A Concurrent Real-Time White Paper

2881 Gateway Drive
Pompano Beach, FL 33069
(954) 974-1700
https://www.concurrent.com/real-time/

Targeting Multi-core Systems from Multi-rate Simulink Models for Hardware-in-the-loop Simulations

By: Anish Anthony, Ph.D.

Principal Engineer, Real-Time Systems

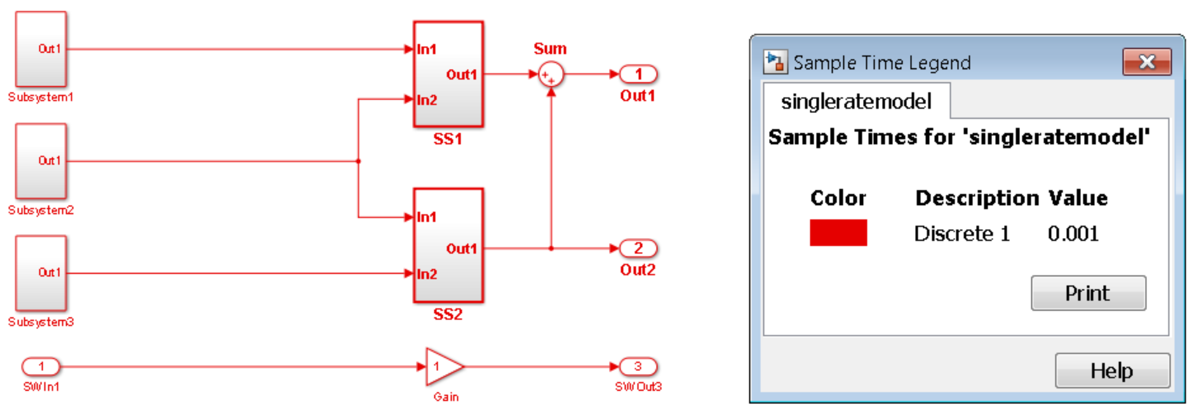
June 2013

Introduction

Simulink™ is one of the most popular tools for multi-domain simulation and Model-Based design. Not only does Simulink provide the users with the ability to exercise the model in a non-real-time environment but also the code generated from Simulink is often targeted towards real-time, rapid prototyping, hardware-in-the-loop, and embedded systems. As more powerful computers become available, engineers are constantly pushing the boundaries by developing exceptionally large, high-fidelity models of control laws and physical systems. Though, non-real time simulation provides a reasonable understanding of the system, the tremendous advantages provided by the real-time testing requires these large high- fidelity models to execute deterministically. However, the size and complexity of some of these models prevents them from executing in real-time. Especially, the dynamics and non-linearity's involved in the modelling of physical systems requires finding a combination of model complexity, solver type, solver settings, and hardware that permits execution in real-time and delivers results sufficiently close to the results obtained from desktop simulation. This paper covers how multi-rate Simulink models are leveraged by real-time simulation tools to target complex and high-fidelity Simulink models for real-time execution on multi-core platforms. Concurrent's RedHawk Linux and SIMulation Workbench™ real-time platform leverages these capabilities of Simulink with demonstrated abilities to execute large, complex, and high-fidelity multi-rate models on multi-core systems in real-time.

Multi-rate Simulink models

For a single-rate Simulink model, the fundamental sample time dictates the fastest rate at which all the blocks in the model will executed. For large, complex, and high-fidelity models, real-time execution at such a fast rates may not be possible especially when wanting to run in the microseconds range. Fig. 1 shows an example single-rate model with a fundamental sample time of 1000us. Attempting to run this model in real-time causes overruns because the model requires on an average at least 1237us to execute as show in Fig. 2.



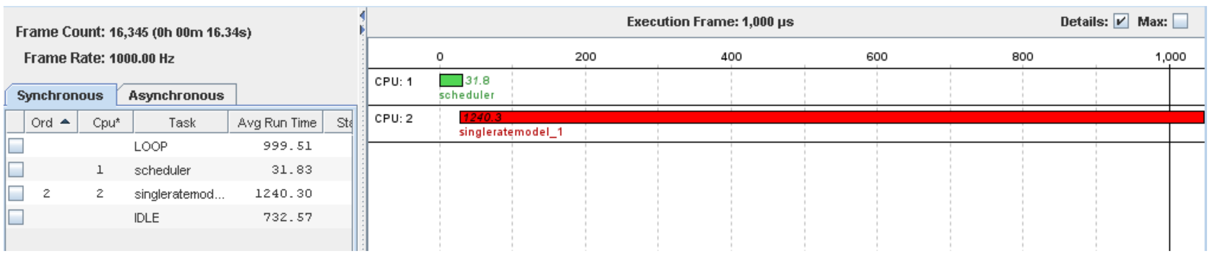
Sample Time Legend

singleratemodel

Sample Times for 'singleratemodel'

Color	Description	Value
🟥	Discrete 1	0.001

Fig. 1: Example single rate model with a sample time of 1000us.


Frame Count: 16,345 (0h 00m 16.34s)
Frame Rate: 1000.00 Hz

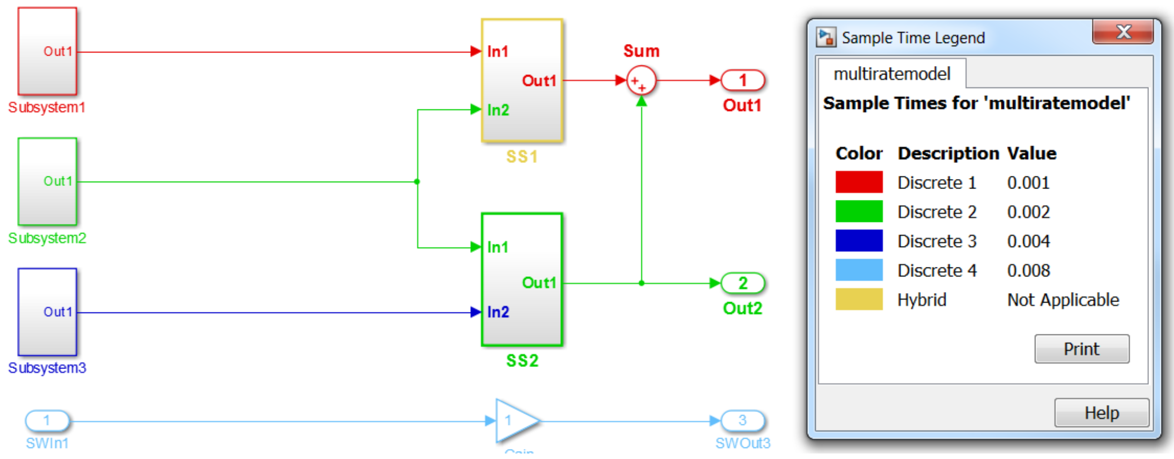
Synchronous | Asynchronous

	Ord	Cpu*	Task	Avg Run Time	Sta
☐			LOOP	999.51	
☐		1	scheduler	31.83	
☐	2	2	singleratemod...	1240.30	
☐			IDLE	732.57	

Execution Frame: 1,000 µs   Details: ☑ Max: ☐

Fig. 2: Model requires more than the allotted time of 1000us to execute causing an overrun.

This common overrun problem in HIL simulation can be remedied by the fact that large complex models typically have a large disparity between the characteristic speeds of different components of the model. One way is to adjust the stiffness of the model by eliminating faster dynamics of the model in favour of slower dynamics thereby decreasing the model fidelity in order to execute in real-time. Another option is to split the models into their different integration rates and providing more system resources to the faster rates giving the model a possibility to execute in real-time. Not wanting to compromise on model fidelity, the second option is typically employed and the model is split into different rates as shows in Fig. 3.



Sample Times for 'multiratemodel'

Color	Description	Value
🟥	Discrete 1	0.001
🟩	Discrete 2	0.002
🟦	Discrete 3	0.004
🟦	Discrete 4	0.008
🟨	Hybrid	Not Applicable

Fig. 3: A multi-rate Simulink model with a fundamental sample time of 1000us.

Simulink provides the ability to run this multi-rate model in single tasking mode. All the rates in the Simulink model are grouped under the same task. Internal counters in the task determine which rates to execute at which sample times. Real-time execution of a single-tasking system requires a base sample rate that is long enough to execute one step through the entire model. Assuming each sub-rate in the model requires less than 150us for execution, one step through the entire model requires 600us which is still less than the base rate of 1000us. Thus, single-tasking mode works in this example as shown in Fig. 4 but can result in inefficient use of available CPU time.

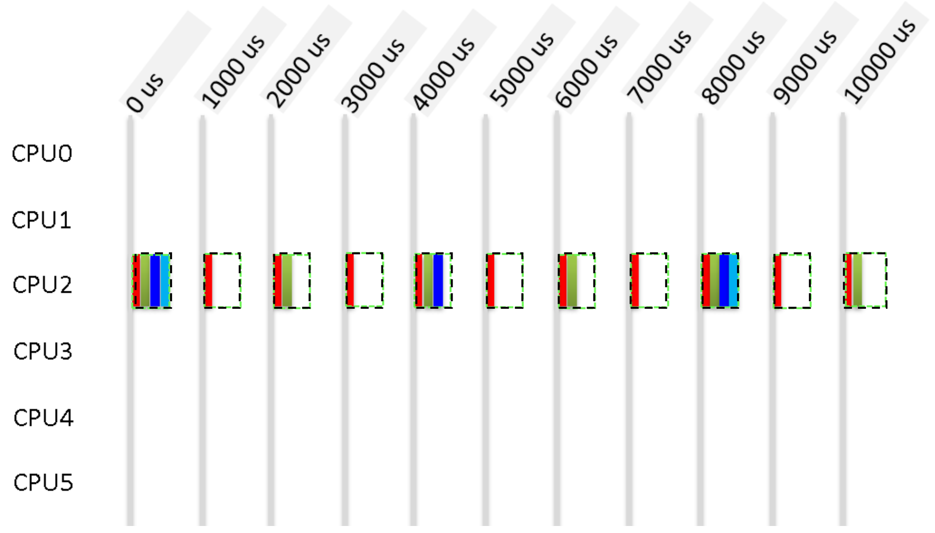


Fig. 4: Multi-rate model in single tasking mode on a single CPU

However, real-time execution of a single-tasking system results in CPU overruns if it takes longer than the base rate to execute one step through the entire model. Assume the Discrete 3 sub-rate requires 2000us for execution for certain inputs to the model. To continue execution in single-tasking mode, the base rate has to be increased to be greater than 2000us+150*3us or the execution will result in CPU overrun. Fig. 5 shows how a base rate of 4000us seconds is used to avoid CPU overruns.

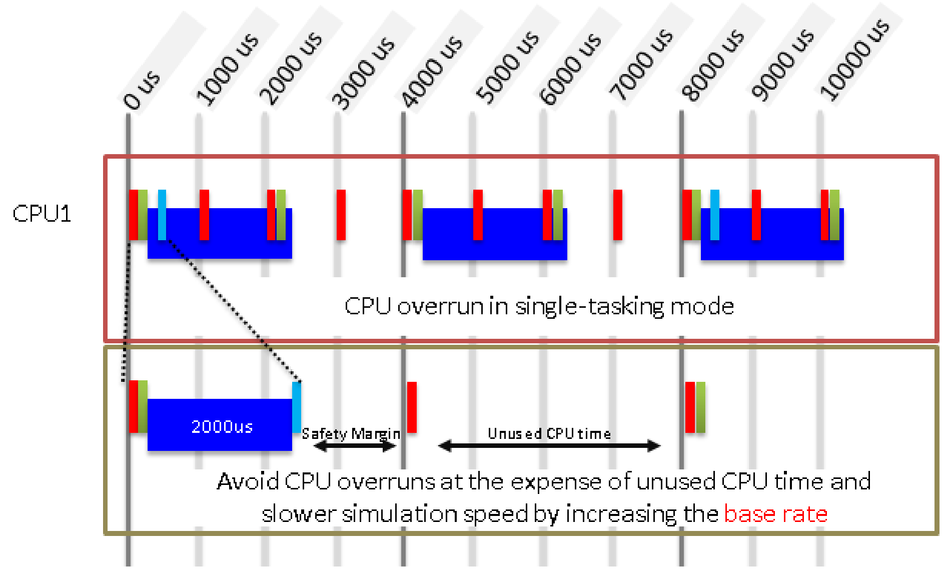

0 us
1000 us
2000 us
3000 us
4000 us
5000 us
6000 us
7000 us
8000 us
9000 us
10000 us
CPU1
CPU overrun in single-tasking mode
2000us
Safety Margin
Unused CPU time
Avoid CPU overruns at the expense of unused CPU time and slower simulation speed by increasing the base rate

Fig. 5: In single tasking mode, increasing the base rate to handle large unpredictable rate execution times is the only option to avoid CPU overruns.

However, Simulink also provides the ability to run a multi-rate mode in multi-tasking mode. In multi-tasking mode, different sample rates in the model are executed using a prioritized, preemptive multitasking scheme. In multi-tasking mode, each rate in the Simulink model is grouped into a separate task with faster rate tasks assigned higher priorities. When periodic tasks execute in a multitasking mode, by default the blocks with the fastest sample rates are executed by the task with the highest priority, the next fastest blocks are executed by a task with the next higher priority, and so on. Time available in between the processing of high-priority tasks is used for processing lower priority tasks as shown in Fig. 6. Simulink and Real Time Workshop (Simulink Coder) generate code for multi-rate multi-tasking models to execute only on one CPU.

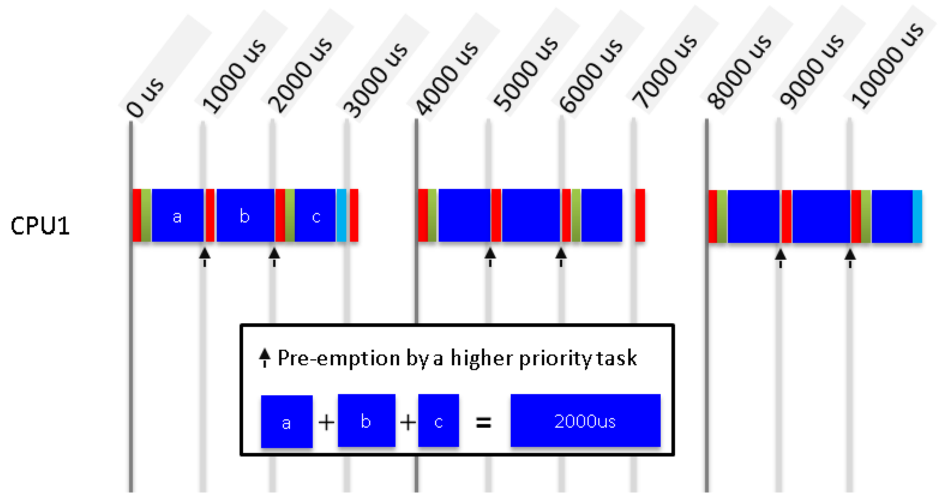


Fig. 6: Multi-rate Simulink model in multi-tasking mode with higher priority tasks preempting lower priority tasks.

When running multiple tasks on the same CPU, the higher priority tasks pre-empt the lower priority tasks. No CPU over run occurs if all the tasks are completed before they are run again. Occasional CPU overruns can occur due to changing task execution times and context switching times resulting in CPU idle times not being enough to complete the pre-empted tasks. If these tasks could be multi-threaded then they could be run of different CPU cores as show in Fig. 7. Running multi-rate tasks on different CPU cores provides an added safety margin for tasks which may occasionally take long to run. For example, the Discrete 3 task shown in blue could possibly take another 2000us before causing an overrun.

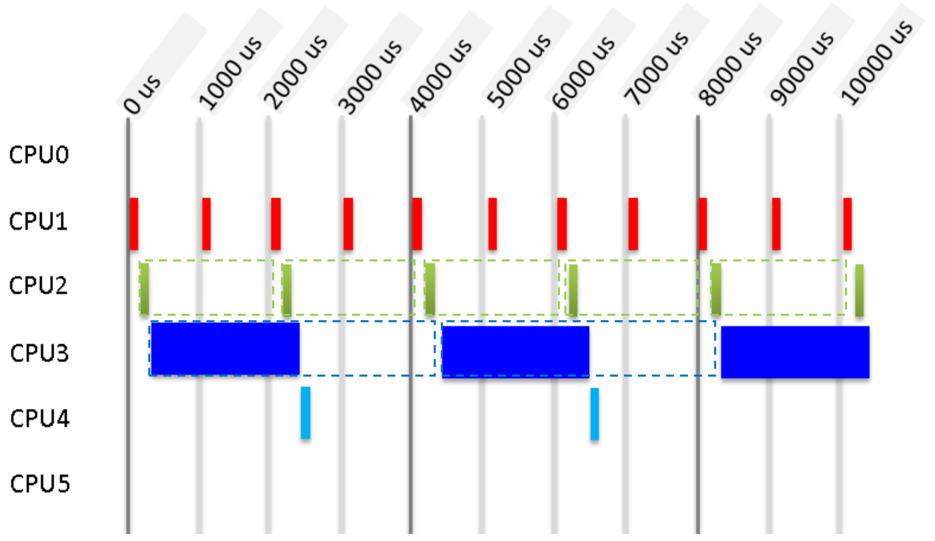


Fig. 6: Multi-rate Simulink model in multi-tasking mode with threads running on different CPU cores.

SIMulation Workbench real-time platform has the ability to automatically generate multi-threaded code from a multi-rate multi-tasking Simulink model and schedule it to execute on multiple CPU cores. SimWB also provides the ability to visualize the task execution time (TET) for different tasks during runtime and reassign the tasks to run on different CPU's for improved CPU load balancing as shown in Fig. 8. Fig. 9 shows an actual screenshot of the SimWB Real-time Viewer Tool showing the execution on the threads on different CPU cores.

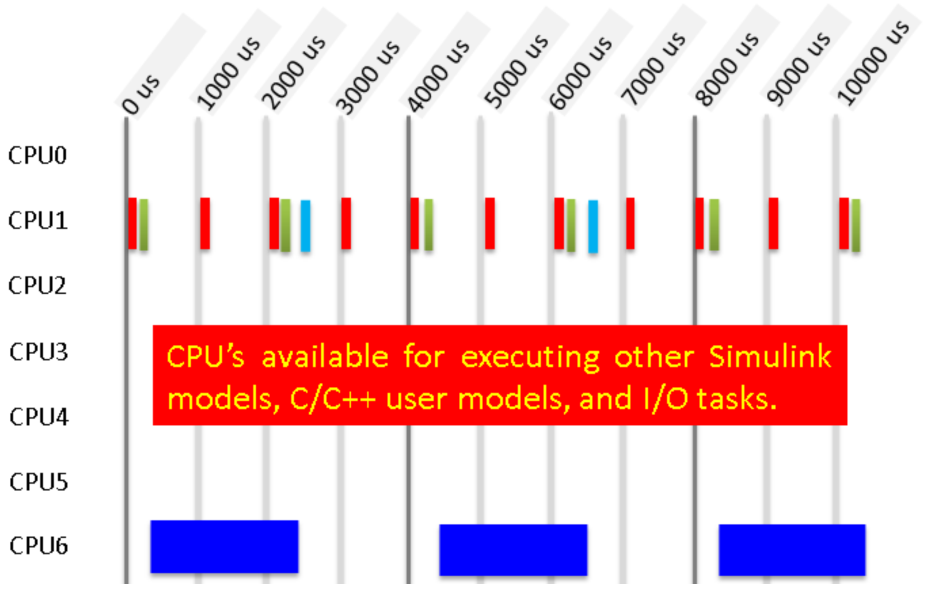
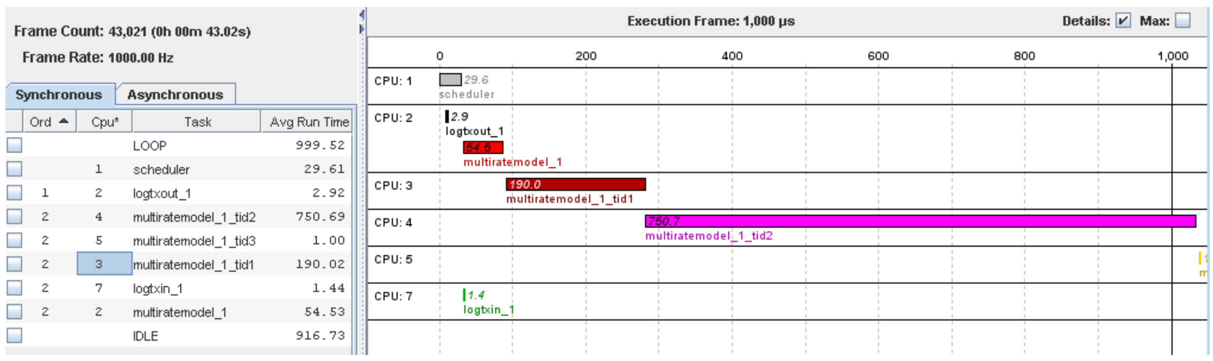


Fig. 8: Reassign multi-rate threads to run on different cores during runtime for improved CPU load balancing.


Frame Count: 43,021 (0h 00m 43.02s)

Frame Rate: 1000.00 Hz

Synchronous | Asynchronous

Ord	Cpu*	Task	Avg Run Time
		LOOP	999.52
	1	scheduler	29.61
1	2	logtxout_1	2.92
2	4	multiratemodel_1_tid2	750.69
2	5	multiratemodel_1_tid3	1.00
2	3	multiratemodel_1_tid1	190.02
2	7	logtxin_1	1.44
2	2	multiratemodel_1	54.53
		IDLE	916.73

Fig. 9: Screenshot of the SimWB Real-Time Viewer Tool showing task execution times of different threads as they execute on different CPU cores in real-time.

Conclusion

Complex, high-fidelity models in Simulink when split into their different integration rates can be provided with more system resources for the faster rates and will execute in real-time in hardware-in-the-loop simulations. The multi-rate, multi-tasking feature in Simulink can be leveraged by real-time simulation tools to target these Simulink models for real-time execution on multi-core platforms. Concurrent's RedHawk Linux and SIMulation Workbench™ real-time platform leverages these capabilities of Simulink and generates SimWB compliant code from Simulink which will allow these different rates to be split into different threads and execute in real-time on different CPU cores.