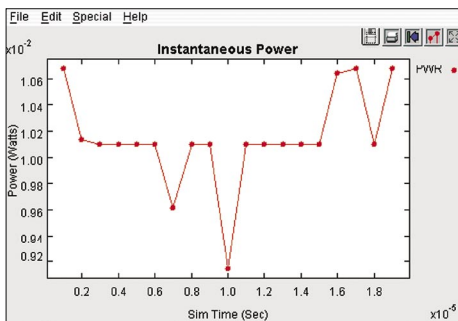
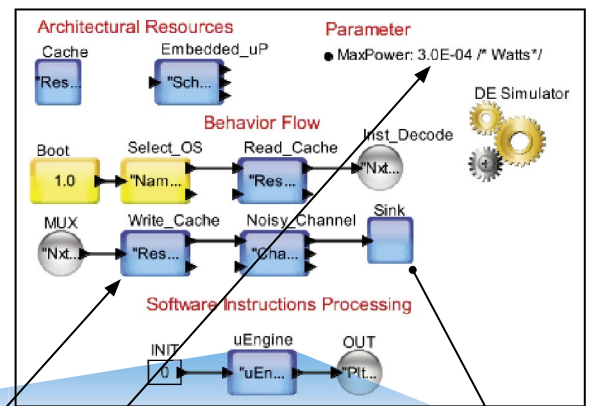


Custom block authoring language implemented as a Micro (μ) Engine



Power Consumption for a Software Sequence as defined in the uEngine

INSTR	OP_X	OP	OP_Y	OP_Z		
DEF	Mem_A	local	MaxPower		;	/* Model Parameter */
DEF	Mem_B	local	Power_DS		;	/* Model Data Structure */
DEF	Sched	global	ref		;	/* Scheduler Reference */
LBL	Begin				;	
WHL	Mem_E		<= Mem_A	Max Power_Case	;	/* Power Available? */
ACT	Mem_D		= rand(1.0E-06, 1.0E-05)		;	/* Incremental Power */
ACT	Mem_E		+= Mem_D		;	/* Cumulative Power */
ACT	none		wait 1.0E-07		;	/* Software Delay */
ACT	Sched		= schedule(Mem_D, Mem_B.ID)		;	/* CacheRequest */
RTN					;	
LBL	Max_Power_Case				;	
ACT	Mem_B.Power		= Mem_E		;	/* Set DS Value */
ACT	Mem_B.TIME		= TNow		;	/* Set DS Value */
ACT	output		send Mem_B		;	/* Send Value to Output */
END					;	

Key Features

- ◆ Modeling-specific language for Rapid prototyping of software instructions, algorithms and resource processing
- ◆ Create complex models and explore critical conditions using a small, efficient, compiled instruction set
- ◆ Define multiple levels of abstraction and control model resources for extensive ad-hoc exploration
- ◆ Instructions and syntax use conventions common to hardware and software engineering

SmartMachine is a modeling-specific Turing Machine that executes a script within a model block. The SmartMachine combines the power of custom-code with graphical modeling and create a highly productive modeling paradigm for solving complex problems. The advantages of this approach are ease-of-learning and use, native code speed simulation performance, small code footprint and language flexibility. Modelers can author custom blocks for reuse elsewhere.

Application

SmartMachine serves three major purposes: significantly accelerate processing-intensive tasks, detailed refinement of architecture, and consolidate multiple functional blocks. This can be used to model hardware, embedded software and complex traffic profiles using common instructions and syntax. The SmartMachine is used for power analysis, functional validation, software estimation, RTOS evaluation and hardware description of complex systems such as switch fabrics, distributed or programmable processors and graphic engine controllers.

Technology

The solution consists of the μ Engine™ execution block, Just-In-Time (JIT) compiler and 10 micro instructions. The script is constructed using the instructions, Expression Language and incoming Data Structures. SmartMachine instructions include Define, Action, Jump if True, Jump if False, While-Do, Goto, Subroutine, Goto with thread, Return, End and Label. The script, extends the Expression Language, to describe functional and performance details of algorithms, implementations and sub-systems. The JIT compiler creates an optimized execution classes of the script that is executed at run-time.

The μ Engines can communicate with graphical blocks through unrestricted number of ports and virtual connections. The script supports sixteen threads enable multi-threaded operations and concurrent event processing. The script can access queues, memories, virtual resources, schedulers and other VisualSim Resources. The block has built-in debugging, code tracing and syntax checking.

Benefits

- ◆ SmartMachine-based models can be executed at one to four million instructions per second
- ◆ Create significantly large models by combining a few graphical resource blocks and a small number of script lines
- ◆ SmartMachine encourages modularity and leverages the extensive SmartBlocks library
- ◆ Eliminate syntactical errors and reduce debug time with built-in assertions and verbose debugging
- ◆ Reduced instruction set enables users to learn and create a script in a few hours