically display and analyze performance data collected from the simulation. The Post Processor can organize results into a variety of x-y graphs and histogram plots, and displays them in either graphical format.

VisualSim Batch-mode Simulation enables modelers and verification engineers to conduct batch simulation for exploring combination of multiple model parameters values. The simulation can be executed sequentially on a single machine or run in parallel on a server farm. This approach can also be adopted to run VisualSim with Verilog or software code. This mode requires that the model has no graphical viewers.

VisualSim Explorer enables the user to embed the models within an html page. This page can be viewed from within a Web Browser that has the right permissions. The granting of the permission is the responsibility of the VisualSim user organization. The web user can view the specific hierarchy exposed, and change parameters, run simulation and view results, if

permitted. Use the Export to HTML to generate the files for importing into the Explorer environment.

2. Transform Ideas into Modeling

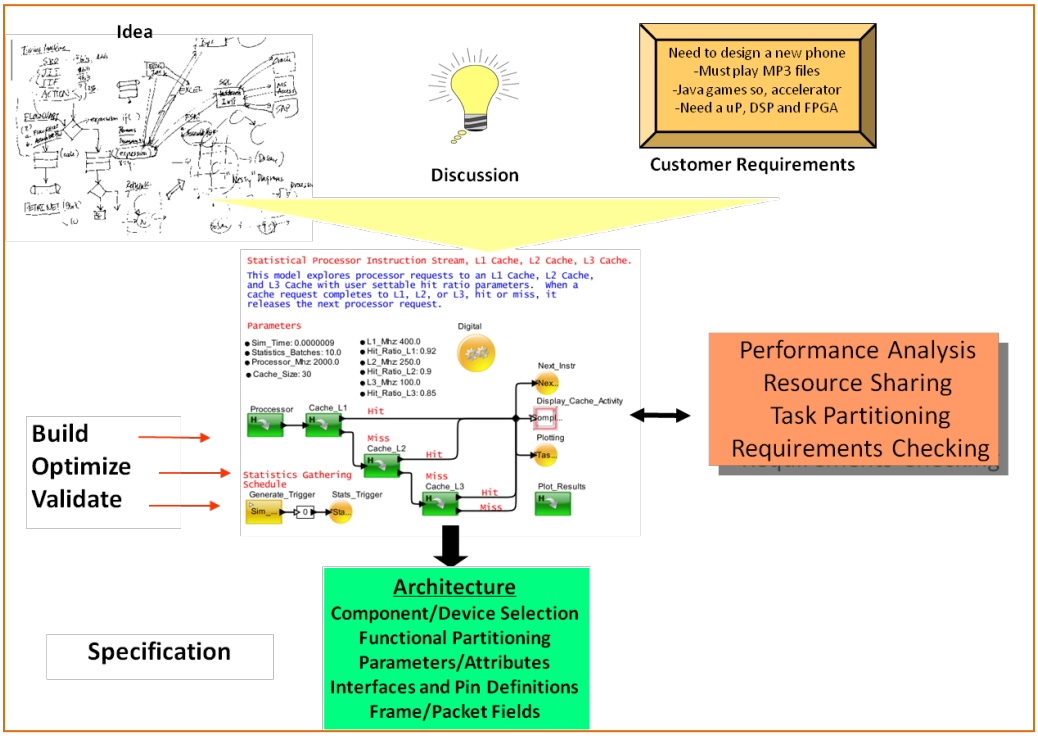


Figure 2: Transform Ideas into Modeling

To transfer your ideas into a system model, VisualSim offers a number of flexible methodologies. Irrespective of the preferred method, it is important to follow a Build-Optimize-Validate approach in constructing a model. Users start from a rough concept of the flow of data through an execution cycle and a platform architecture consisting of hardware, RTOS and software middleware. This concept is translated into a block diagram using blocks in VisualSim. The block parameters determine the specific functionality. After the raw model has been constructed, the flows and timing are tested against a variety of traffic stimulus. If using the statistical or scheduling model, the traffic generators emulate the operation of input queues and software execution. If using an instruction sequence model to emulate the software code, the code activity generates the stimulus to the model. Either approach can generate a large variety of test cases to thoroughly test the model. After validation, the model is refined for algorithm performance, architecture bandwidth and utilization, task deadlines and power constraints. The same sets of tests are repeated to make sure the block diagram compliant with the requirements.

Now, the model is iterated over the operating scenarios to determine the functionality, performance timing, and power or energy consumption. At this point, the user can conduct intelligent optimization for resource sharing, task partitioning, power minimization and improved functional operation. Out of these experiments comes the detailed specification that can then be implemented.

Methodology

A top-down design may be a new design that can start with a "blank piece of paper" or one that focuses on key algorithms in a new way. Whereas a bottom-up design may focus on incremental improvements to a successful design, that maintains the same internal structure, yet may improve the individual elements of the conceptual block diagram. Figure 2 and Figure 3 illustrate concepts relative to the conceptual block diagram, and the type of design one is creating. The best modeling approach is to use a combination of top-down and bottom-up design in the same system. This can result in a higher quality design, since concepts and details are given equal weight from day one.

Using both top-down and bottom-up design in the same system can result in an even higher quality design, since concepts and details are given equal weight from day one.

How to build a model like one thinks

Building a model like one thinks must relate the concept or idea to a model that is consistent in terms of the conceptual block diagram, model questions, model abstraction, modeling topology, and model data structures. In addition, there is notion of how the model might be built in steps, or phases, that coincide with the top level block diagram. In fact, translating the concept to a top-level block diagram is the first step to modeling a complete system, or portion of a system. Figure 2 illustrates a typical performance model that consists of the user's system represented as the green "Performance Model". The model block diagram also has a couple added items, namely a simulator, "Digital", a traffic source block, an analysis output block, and parameters in the upper right corner. Think of the green "Performance Model" as the concept, or idea, drawn on a napkin, or white board without any of the additional modeling information. It could be two to twenty interconnected blocks with interconnecting arrows. The arrows represent the movement of information between the blocks and can be data, control, or status. The transactions (arrows) can be simplex, or duplex, and there are no restrictions on how to represent a system.

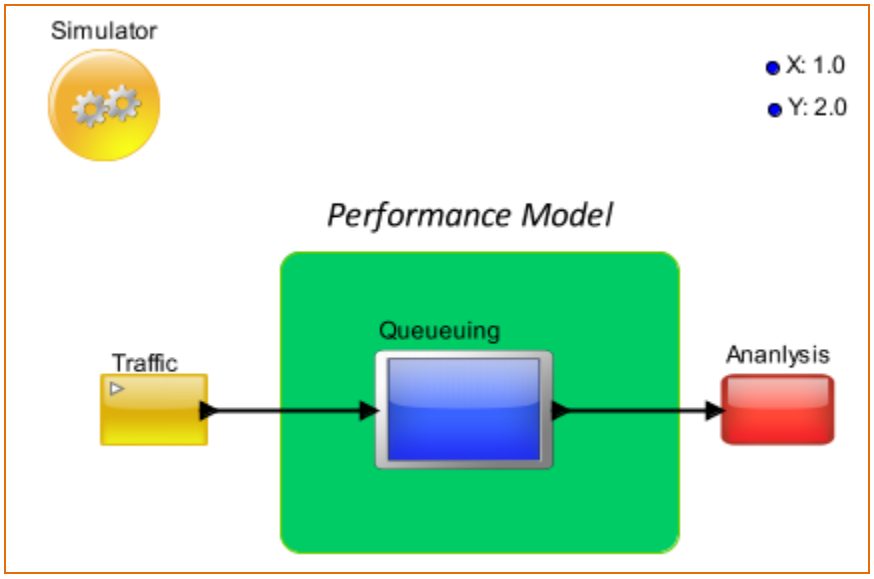


Figure 3: Model Block Diagram

Top-Down Modeling: This figure illustrates how one might consider the system choices and modeling choices at the same time. A model is like Einstein's "mental model" except it moves to a framework that can be tested and probed in the real world. It can be shared with other departments, or dispersed to groups around the world, so the mental model becomes an objective, real thing to discuss and evaluate.

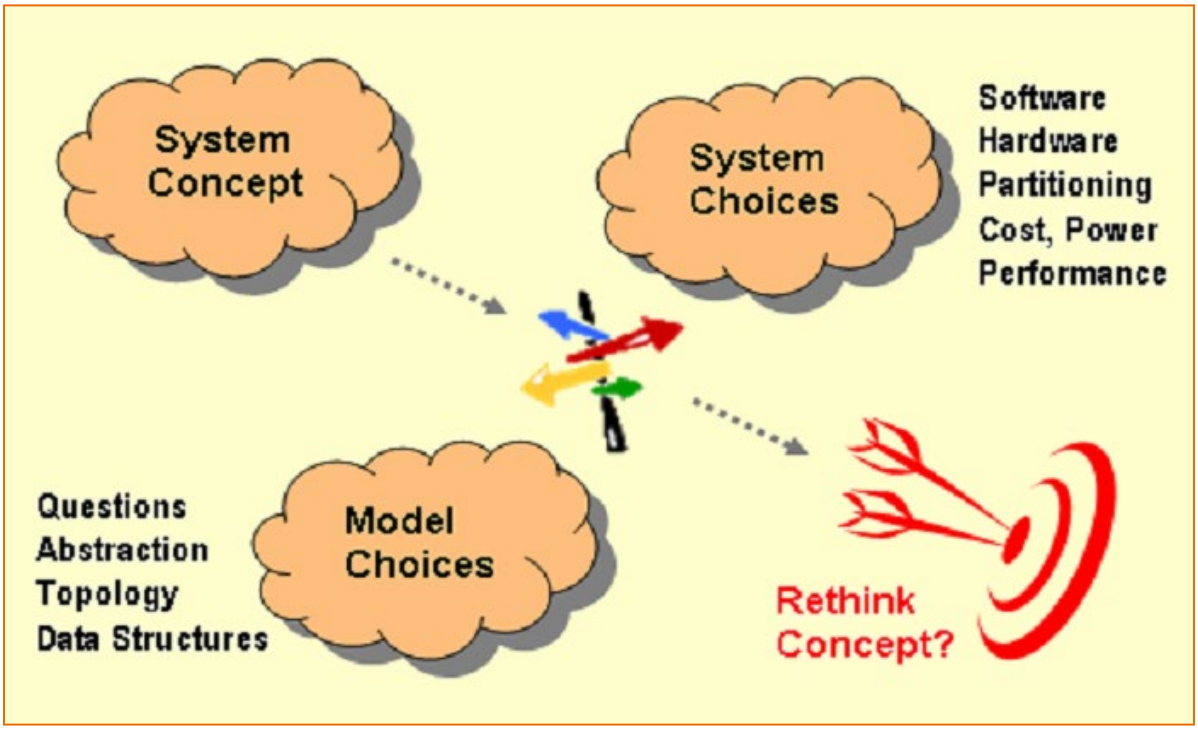


Figure 4: Top down Modeling

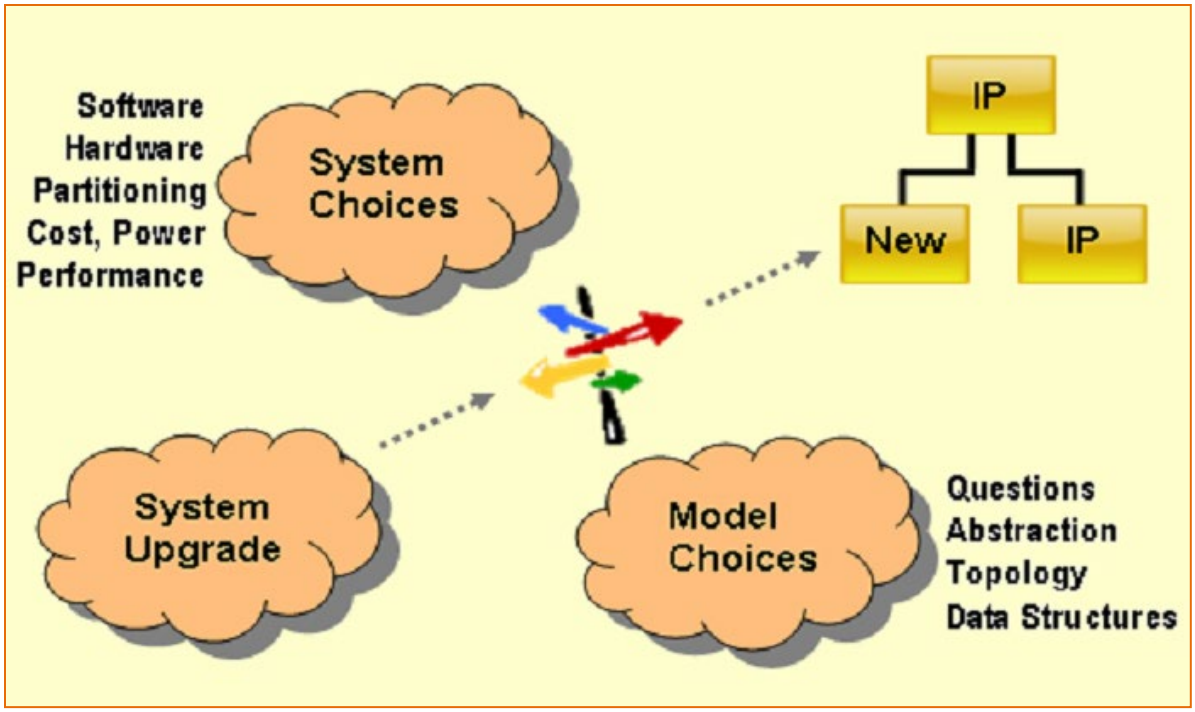


Figure 5: Bottom-Up Modeling

Bottom-Up modeling is analogous to top down modeling, except it starts with known details and builds into an existing framework. One will notice that the same considerations arise, just in the context of existing elements of a system or subsystem. Now let's look at the Model Choices in more detail. The model choices might be considered the alter ego of the system itself, in that it looks at the system differently than a set of interconnected blocks in a block diagram.

Model Questions

The most important aspect of modeling is the question, or questions, to be answered by the model. While this step may seem intuitively obvious in one respect, modeling questions may be overlooked too quickly. This is true if one starts to build a model before the modeling intent is crystal clear. First time modelers sometimes think the entire system, or chip, needs to be modeled to sufficient accuracy to obtain good modeling results. While there are cases where the entire system, or chip, may need to be modeled, there may be many cases where it does not, just the portion of the system that may be of concern in terms of utilization, throughput, or latency. Building a portion of a system as a model can reduce the modeling effort without sacrificing the quality of results in terms of accuracy or insight. If one questions the bus throughput in a new high speed multi-core processor, for example, then a bus centric model may provide quality results without all of the processor detail in many cases.

So, what are some typical questions one might be interested in modeling? If one assumes new software tasks, or threads, are going to be processed with differing priorities, how do the priorities and task arrival time impact overall processing? If one changes the priority on a critical task, will this be sufficient to improve throughput, and reduce task latency? In most cases, this will be true, but there may be a relative time aspect to the critical task that may reduce latencies on lower priority tasks, such that both benefit from the new ordering.

A big consideration of what questions one may want to answer relative to specific design, relates to what information is available, as well. If one wants to model a specific cache at a detailed level, but does not have the cache replacement policy from the data sheet, then one might consider using existing cache replacement algorithms in the architecture block first.

Another important modeling question that typically cannot be answered with a spreadsheet or many golden C models (not all): what are the peak utilizations for system processing elements? If the "peak" processing is above 80% for a system processing element, then the system may be vulnerable to last minute tasks added, or future growth of system itself. So, peak utilization, throughput, and latency are typical modeling questions of importance for different blocks in a conceptual block diagram.

After postulating the questions a model might answer, one might also consider what the output, or analysis of the model might look like. This is a useful step in creating mental models, building them, and running them: what type of output does one expect? In some cases, even very simple models can have outputs that are somewhat different than envisioned. This is useful in understanding the underlying system processes, interactions, and/or potential bottlenecks.

Modeling Topologies

The performance model shown in figure 2 is one type of modeling topology. There are other modeling topologies that may be more pertinent to a particular system or design. The modeling topology is getting closer to creating the conceptual block diagram in VisualSim. Typically, one adds a hierarchical block for each block in the conceptual block diagram and adds input and output ports to match the flow in conceptual block diagram. It is here that the simulator, traffic, analysis, and parameters can be considered for a performance model. One may want to contain the conceptual block diagram inside a hierarchical block to present the system as a full system, for a marketing or application engineering use, making the model easier to understand.

A platform model consists of behavior, or pure functionality, mapped to architectural elements of the platform model. Figure 6 shows an illustration of this mapping. The key advantage of a platform model is that the behavior algorithms may be upgraded without affecting the architecture they execute on. In addition, the architecture could be changed to a completely

different processor to see the effect on the user's algorithm, simply by changing the mapping of behavior to architecture. The mapping is just a field name (string) in a data structure transiting the model, so it is very easy to make this change.

Some platform models may have more than one architectural platform at the same time in the same model to compare them side-by-side, again modifying the mapping from one architecture platform to another. So, instead of a separate model for each architectural platform, one could have up to three architectures in one model, to compare side-by-side.

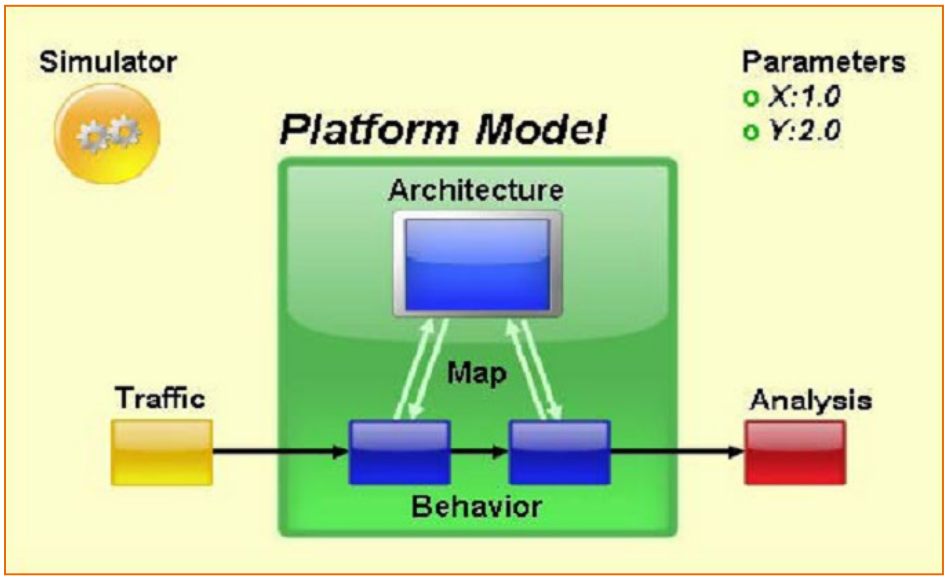


Figure 6: Platform Model

VisualSim can support static or dynamic behavior to architecture mapping, simply by changing the destination during execution. The destination resource might be modified based on actual processor loading to affect dynamic mapping. The architecture portion is where time-based events are executed and the behavior is where application-specific algorithms are executed.

An example platform model, shown Figure 7, has a simple behavior to architecture mapping. The behavior generates instructions to the Processor block. The Processor block sends fetched instruction and data cache information from the external cache, SDRAM blocks via the Bus blocks, for example. The user can modify the behavioral task, while the platform architecture remains the same. This means the processor, bus, cache and SDRAM blocks are fixed, while the behavior is modified.

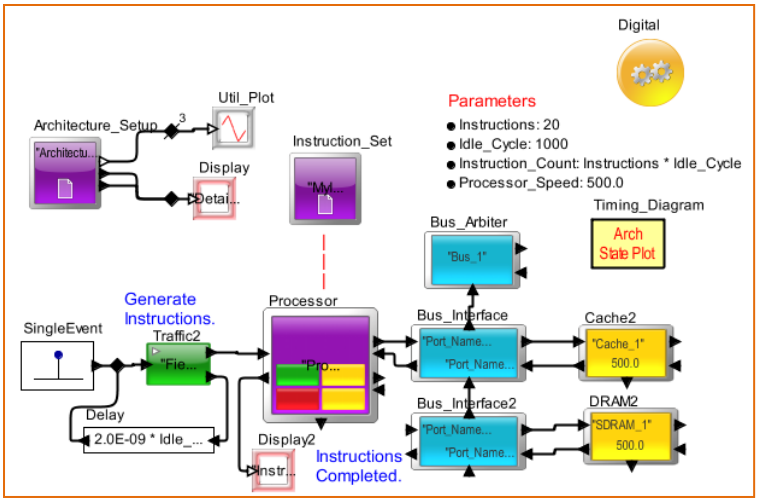


Figure 7: A Simple behavior to Architecture Mapping block diagram

The model can be further refined by adding a Finite State Machine (FSM) or script-based block to refine the performance or platform level models. The support blocks such as traffic and analysis, are quite similar between all the topologies. In general, custom models provide a higher level of accuracy for those portions under investigation, such as a new cache policy in a multi-core environment.

A good example of a custom model is the Audio Video Bridging libraries in Figure 8 which can be used to design a completely new AVB-based network to integrate all the equipment, upgrade existing networks, and to design the electronics that are used in such networks. The AVB system can include the talkers and listeners such as video cameras, radars, broadcast systems, displays, and Electronic Control Units. The network can include AVB interfaces, bridges, switches and gateways.

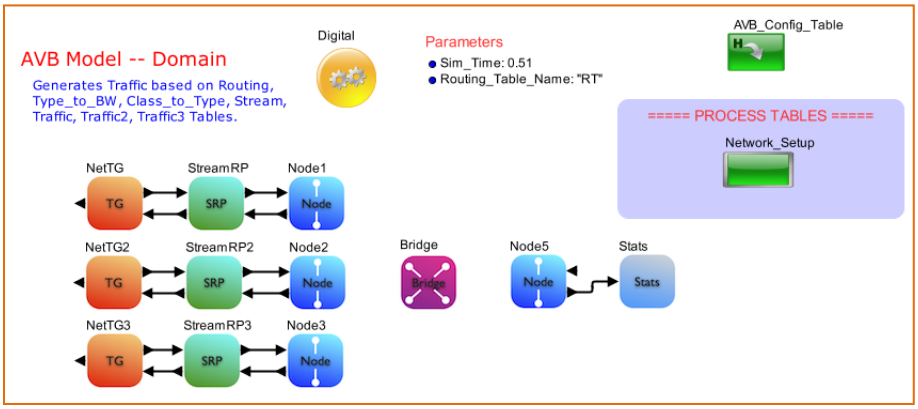


Figure 8: Custom Model

3. System Modeling

System modeling is all about designing the right product and implementing the product right. System design gives a greater flexibility in designing. Modeling can initiate from a block diagram (work out in paper with all objectives and flows finalized). The system model consists of simulator, parameter, traffic, system and Analysis. Simulator should be selected based on the user requirement. Parameters are used for performance analysis of the system and it is set fixed till the simulation completes.

Traffic blocks generates input stimulus based on the user-preferred timings. The input stimulus must contain the details about the type of operation (read or write), priority, etc. System blocks will process the input stimulus from traffic and the performance can be analyzed (in term of throughput, latency, utilization, etc...). Before assembling a model, user has to decide on:

- Questions answered by the model (eg, speed of the bus, total channel needed)
- Data Structures to support the information flow
- Topology (type of communication network)
- Once the System model is ready to simulate, the user must match the system output with the requirements. The user may try out different parameters and topology.

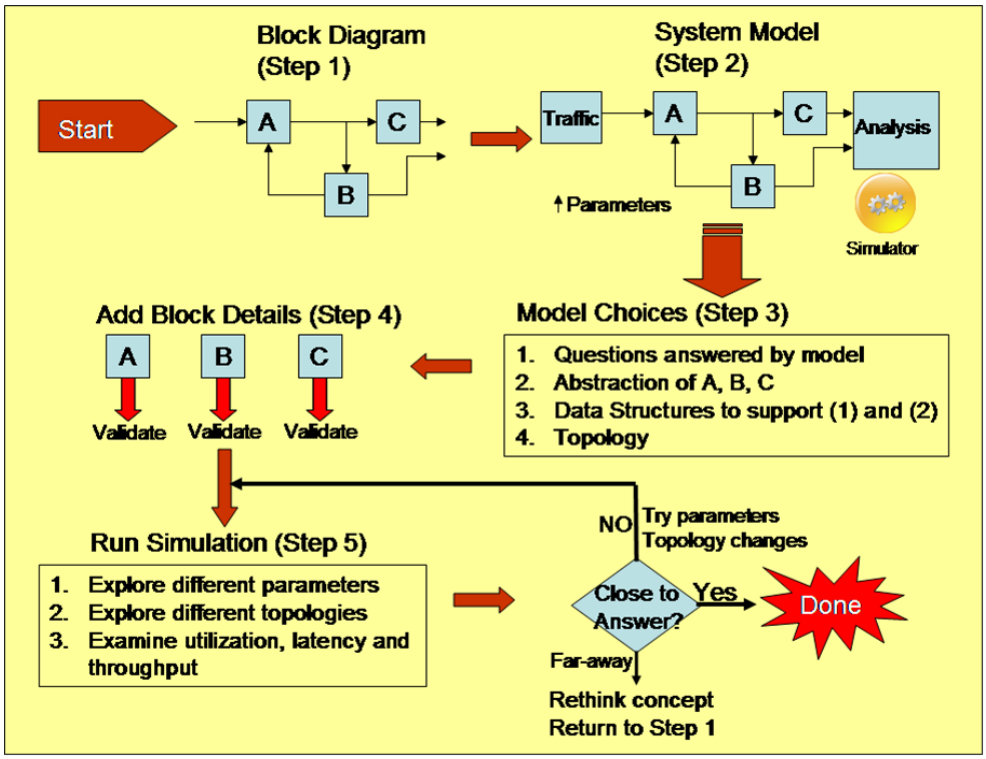


Figure 9: Modeling Flow Diagram

Why System Modeling

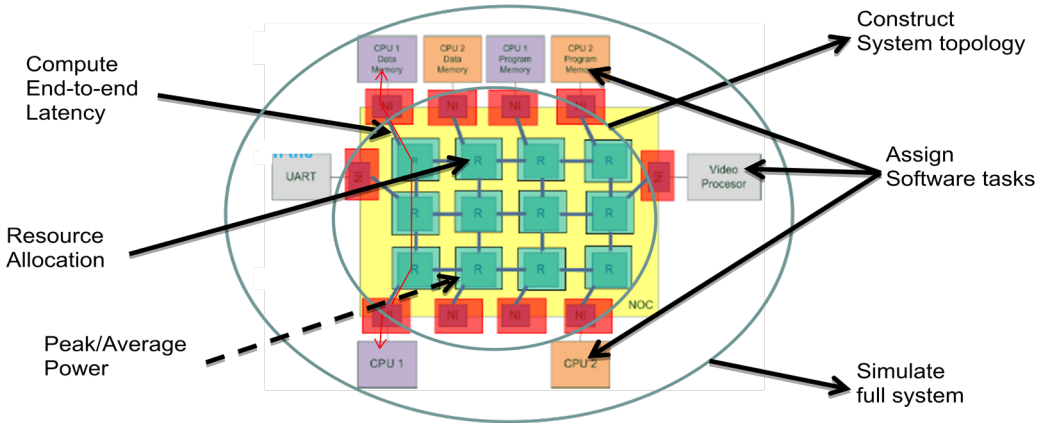


Figure 10: System Exploration Complexity

Figure 10 shows the hardware and software complexity of today's systems. It is difficult to conduct exploration with so many variables in Verilog or using cycle-accurate models. Hence a better alternate is required. This alternate is system design using VisualSim.

If there is only one application or the application is fully pipeline with no contention for resources, then a simple Spreadsheet can be used to compute the throughput and latency. The applications can be easily scheduled and the arbitration will be straight-forward.

Unfortunately, most of today's systems have multiple applications. Take a look at the flow and the allocation on Figure 11. There are multiple data flows through the system. There is contention at the processors, I/O such as USB, bridge and the buses. It is very difficult to predict the system performance with multiple variables, different distributions of traffic patterns and multiple hardware and software components, and all the flows. The knowledge of all the flows, traffic rate and execution time is required to create a high performance scheduling scheme.

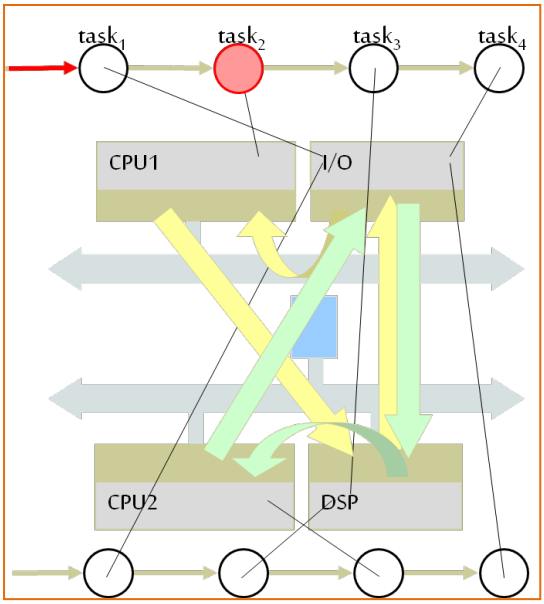


Figure 11: Complexity of System Design

VisualSim provides the ability to quickly construct a system and run a large number of design variations. A real system is always migrating from one application or throughput to multiple application and higher throughputs. The designer must look at system requirements, the increased traffic, the application flows and has conducted a variety of trade-studies. Such models can be assembled using VisualSim, run analysis and determine the optimal system configuration. The configuration is based on the power and performance consideration.

In VisualSim, you can start the analysis for a specific sub-system and then extend for the full system. There is no need to build the full system at the beginning.

4. Types of Modeling

There are many types of system modeling possible in VisualSim including:

- Flow control and behavior modeling ($VS/demo/networking/Flow_Control/Flow_Control_Xon_Xoff.xml)
- Performance Modeling (Open the following model in the BDE: $VS/demo/performance/VME/ VME_Bus_Model.xml)
- Signal Algorithmic Modeling (Open Help Page or the following model in the BDE: $VS/demo/signal_processing/ConvolutionalCoder/ConvolutionalCoder.xml)
- Control and Mixed Signal Modeling ($VS/demo/analog/SigmaDelta/SigmaDelta.xml)
- Architecture Modeling (Open the following model in the BDE: $VS/demo/Partitioning/SoC/Power_Perf_example.xml)

- Software design and verification
  $VS/demo/others/EnergyConsumptionModel/EnergyConsumptionModel.xml

Flow Control and Behavior Modeling

Queue management, flow control, arbitration and scheduling are design trade-offs that are based on variable

1. Number of input streams
2. Data rates
3. Queue depths
4. Scheduling or scanning or polling logic
5. Credit policy
6. External flags

This analysis is a stochastic simulation to explore the quality of service and to determine whether the required throughput has been achieved. This model requires knowledge of buffer state and usage at multiple locations, before making a decision on data transfer. The ingress and egress can have a large number of channels or virtual connections. These are modeled using Traffic, ExpressionList, Queue and Server. The logic is constructed using the Script or Finite State Machine. Output analysis will be latency, buffer occupancy, and throughput. Figure 12, shows an example of a Xon-Xoff flow control logic.

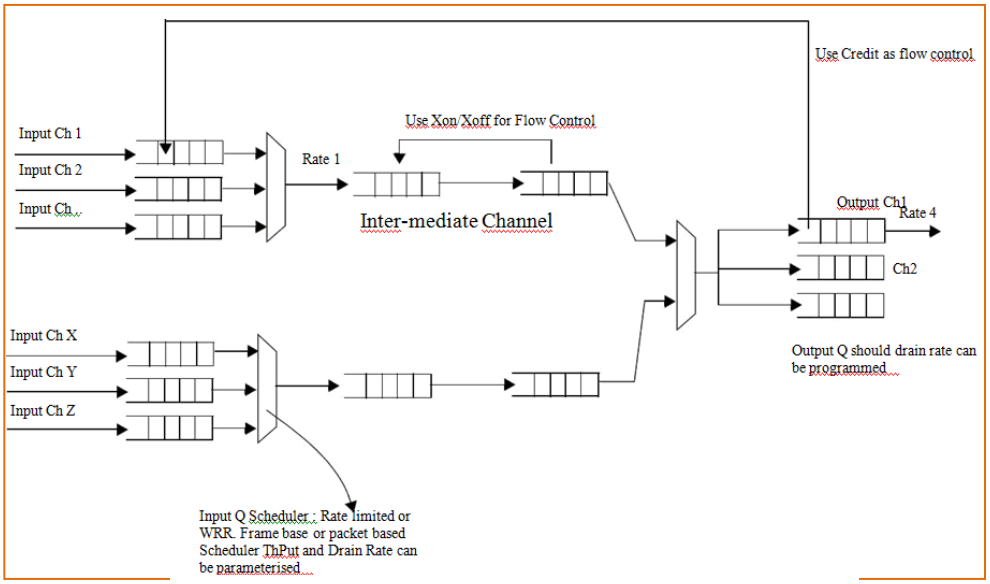


Figure 12: Workload and Flow Control Studies

Performance Modeling

Performance models will be created using the Resource and Processing blocks. Performance modeling focuses on the overall system metrics such as power, performance and cost. This

analysis is used to identify capacity limitations and system bottlenecks. These models tend to be based on event trigger and do not run at cycle-accuracy. At this stage of the design, a lot of unknowns need to be captured, specification is under development and software is not available yet. Figure 13 shows the performance model of a Hardware Sub-System- Bus + Master/Slave devices.

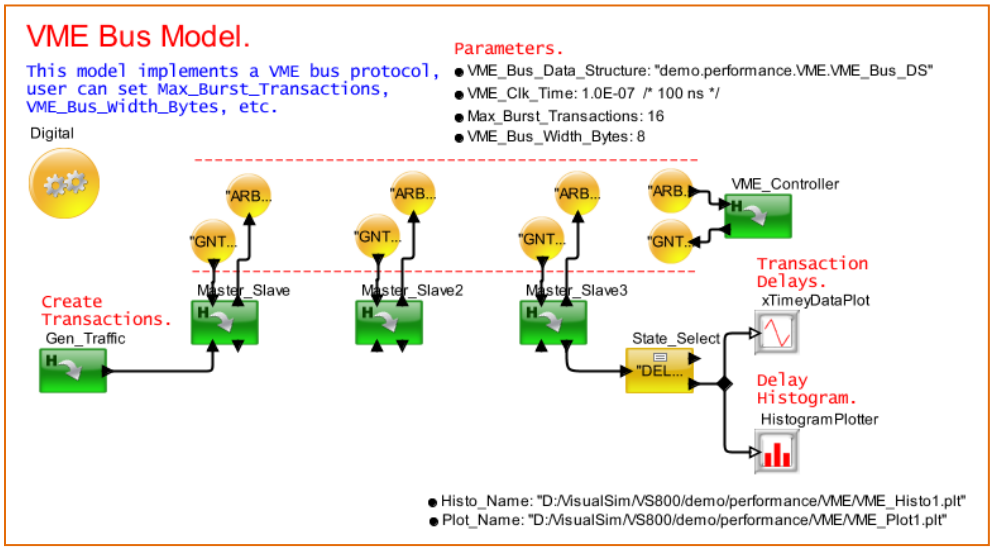


Figure 13: Performance Modeling

Algorithmic Modeling

Figure 14 shows a signal processing algorithmic model that is assembled using the Functions and Application-specific libraries. Algorithmic modeling is related to control, DSP, image processing and analog functions. The emphasis is on the correctness of the mathematics. The standard analysis recorded would be Bit-error rate, Signal-to-noise ratios, waveforms and pole diagrams.

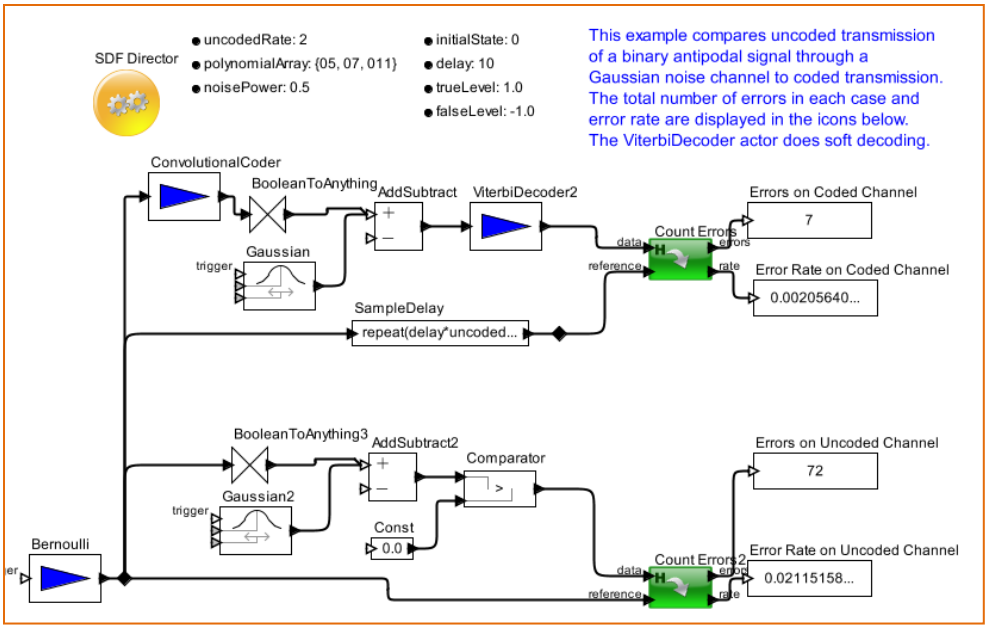
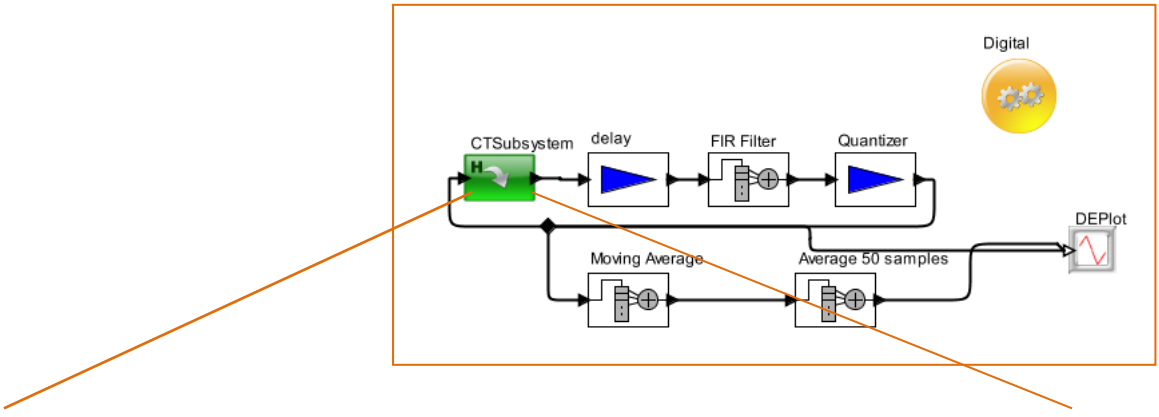


Figure 14: Signal Processing Algorithm model of a Convolution Encoder

Mixed Signal and Control Systems

Mixed Signal and control system modeling requires the knowledge of two different time domains- one where the time changes in a continuous manner (Continuous) and the other where the time moves in discrete but random distances (Discrete Event). Good examples of control system design are MEMS accelerometers and evaluating the impact on the engine control in a noisy car tracking situation. For mixed signal, the Sigma-Delta A/D converter in Figure 15 is a good design target. Here the evaluation is to look at the impact of frequency and signal changes over time. Another aspect is the loss of data when moving from one time domain to another.



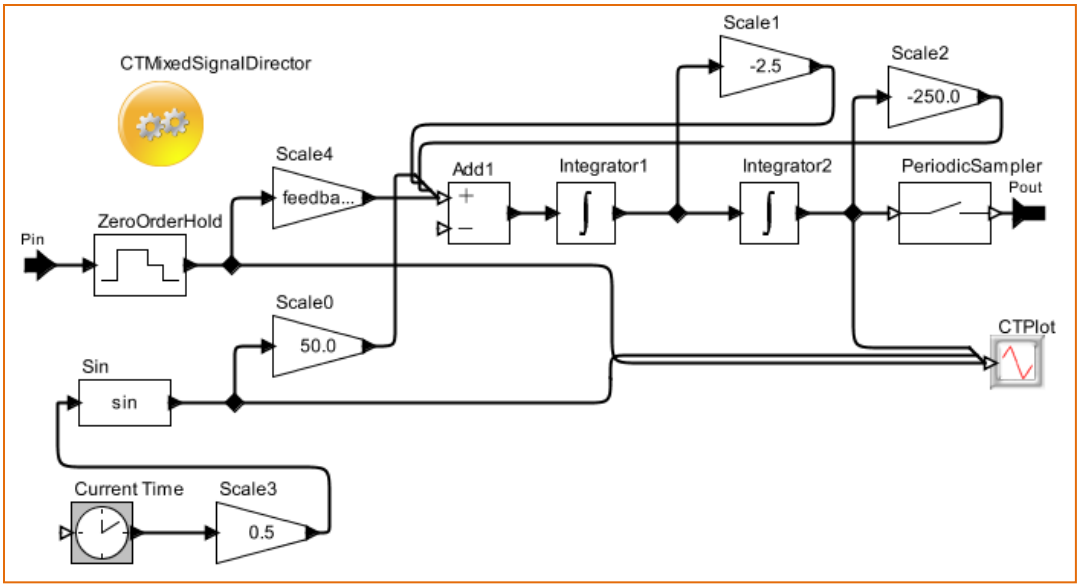


Figure 15: Mixed Signal Modeling

Hardware and Software Architecture Exploration

Figure 16 shows an Architecture model that will combine the Architecture library components with the Processing and Resource blocks. Architecture exploration is a detailed and accurate exploration of a system platform. The system platform can be a SoC, software or a network of systems containing hardware and software. The focus is on sizing the individual components, distribution of tasks on to the distributed system connected by networks, partition into hardware and software. The evaluation is for both Power and Performance.

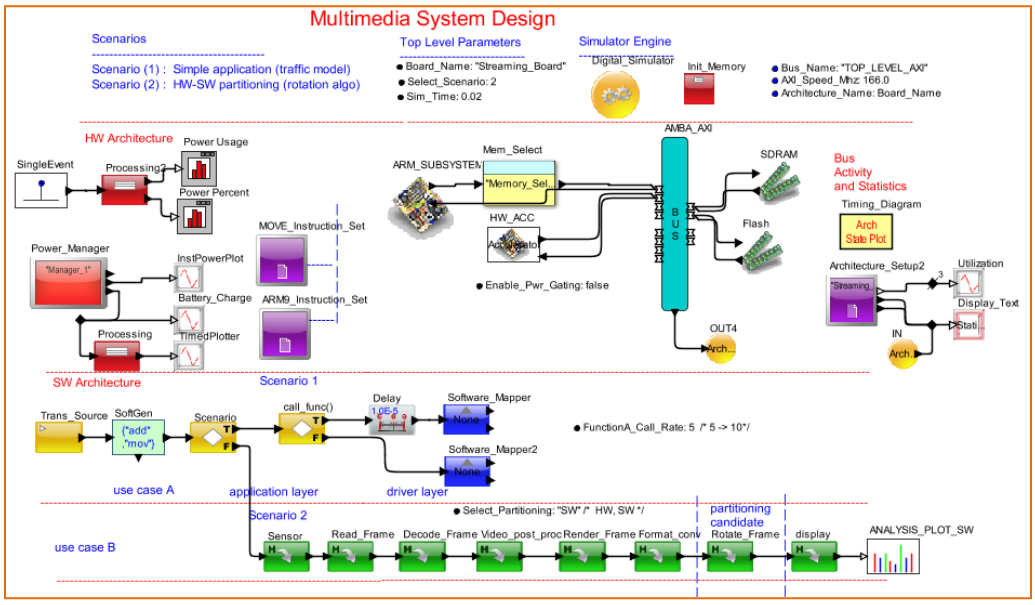


Figure 16: Architecture modeling using the hardware architecture library

5. Power Exploration

The VisualSim Power Modeling Toolkit is a System-Level Power exploration solution that captures dynamic power of the entire system in a model. The solution enables the Architect to trade-off performance and power in a single architecture model. The model can be of a SoC or a Distributed, Networked system. The Power Analysis is conducted in the specification phase. The Power Analysis does not require detailed software code, RTL or placement information to execute the simulation.

Using this solution, an Architect can determine the instantaneous and average power consumed by the entire system or a specific component. The power exploration can handle both standard components such as processors, and custom components such as hardware accelerators.

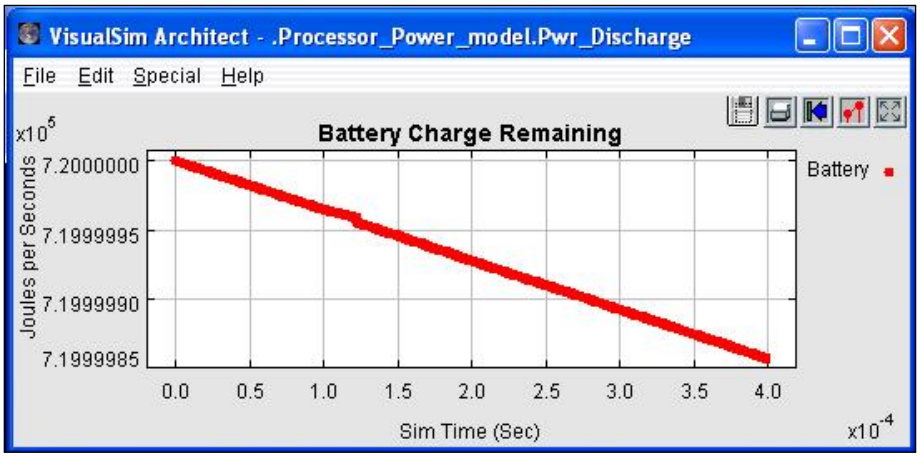


Figure 17: Battery Charge Remaining

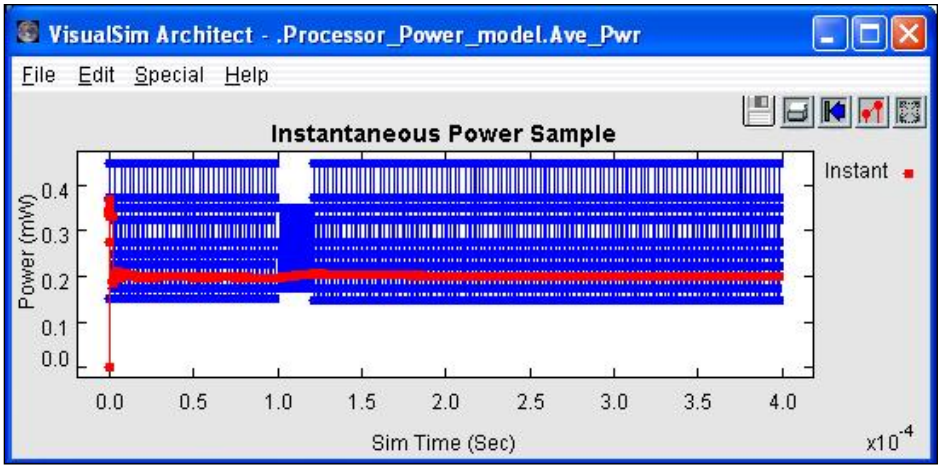


Figure 18: Instantaneous Power Sample

The Power Modeler updates the instantaneous, average and cumulative power using dynamic state change information of the individual devices. The power can be analyzed on an existing system model by instantiating this power module and entering certain power attributes for each device or IP block. The effect of the power management performance such as transition cycles is available as a definition.

The power manager is fully integrated with the Architecture Modeling Toolkit, scheduler blocks and Script. Moreover there are function calls to view current power consumed by device, update the power levels, change the power state and charge the battery.

6. Parts of the Model

1. Create a functional decomposition of all the operations in the system.
2. List the types of resources and the level of detail that each resource must be described at. The refinement of the resources creates architectures. At that point, what types of architectures must be evaluated.
3. Determine the input stimulus to the model. This can be the use cases, arrival rate of packet, sensor traffic or an instruction stream.
4. Create a list of analysis by resource or by behavior that needs to be evaluated. This is the evaluation criteria.
5. **Behavior Flow**: Draw or create a flow diagram to identify all the steps that the incoming data request is going to go through. For example a packet in a router will go through an Input **Port**->**PHY**->**Mac**->**TCP Offload Engine**->**Processor** (**Side Path** to **CAM** and **SRAM**)->**Output Port**.
6. **Control Decisions**: Examples are a lookup in a CAM (Content Addressable Memory) to determine the action to be performed on the packet based on a content of the header field. The control operation in the processor decode stage will trigger the dispatch queue to send the next instruction to the Stage 1 of the pipeline.

7. Explanation of a System Model

System Model = **Source** + **System** + **Results** + **Simulator** + **Parameters**

System Model is the mathematical representation of the stimulus, description of the actual or theoretical physical system and the viewers for the output. It consists of a **Source**, **System**, **Results**, **Simulator** and **Parameters**. VisualSim library contains an individual directory for Source and Result. System Model is used to analyze the performance of large network in term of utilization, throughput, or latency and estimates the power. A System Model consists of traffic sources sent to a block diagram of a system that generates real-time results, based on top level parameters for a given simulation and simulator setup. VisualSim also provides many pre-built libraries of common system components that can be drag-n-dropped into a model and configured with a few block-level parameters. Unlike EXCEL spreadsheets, VisualSim can determine peek loading conditions based on concurrent, or parallel execution using a global simulation time.

To Monitor Real Time Execution : Right-Click (on Block) -> Listen to Block

To Monitor Real Data Flow : Right-Click (on Block port) -> Listen to Port

More Info on Block : Right-Click (on Source Block) -> Get Documentation

Source

This is the stimulus to the model. This can be a distribution-based, sequence of reads and writes, and triggers the series of activity that the system under test responds too. The traffic can be either a data structure (transaction containing fields), pin value or token (any data type). The rate of arrival of the data can be modeled as a fixed sequence, distribution-based, standards-defined or a trace from an existing system. Designing the right traffic generator is important in developing a reliable model. Traffic Generators can represent packet data at an input port or a stream of instruction to a Processor/RTOS.

System

This is the design. This can be Hardware oriented, Traffic oriented, Network oriented, Multi Bus Topology, etc. The System construction contains Behavioral flow and System flow. The behavior flow describes the order and dependency of the tasks that are processed on the data. The behavior can contain multiple flows that are dependent or independent. The behaviors have no notion of an implementation. In a performance model, they do contain the timing information. The actual execution of the timing is performed on the architecture. Architectural definition focuses on the hardware executing custom instruction and processors executing software

instructions, or software tasks. For advanced models that require cycle-accurate and instruction accuracy, and use standard component definitions, the Architecture library components are very well suited. VisualSim contain large number of IP Blocks such as Processor, Cache, SDRAM, SRAM, DRAM, Memory Controller and Flash. Large numbers of bus model such as Shared and Linear Bus, Request Acknowledge Bus Controller, Point-to-Point Bus Controller, DMA Controller and I_O Controller. Some example systems are categories based on operation are:

Behavior flow + Architecture Flow

Pre-Built Models ->Popular Examples ->Application -> Multi functional FAX System

More Hardware Oriented

Pre-Built Models ->Popular Examples -> Network Equipments -> 3G Data Router

More Traffic Oriented

Pre-Built Models -> using performance modeling library block -> Arbitration -> Systems with multiple sets of queue, muxes, flow control and credit policy

Cycle Accurate Modeling

Pre-Built Models -> Processor, Bus and Networking Standard Models -> AHB to AXI

Simulator

The simulation engine, or simply simulator, controls and coordinates all system components executing in time simultaneously through a digital event mechanism. The digital event mechanism insures that all concurrent events execute the same way each time the simulator is started. In addition, the top digital event mechanism can support analog execution and finite state machine execution in lock-step. The simulator has a parameter for the stopTime of the simulation, and during execution of the model coordinates control, data, and status being sent between system components, or blocks. There are also blocks, called hierarchical blocks, that simply contain more blocks to make the top level system model look exactly like the original block diagram. For time-based simulators such as Discrete-event simulator, it is important to add the local simulator icon to large models with many levels of hierarchy. This will keep the events occurring at that level of hierarchy local and maintain a smaller calendar of events, thus increasing simulation performance.

Data Structures and Tokens

A model is simulated by sending information across the topology of blocks and wires. The information can be a single value such as integer, double/float, string, complex vector, arrays or matrix. Alternately a set of these values in a single package called Data Structure can be passed. The approach of using single values is suitable for algorithm, control systems and implementation -level co-simulation. Data Structures are best suited for conducting performance and architecture analysis.

Parameter

Parameters can be added to any model and at any level of hierarchy. These are added to the BDE by selecting the parameter from the Library Folder at Model->Parameter. Parameters are a convenient way to modify fixed model attributes to create different simulation runs and conduct different scenarios. A parameter can be used to define the number of input queues, processor speed, input rate or the number of statistics bins. Parameters can be linked to the equivalent parameters in the blocks of the same level by using the parameter name within the block parameter. Parameters can be expressions containing other parameter names. Parameters are constant during the simulation. They will be evaluated at the start of a simulation and will be constant for the duration of that simulation.

Results

Simulation results can be user defined metrics, such as the best bus latency for a given packet size; or more common results like end-to-end system latency. Users can also calculate data path throughput, and the utilization percent of individual system components, like a cache. In addition, the results from a detailed traffic model can be imported into a larger system model, or be reused for verification of the implementation.

8. Model Abstractions

Statistical level Modeling

Statistical level models are constructed to gain a better understanding of the whole system which would not otherwise be possible with analytical methods. Refinement models are done for potential problem areas to explore in detail. For example a higher level of abstraction/statistical level mean it is modeled at the transaction level. Such models will have an accuracy range of 60% to 80% for functionality and 80% for the transaction flow to the element being investigated. 75% to 80% accurate traffic profile with a detailed processing element is sufficient to make valuable design decisions on peak utilization, throughput, or latency.

Abstract model provide an insight into system in an entirely different light. In terms of innovation, thinking along parallel, but separate lines of thought, can provide insight into a system that was otherwise hidden in the details.

Statistical level of modeling is the right choice if the user is doing first level of system sizing to complete system architecture.

Figure 19 is an example statistical processor based system with three levels of cache hierarchy. The Statistical process is actually a traffic which generates statistical stream of instructions based on the processor speed. The Statistical cache hierarchical block accepts cache requests and it delays the request based on the instructions defined, and also it checks for the cache hit and miss ratio to transfer the control to other blocks for further processing.

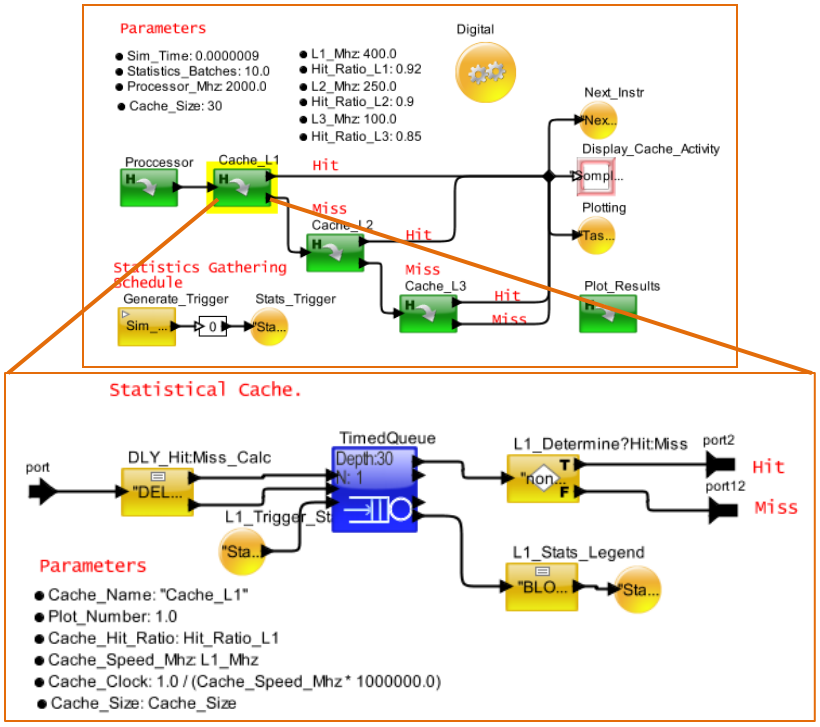


Figure 19: Statistical Processor Model

Hardware Level Modeling

Hardware level models are constructed to determine the system impact of particular technology detail of the hardware then one should choose Hardware level modeling or timing accurate abstraction level. Accuracy range of Hardware level models are of 80% to 95% based on the level of details at which model has been developed.

Figure 20 is an example hardware level accuracy model which is developed using prebuilt library blocks which are timing and functionally accurate. At this level of modeling there will be a very limited scope for modifying the system topology and parameterization. This level of modeling methodology will be ideal when the architecture is finalized but needs further verification of specification.

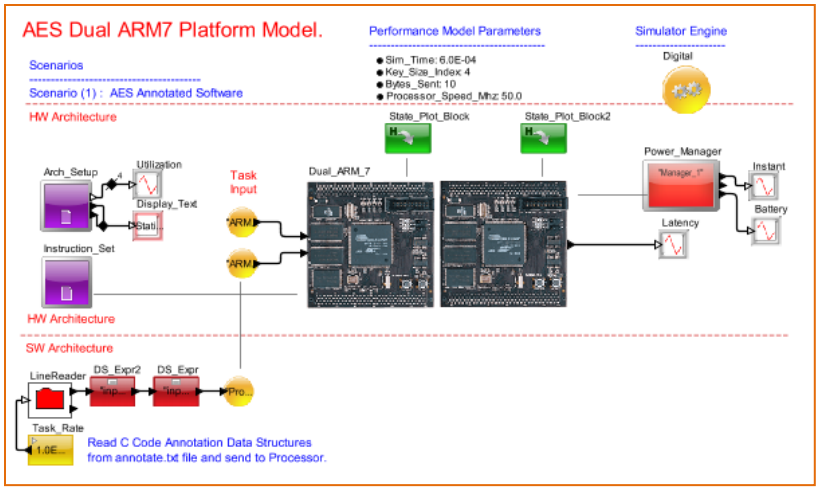


Figure 20: Hardware level model

Cycle-Accurate level model

At this level of abstraction, the models will be close in accuracy to the implementation. The quality of the models is only as good as the data provided. For example, a key ingredient of response times in storage devices is cache-hit ratio. System designers generally have limited knowledge of the exact cache-hit ratio. The actions leading up to and after this specific function are modeled in detail. Here experiments with different attributes of the cache algorithm are used to determine the average and worst-case response times.

If the user is concerned about tuning of specific parameters of a bus or developing a new arbitration algorithm or enhancing an existing bus then Cycle accurate level modeling approach should be used.

Figure 21 is a cycle accurate level model with trace file based traffic generator, Cycle accurate cache and AHB bus. The model developed at this level needs very detailed level of details on each transactions happening between different devices on system. The Architecture may not be suggested to make any modifications but is purely developed to verify details in RTL. User has freedom to change the parameters like clock speed and buffer size.

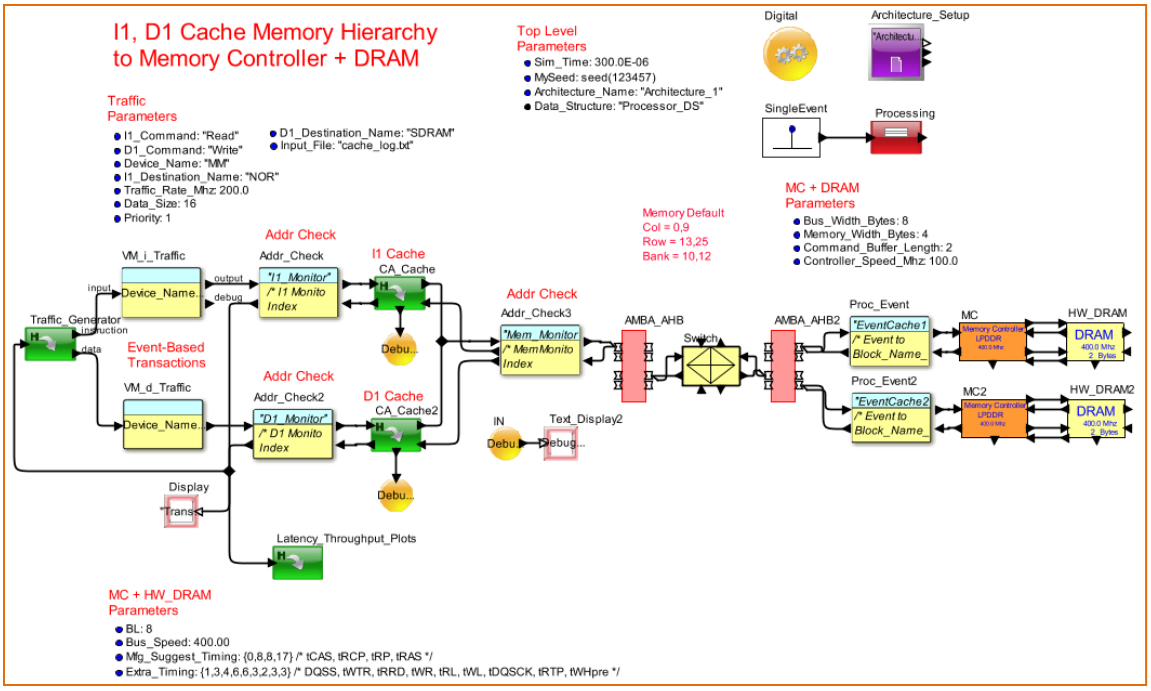


Figure 21: Cycle accurate level model

9. Assembling a Model in VisualSim Architect

Model Creation

This section is about creating new system models using VisualSim library blocks and run simulations to validate the functionality of the model developed.

Example:

Goal: This tutorial will teach you how to use the Block Diagram Editor (BDE), construct models out of the blocks contained in the VisualSim libraries, run dynamic simulations, combine different outputs, and interpret the simulation results. The following summarizes the concepts and tasks you will learn in this session.

1. **Understanding the Block Diagram Editor:** It is very important to know how to populate the required library blocks from the modeling library section on tool and also understanding the GUI features. In a modeling and simulation environment one should aware of making connections between different blocks in a system model.
2. **Generate transactions:** Input transactions play a very important role in analyzing the system performance whether it could be latency, throughput etc.
3. **Add content to the transaction:** The data structure fields are required to carry out various computations in the system models, identifying and including those contents or data structure fields in a transaction will be done in multiple ways.
4. **Manipulate FIFOs/Queues:** FIFO's/Queues are crucial part of a performance level model. One can manipulate a FIFO/Queue block by providing values such as Priority, FIFO/Queue depth, Reject mechanisms etc.
5. **Display output and generate latency graphs:** Output results provide the answers that one is looking for from the system model developed. Results are mainly provided in Text and Plot formats using which users can analyze them based on their requirements and the level of detail provided for the system model

Block diagram overview:

This model will be used to study the performance of a system that has a priority-based FIFO/Queue. The block diagram is shown in Figure 22. A Traffic Generator sends data packets or requests to the FIFO. The FIFO outputs the next transaction after the previous transaction has been processed.

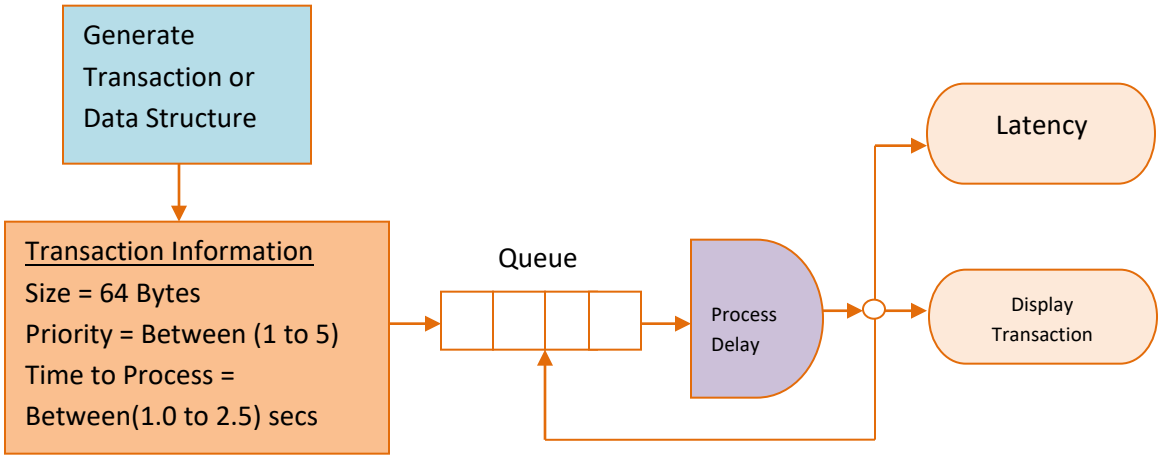


Figure 5.1: System Block diagram

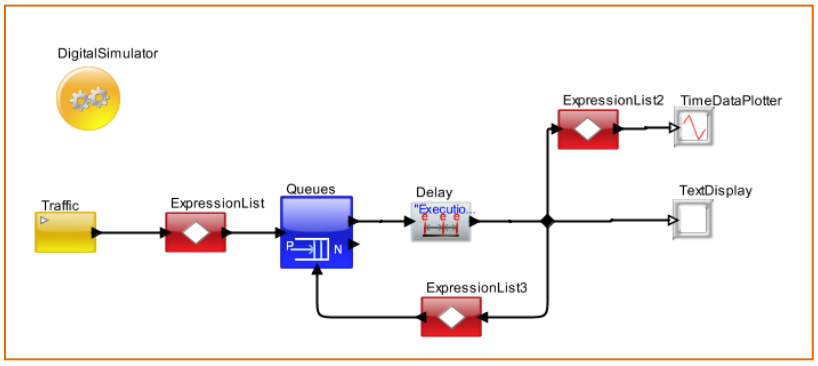


Figure 22: VisualSim Model

Construction Steps:

1. Open a Block Diagram Window.
2. Select the desired Simulator, as this example is for performance analysis; select **Digital Simulator.**
3. Define the data structure fields in a text file. The contents in data structure fields will be parsed as transaction.
4. Drag and drop the required library blocks from Libraries section in BDE.
5. Configure the parameters in library blocks as required.
6. Drag and drop Plotters and text display blocks from library section in BDE. These will show contents of each transaction and end to end latency achieved.

Please refer our Tutorials section for detailed instructions here.

Error Messages

If during model simulation you encounter error messages due to the following type conflicts:

(VisualSim.actor.TypedIOPort(..Expression2.output),general)<=
(VisualSim.actor.TypedIOPort(..xTime_yData_Plotter.input), double)

Right-click on **ExecutionList2** block and select **Customize**->**Ports**. The second port in the list is **output**, which is mentioned in the error message. In the column called **Type**, select "**double**" to fix this problem (note that step 4b above must not have been completed).

Results

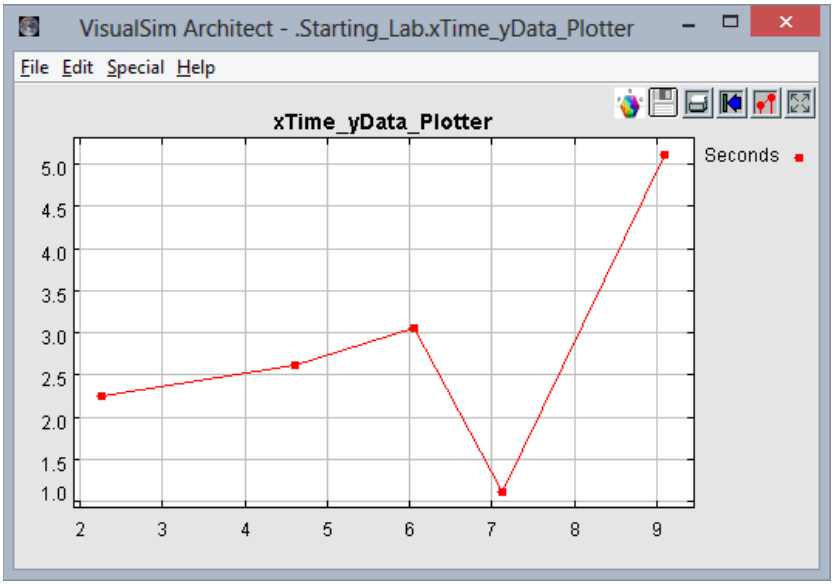


Figure 23: Latency Plot

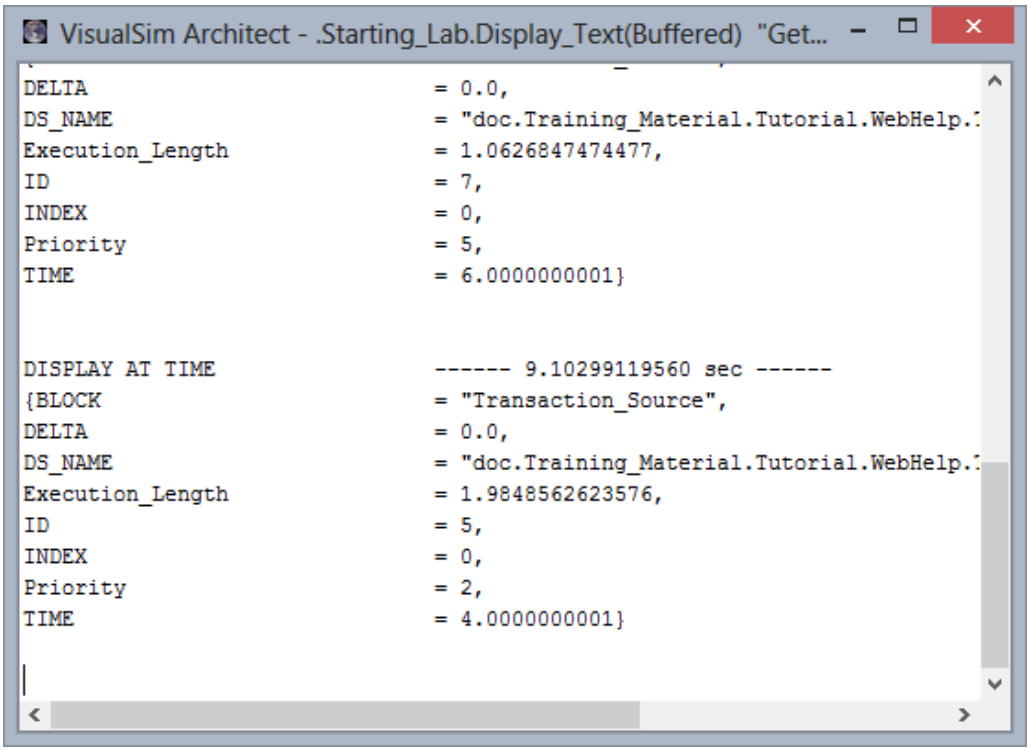


Figure 24: Text Output

Analyzing Results

1. Vary the parameter "**Execution_Length**" in "**ExecutionList**" block and analyze the Latency plot.
2. Vary the Inter-arrival time in Transaction source by varying "**Value1**" and "**Value2**"

Model Animation

You can animate block diagrams to debug your simulation or simply to view the dynamic operation of the system model. VisualSim animates a model by highlighting the executing block, based on the 'Debug/Animate Execution' time setting (msec). One can also turn off animation by using the main menu bar 'Debug/Stop Animating'. The figure below illustrates "ExpressionList" Block is currently firing with animation enabled.

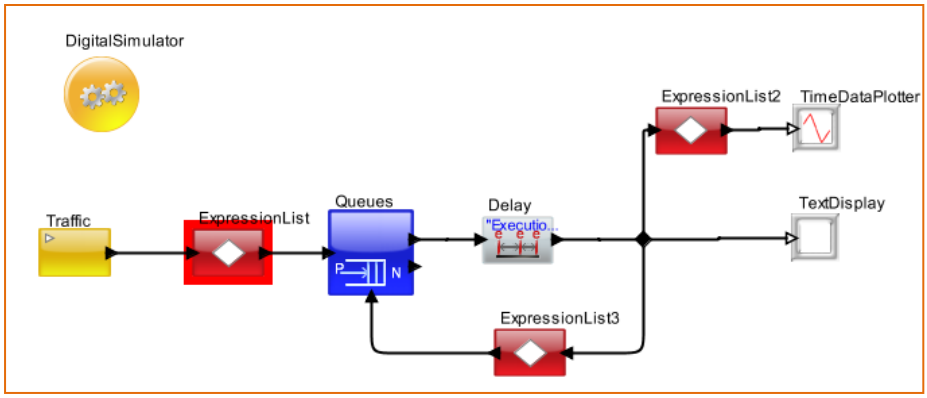


Figure 25: Model Animation

10. Understanding Errors and Messages

Exception Message

If incorrect value is set for the parameters you will find an exception message with following details:

1. **Block Name** - Specify the exact location of the problem in which Block

   Format: Model name.Block name.

2. **Error** - Specify the outline of Error.
3. **Error_Number** - For related error number refer the Debugging Document
4. **Possible_Solution** - It displays the probable solution to rectify the error.
5. **Description** - It displays a brief description on the error

An example exception message is given below:

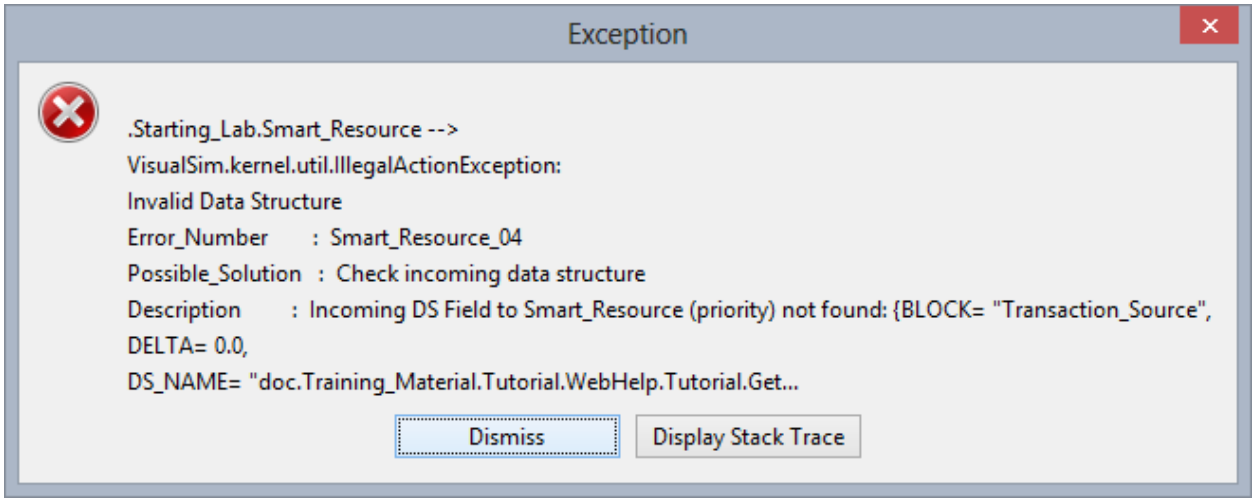


Figure 6.1: Exception Message for incorrect value set

Port Conflict

The Exception windows describe that conflict in data type occurred between input and output port. The possible solution for this exception is to set the data type of both port to be same.

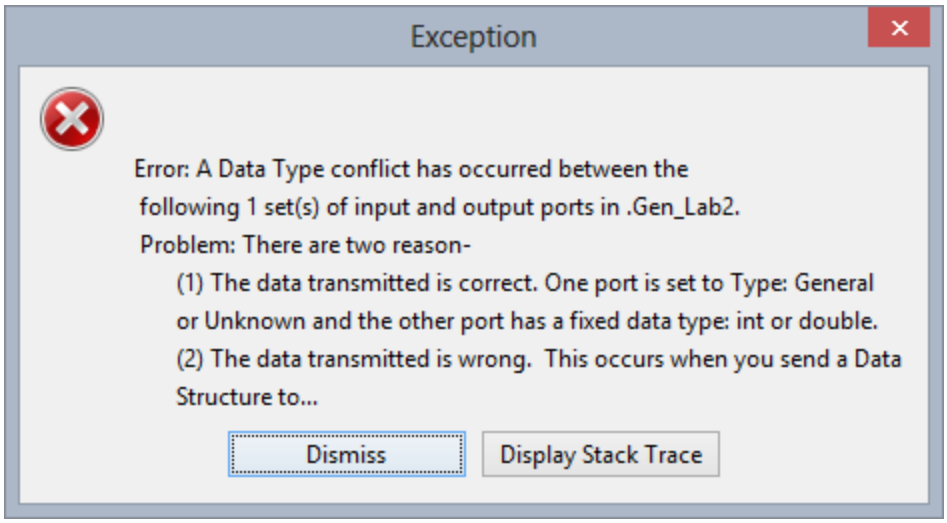
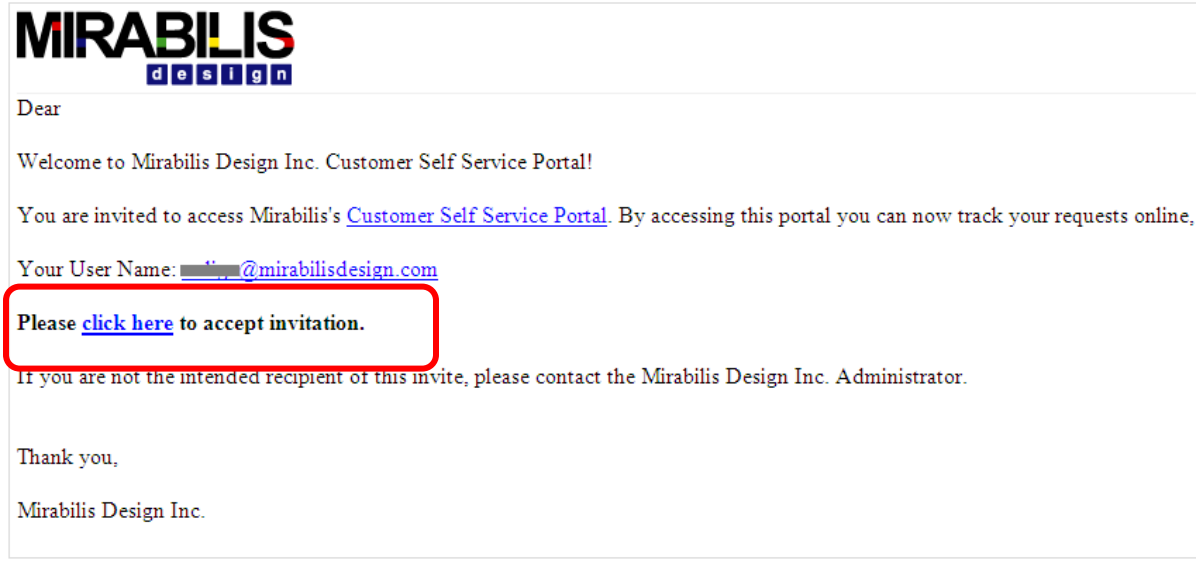


Figure 6.2: Exception message due to Port Conflict

11. Online Technical Support

Creating Customer Self Support Portal Account.

a. Accept the invitation from Mirabilis Design to join Customer Support portal



Dear

Welcome to Mirabilis Design Inc. Customer Self Service Portal!

You are invited to access Mirabilis's Customer Self Service Portal. By accessing this portal you can now track your requests online,

Your User Name: [redacted]@mirabilisdesign.com

Please click here to accept invitation.

If you are not the intended recipient of this invite, please contact the Mirabilis Design Inc. Administrator.

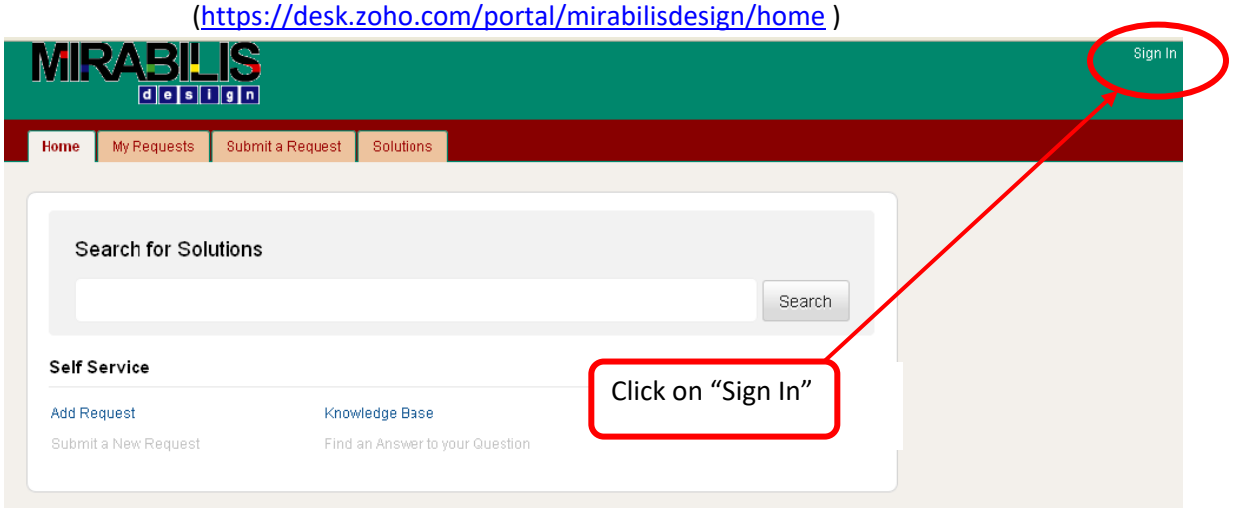
Thank you,

Mirabilis Design Inc.

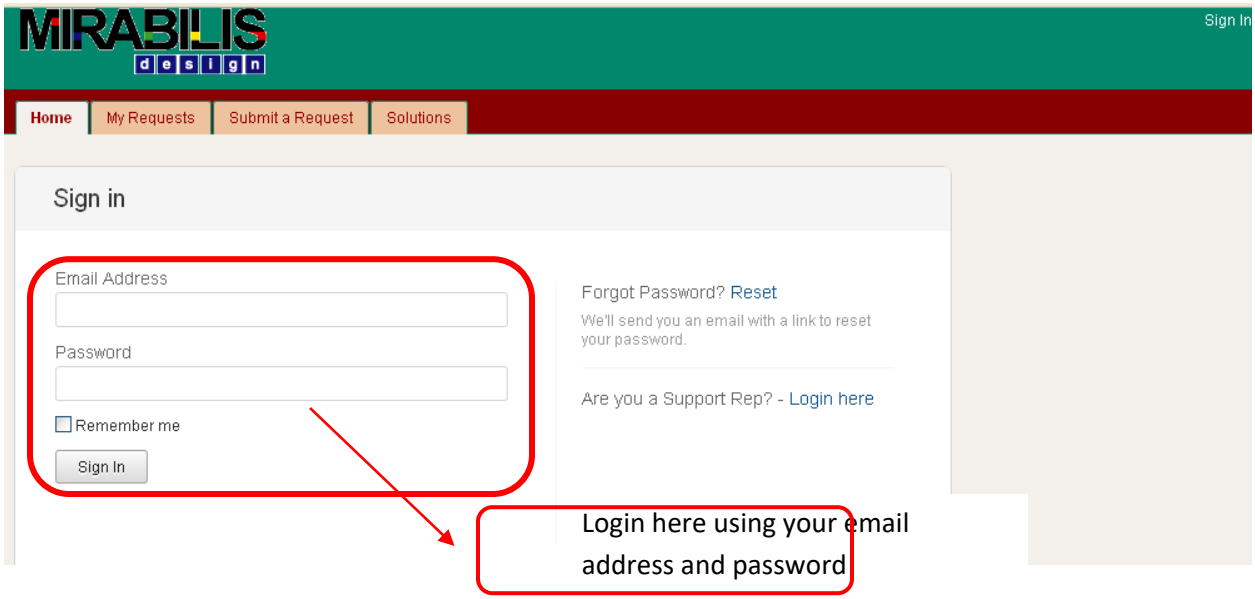
b. After accepting the invitation, you will be redirected to a page where you should create a password for your account.

Submitting a new Request via Customer Self Support Portal

c. Go to www.mirabilisdesign.com, VisualSim Cloud → Support. Click on Support (https://desk.zoho.com/portal/mirabilisdesign/home)

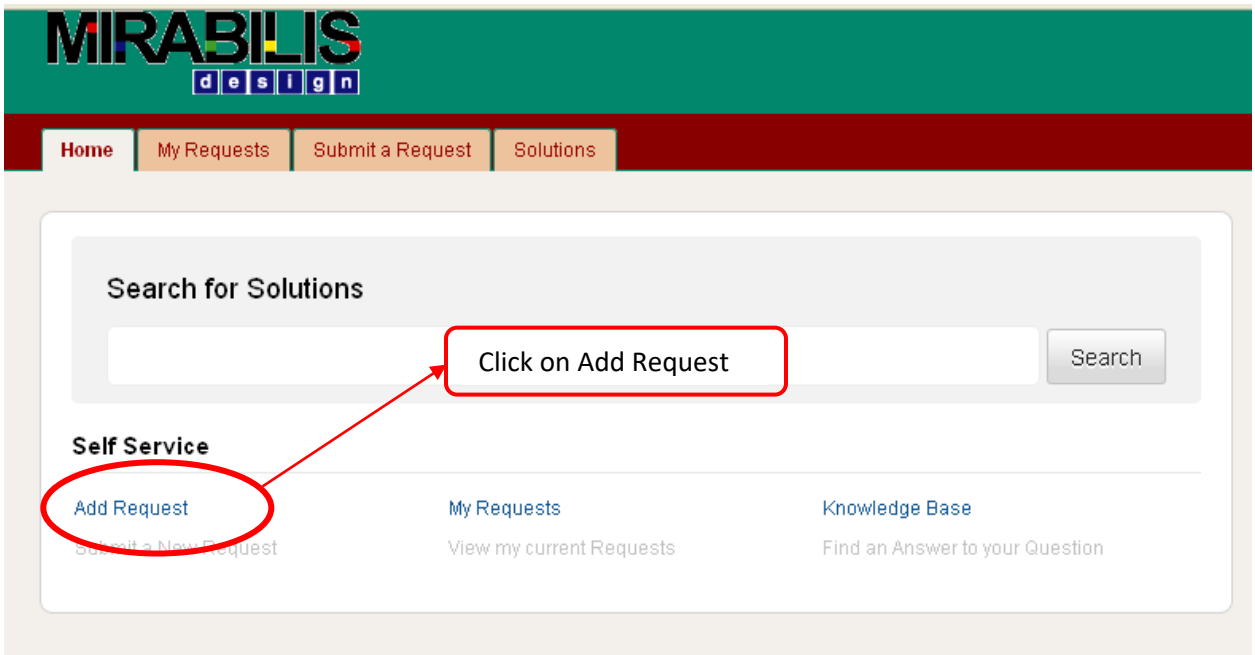


Click on "**Sign In".**

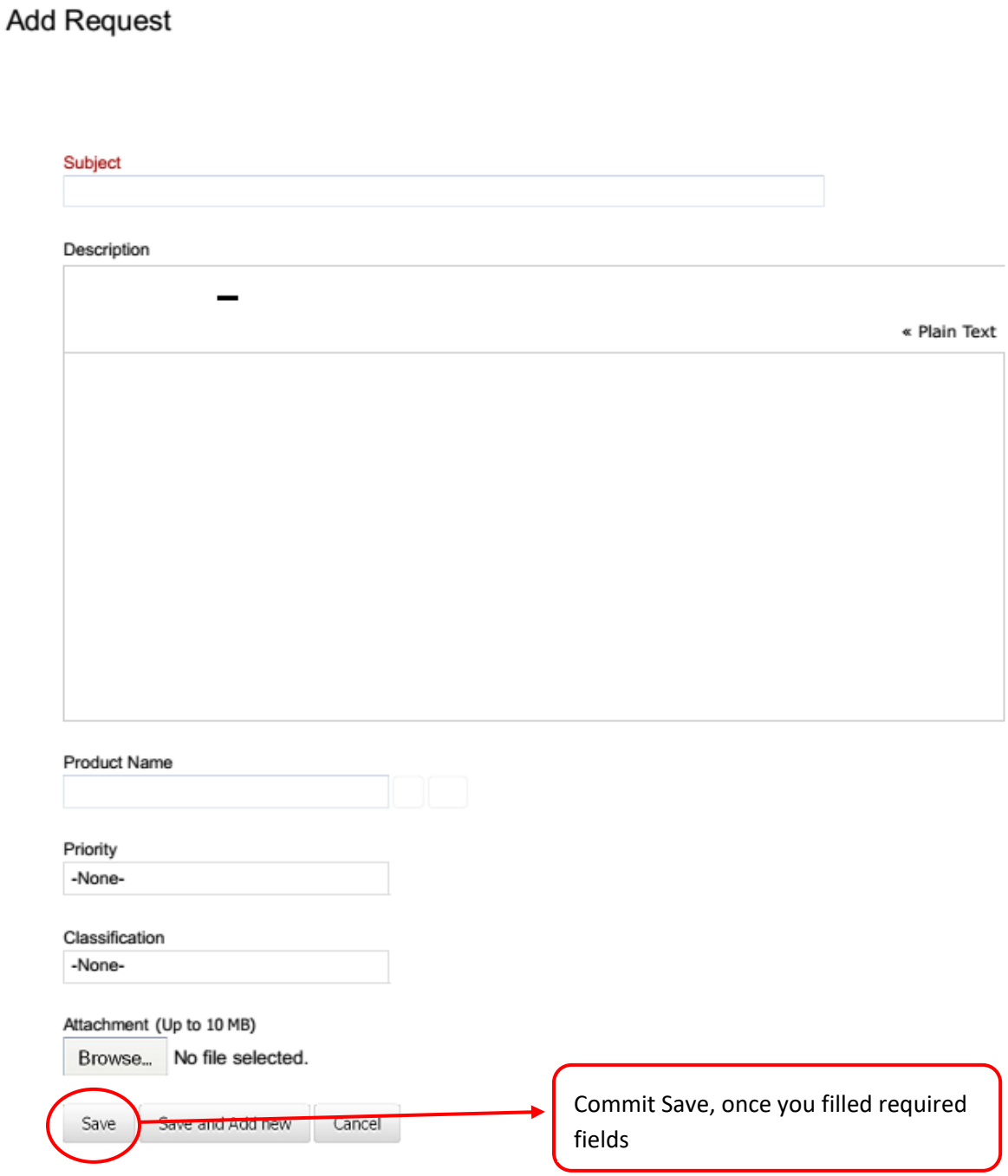


Enter your **Email address** and **Password**.

d. After login to Support Portal, To create a new Request, click on **Add Request**,

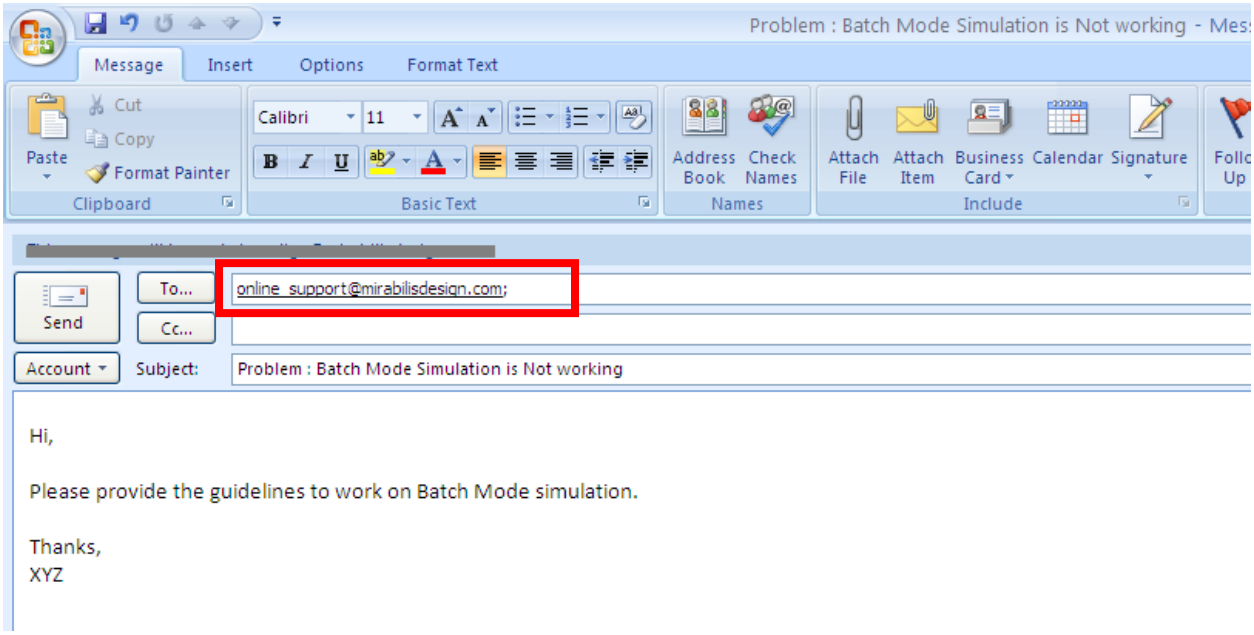


e. Enter the details and relevant files to add request.



Submitting a new Request via Email

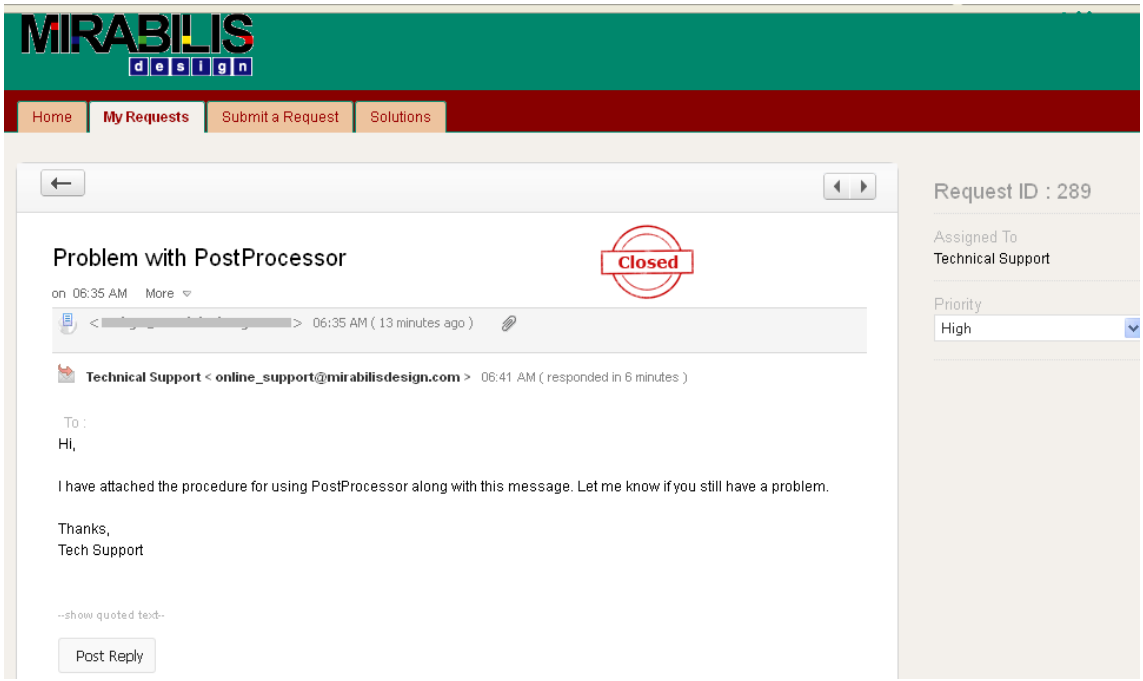
a. You can send Requests via email to online_support@mirabilisdesign.com

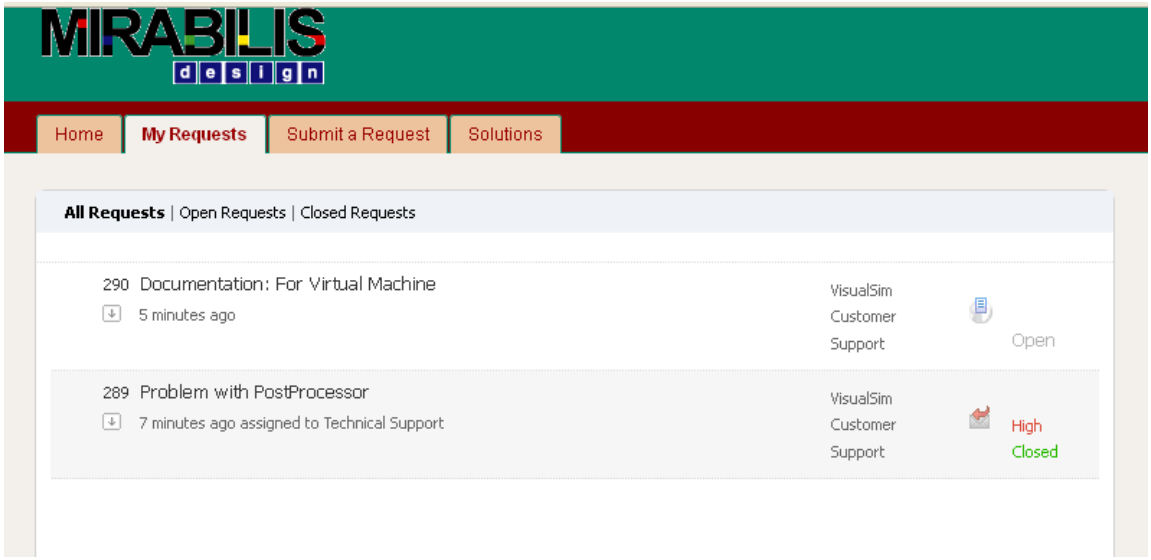


Your email requests will be updated in your Support Portal account

Managing Requests/queries on Support Portal

To view your Requests, click on “My Requests”, "Open Requests", or "Closed Requests".

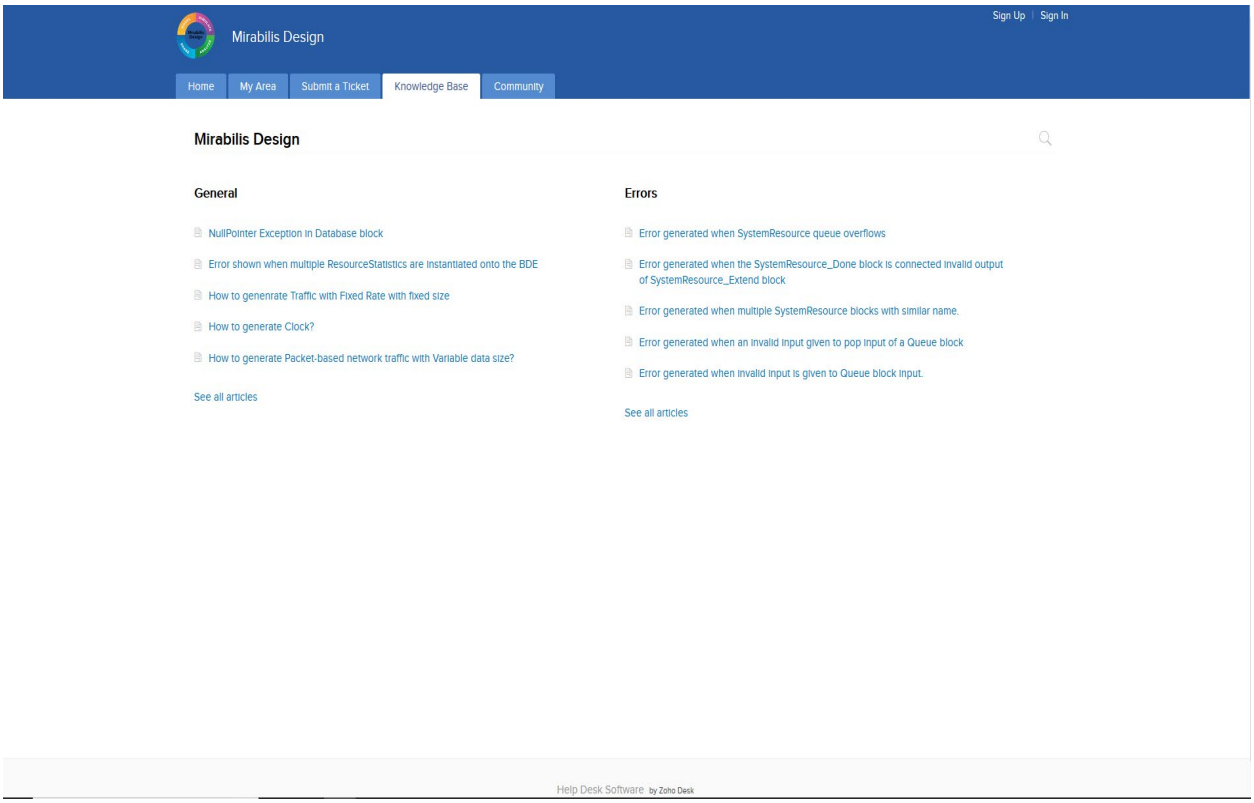




You can reply to the response from Mirabilis Design Support team by selecting "**Post Reply"**

FAQ's and Solutions

Select the "Solutions" tab to view some of the solution. This is a growing section and currently does not cover all the answers.


Mirabilis Design
Sign Up | Sign In
Home
My Area
Submit a Ticket
Knowledge Base
Community
Mirabilis Design
General
NullPointer Exception in Database block
Error shown when multiple ResourceStatistics are instantiated onto the BDE
How to genenrate Traffic with Fixed Rate with fixed size
How to generate Clock?
How to generate Packet-based network traffic with Variable data size?
See all articles
Errors
Error generated when SystemResource queue overflows
Error generated when the SystemResource_Done block is connected invalid output of SystemResource_Extend block
Error generated when multiple SystemResource blocks with similar name.
Error generated when an invalid input given to pop input of a Queue block
Error generated when invalid input is given to Queue block input.
See all articles
Help Desk Software by Zoho Desk

12. Appendix

1. Modeling is the act of representing a system or subsystem formally using modeling IP blocks, Custom-code or scripts.

2. Design is the act of defining a system or subsystem. Usually this involves defining one or more models of the system and refining the models until the desired functionality is obtained within a set of constraints.

3. Model is the mathematical representation of the stimulus, description of the actual or theoretical physical system and the viewers for the output. A model will consist of traffic or input, system or sub-system and analysis or results.

4. Simulation is the discipline of executing the model on a digital computer, and analyzing the execution output. Simulation embodies the principle of ``learning by doing''. To learn about the system, we must first build a model of some sort and then operate the model.

5. Traffic is the test values that represent the flow of data through the system. In a real system, this can represent video frames, signals through pins, network packets and control messages in hardware or software.

6. System is the physical entity that requires to be analyzed. A system can be a computer, SoC, Processor, network, FPGA or any combination of the above.

7. Abstraction represents the type and detail of the information used to describe the system. The abstraction closely aligns the modeling question. The model can be described as un-timed algorithmic, timed performance or queuing, cycle-approximate architecture or control theory. For example, a processor can be described as a queue + processing time, scheduler or a pipeline-based system. A network can be described with latency for the links and a scheduler for the processing nodes.

8. Model Question is the purpose of the simulation and will determine the model abstraction.

9. Analyses are the output of the simulation that will provide sufficient information to make decisions and to answer the modeling questions.

10. Methodology: A top-down design may be a new design that can start with a “blank piece of paper” or one that focuses on key algorithms in a new way. Whereas a bottom-up design may focus on incremental improvements to a successful design, that

maintains the same internal structure, yet may improve the individual elements of the conceptual block diagram. The best modeling approach is to use a combination of top-down and bottom-up design in the same system. This can result in a higher quality design, since concepts and details are given equal weight from day one.

11. Hierarchy is a modeling method to hide the details of a particular component in the graphical editor. Hierarchical modeling is a flexible and organized method to create larger and reusable models. This approach allows for easy understanding and replacing with instances from another simulator or containing different amount of detail.

12. Modeling Topologies is getting closer to creating the conceptual block diagram in VisualSim. Typically, one adds a hierarchical block for each block in the conceptual block diagram and adds input and output ports to match the flow in conceptual block diagram. It is here that the simulator, traffic, analysis, and parameters can be considered for a performance model.

13. Links are used to connect blocks and describe the flow of data through a model. A link is created by drawing a line from the output port of one block the input port of another block. Certain blocks allow multiple connections to and from the ports. These ports contain a white port icon. To connect a Hierarchical block port to the port of a block or to connect a port to a relation, click on control and then drag a line between the ports and relations. The links indicate the flow of data through the model, i.e. which block will execute next. No timing or value change is associated with links.

14. Relations are used to propagate values from one port to multiple ports or to present values from multiple ports to a single input port. If multiple connections are required from an input port or to an output port, a black diamond called relation is required. This is placed on the BDE by clicking (not dragging) on the icon. The relation will be placed in the middle of the screen. Move this icon to an appropriate location. Multiple relations can be connected together to provide wire routing, as well. To connect a relation to a port, click on control and then drag a line from the relation to the port. To automatically place a relation at a point in the BDE, Select down on Control + mouse click at the required position.

15. Parameters can be added to any model and at any level of hierarchy. These are added to the BDE by selecting the parameter from the Library Folder at Model->Parameters. Parameters are a convenient way to modify fixed model attributes to create different simulation runs and conduct different scenarios. A parameter can be used to define the

number of input queues, processor speed, input rate or the number of statistics bins. Parameters can be linked to the equivalent parameters in the blocks of the same level by using the parameter name within the block parameter. Parameters can be expressions containing other parameter names. Parameters are constant during the simulation. They will be evaluated at the start of a simulation and will be constant for the duration of that simulation.

16. Hierarchical block is used to encapsulate a number of blocks of the same modeling domain, or different domains. To create a hierarchical block, drags "Hierarchical Block" from the Library Folder Model->Basic Blocks into an existing BDE window. To view the inside of the Hierarchical block, right-click on the icon and select “Open Block”. Select the input and output button Icons to add input and output ports to a hierarchical block. These will show as input/output ports at the parent level model.