BLOCK DIAGRAM BASED REAL-TIME SIMULATION ON A NETWORK OF ALPHA PROCESSORS AND C40 DSPS

U. Kiffmeier and M. Beine

This contribution describes the inner mechanisms of the so called Real-Time Interface to Simulink for Multiprocessor systems *(RTI-MP). RTI-MP is applied in the area of Rapid Controller Prototyping and Hardware-in-the-Loop simulations, where high-end computing power is required that can not be realized on a single CPU. Typical users are control engineers who are not too familiar with the details of multiprocessor hardware, distributed real-time kernels, and parallel programming. Therefore RTI-MP is based on an intuitive graphical representation of the multiprocessor system in form of a Simulink block diagram, which is well known to control engineers. The simulation model is implemented fully automatical on a network of DEC Alpha processors and Texas Instruments C40 DSPs.*

2. Simulink block diagrams for multiprocessor systems

Ideally, from the user's viewpoint a block diagram for a multiprocessor system should look as much as possible like a block diagram for a single processor. Of course, the implementation on a hardware with distributed computing nodes requires some additional information to be entered graphically in the block diagram:

- the number and type of computing nodes,
- the assignment of different parts of the block diagram to computing nodes, and
- the signals to be communicated between CPUs and the corresponding transfer protocol.

F. J. Rammig (ed.), *Distributed and Parallel Embedded Systems*
© Springer Science+Business Media New York 1999

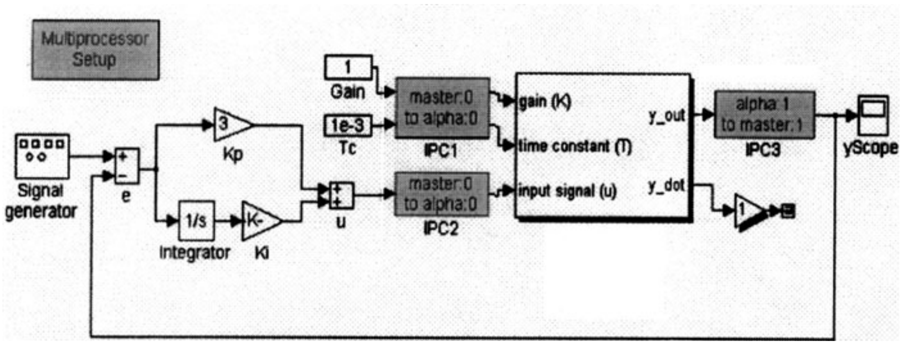


Figure 1. Typical RTI-MP block diagram.

RTI-MP provides the so called *Interprocessor Communication* (IPC) blocks to define this information in a Simulink block diagram. A simple example of a PI controller and a plant model computed on two CPUs is shown in figure 1. The IPC blocks mark the boundaries between CPUs. For example, the block IPC2 transfers the signal u from CPU `master` to CPU `alpha` via communication channel 0. The CPU identity of all other Simulink blocks is determined automatically by RTI-MP from the connected IPC blocks.

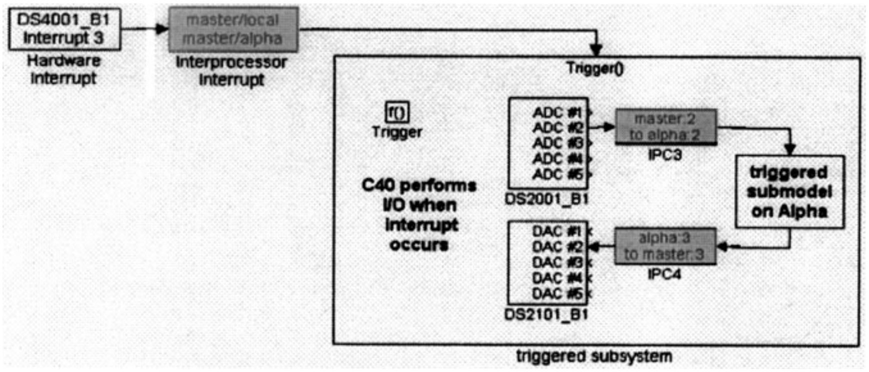


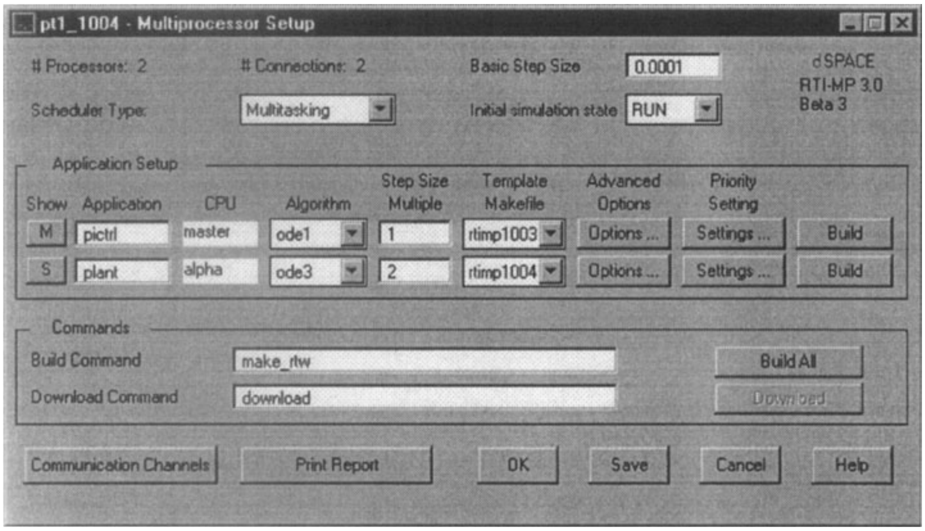
Figure 2. Definition of an interrupt triggered subsystem, which is executed on two CPUs.

Figure 2 shows a triggered subsystem, which is executed each time a trigger event occurs. The trigger event can be an external interrupt from peripheral hardware, or a software interrupt, which is raised when a signal in the block diagram crosses the trigger level. In a distributed system the interrupt source can be located on another computing node than the task to be triggered. It is also possible, that the interrupt task is split over two or more CPUs. For example, the end-of-conversion interrupt from an A/D converter may trigger a task, where the A/D input is read on CPU A and transferred to CPU B, which computes a control algorithm and sends the result back to CPU A for D/A output. RTI-MP supports this with *Interprocessor Interrupt* (IPI) blocks, which define the source and destination CPUs of interrupts. A single interrupt can be sent to multiple destination CPUs, if necessary.

Note, that RTI-MP does not perform an automatic distribution of a Simulink model over a network of processors. It is the user's task to assign submodels to certain CPUs of the hardware system. Experience shows, that the manual partitioning is not too critical in many cases, because

- the assignment of submodels to CPUs often follows directly from physical considerations. For example, in a vehicle dynamics model the front axle is simulated on one CPU and the rear axle on another.
- some submodels require certain I/O resources, which are only available on a specific CPU.

Nevertheless, RTI-MP provides some mechanisms to balance the load in a multiprocessor system, e.g. by allowing to assign individual integration algorithms and step sizes to each CPU (see figure 3). The execution times of all tasks can be monitored in each simulation step. An overload check detects if a task can not be finished before it is triggered the next time.


pt1_1004 - Multiprocessor Setup

Processors: 2 # Connections: 2 Basic Step Size 0.0001

Scheduler Type: Multitasking Initial simulation state RUN

dSPACE RTI-MP 3.0 Beta 3

Application Setup

Show	Application	CPU	Algorithm	Step Size Multiple	Template Makefile	Advanced Options	Priority Setting	
M	pictrl	master	ode1	1	rtimp1003	Options ...	Settings ...	Build
S	plant	alpha	ode3	2	rtimp1004	Options ...	Settings ...	Build

Commands

Build Command: make_rtw

Download Command: download

Figure 3. Main dialog window to configure integration algorithms, step sizes, etc.

3. The hardware system

RTI-MP was designed to implement Simulink block diagrams on a hardware system consisting of a network of DEC Alpha processors and Texas Instruments C40 DSPs as shown in figure 4. Due to the tremendous computing power of theoretically 1000 MFlops per CPU the Alpha processors are viewed as the main computing nodes. Each Alpha is connected via a dual-port memory to a dedicated C40 DSP performing I/O tasks and data transfer to other Alpha/C40 combos. Between C40 CPUs data is transferred via 6 high speed communication ports. The physical distance between the Alpha/C40 nodes can be expanded with special ComPort interfaces up to 100 m. All 6 communication ports operate byte-serial with a maximum transfer rate of 20 MBytes per second in both directions.

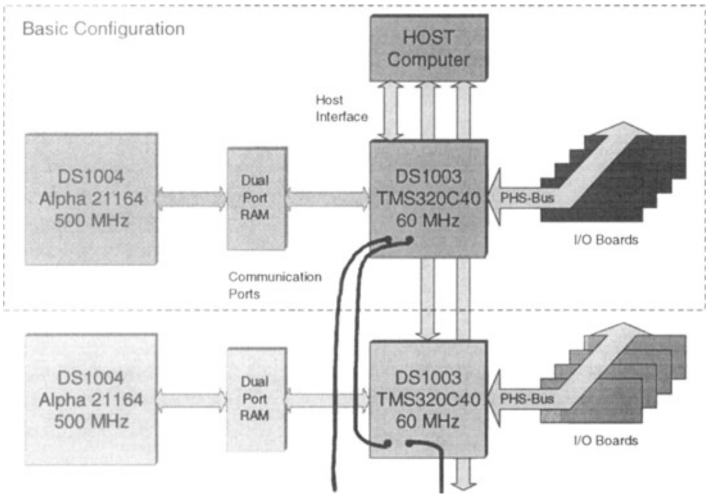


Figure 4. Hardware architecture of the real-time simulator. The PHS bus is used to connect to peripheral I/O boards.

4. Interprocessor communication

For the implementation of Simulink block diagrams on the target hardware described above two transfer protocols are available to send data from one CPU to another:

- The *Virtual Shared Memory* (VSM) protocol (see figure 5) is based on a single transfer buffer, which is written by the sender CPU and read by the receiver CPU independently without synchronization. VSM offers the fastest transfer method by avoiding the overhead for synchronization, but it is well possible that not all data in the transfer buffer results from the same simulation step of the sender at the time it is read by the receiver. It depends on the model dynamics, if this behaviour is acceptable. VSM will mostly be used for slowly changing signals.

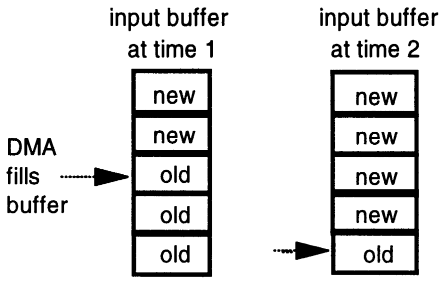


Figure 5. Virtual Shared Memory (VSM) communication protocol.

- The *Swinging Buffer* (SBUF) protocol shown in figure 6 uses a triple buffer system to allow synchronized data transfer and maintain data consistency. When the sender has written all data to one of the buffers, it sets a flag that the data is now available for the receiver CPU. In each simulation step the receiver CPU checks this flag and switches to the latest consistent data buffer available. The sender and receiver CPU never write to / read from the same buffer.

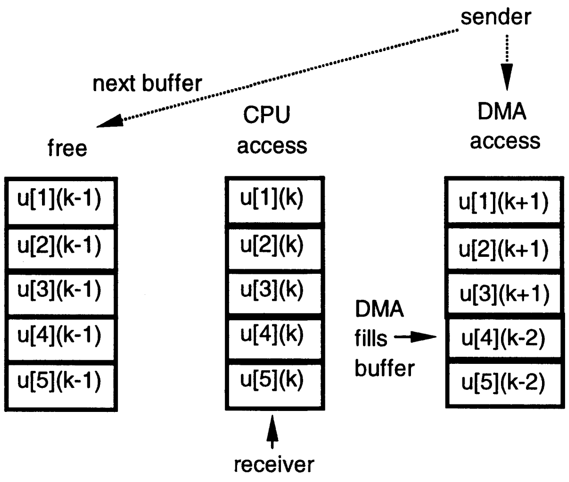


Figure 6. Swinging Buffer (SBUF) communication protocol.

The VSM and SBUF protocols have been implemented for both, the serial data transfer via C40 communication ports and the parallel data transfer via the dual-port memory between C40 and Alpha processors. In the graphical block diagram representation of the simulation model this is fully transparent, i.e. the user needs not to care about the physical type of a communication connection.

5. Interprocessor interrupts

Many control engineering applications require to service asynchronous interrupts. For example, in automotive engine control applications there are typically some periodic time-based tasks and other tasks, which have to be executed based on the crankshaft angle (~ speed) of the engine.

In a multiprocessor system an interrupt occurring on one CPU may trigger actions on other CPUs too. This requires

- the ability to send interprocessor interrupts,
- interruptible, re-entrant communication routines, and
- a priority-based multitasking real-time kernel with low latency running on each CPU.

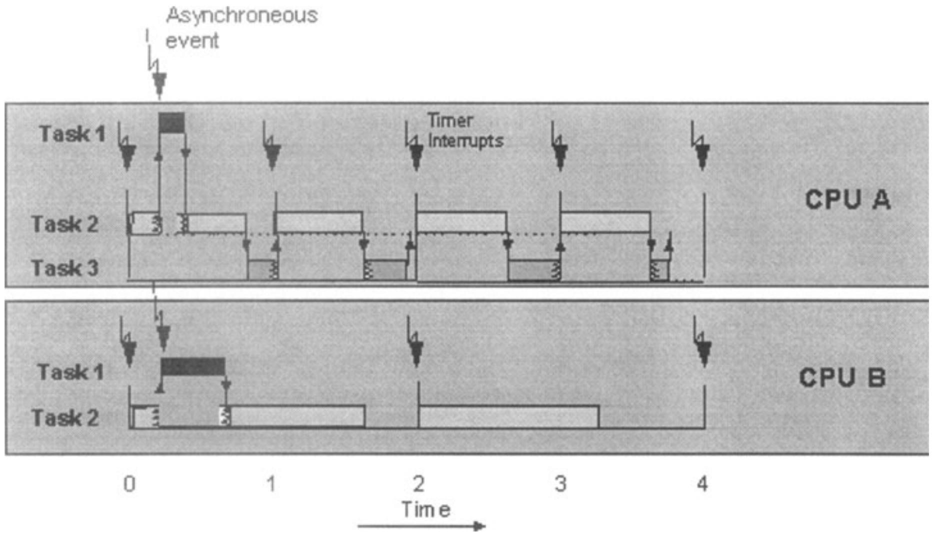


Figure 7. Interprocessor interrupts in a two-processor system. CPU A executes two periodic tasks with 1 ms and 2 ms sample time and CPU B another periodic task with 2 ms sample time. An asynchronous event interrupts the periodic tasks on both CPUs and schedules another high-priority task.

Hardware interrupts between C40 and Alpha processors can be generated by writing a logical interrupt number to a certain memory location in the dual-port memory. When an interrupt occurs, the real-time kernel checks this logical interrupt number and schedules (or queues) the corresponding task depending on the selected priority (see figure 7).

To exchange interrupts between C40 processors a separate communication port line is used. This allows to send interrupts at any time, even if the interrupted task writes a large data block to another communication port at the same time. The DMA coprocessor of the C40 generates an interrupt each time a data word (= logical interrupt number) is received via the interrupt line and the corresponding interrupt handler calls the scheduler to activate the desired task.

6. Real-time simulation frame

In a multiprocessor environment it is especially difficult to perform a coordinated *start* and *stop* of the simulation. Upon simulation start all CPUs must (re-)initialize their internal state variables. If the applications running on each CPU are automatically generated and external I/O is involved, it is very difficult to predict how much time the initialization process will need on each CPU, but for control applications all CPUs in a multiprocessor system must start the simulation *exactly* at the same point in time.

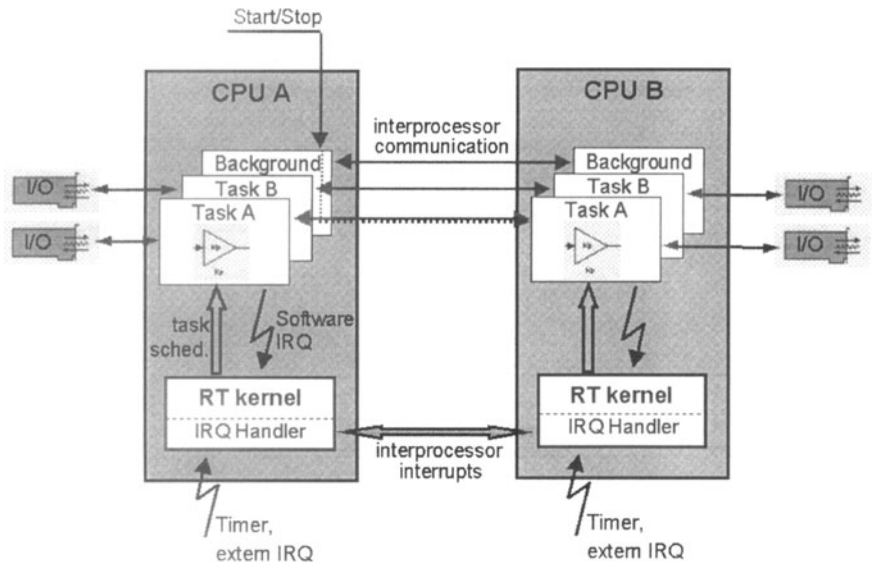


Figure 8. Software architecture of the real-time simulation frame.

With RTI-MP one of the processors is selected as the so called *master CPU*. Only the master CPU can send the signal to start and stop the simulation to other CPUs. In each simulation cycle the desired execution state (= RUN, PAUSE, or STOP) is sent from the master to the slave CPUs. One cycle before the simulation state is actually switched from STOP to RUN, the master CPU sends a command to all slaves to re-initialize their states. This allows to perform a coordinated simulation start in the next step.

Note, that RTI-MP maintains the interprocessor communication even if the simulation is stopped. This is necessary if the user wants to adjust simulation parameters on a distant CPU during STOP state *before* the next simulation run is started. To transfer the new parameter settings to a CPU, which can not be reached from the host directly, interprocessor communication must always be available.

7. Application example

The following industrial application example describes a Hardware-in-the-Loop (HIL) simulation running on two C40 DSPs and two Alpha processors. This simulator was developed by the locomotive manufacturer Adtranz to perform a system integration test of the controller equipment of electric locomotives (Keller et al, 1997). For this purpose all power electronic devices, two asynchronous motors, and a mechanical model of the locomotive had to be simulated.

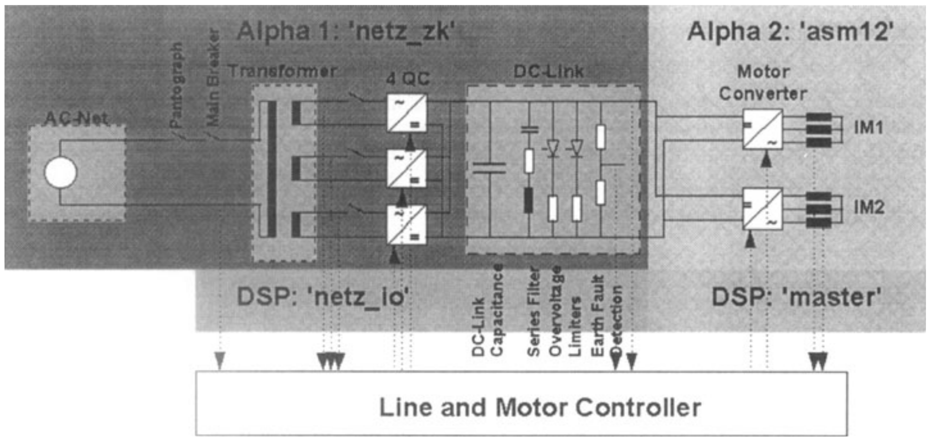


Figure 9. Partitioning of the electrical model on four CPUs.

Modern power electronic switches, like GTOs and IGBTs, which are applied in the line and motor converters of electric locomotives can switch currents with rise times of up to 2 A/μs. To keep the error in the simulated currents minimal, the computational dead-time of the HIL simulator must be as small as possible. Off-line simulations have shown that step sizes of 40 μs or less are required to achieve acceptable simulation results. Even with such small step sizes the error in the simulated currents can be up to 80 A in the worst case, which is about 4 percent of the whole range. Since the desired performance could not be achieved on a single processor, Adtranz decided to implement the HIL simulator on a multiprocessor system with RTI-MP.

Figure 9 shows the partitioning of the electrical model on two C40 DSPs and two Alpha processors. The corresponding RTI-MP block diagram is given in figure 10. The first Alpha computes the line side of the model including three rectifiers and the intermediate DC link. The second Alpha simulates the two motor converters and asynchronous motor models.

Both C40s are completely dedicated to the necessary I/O operations, while the Alphas compute the main simulation model. Not all 84 I/O signals need to be sampled with a step size of 40 µs. To save execution time in the 40 µs task some of the I/O signals are serviced in the background process. The corresponding non-time-critical I/O blocks are placed into the subsystem *background I/O tasks* on the top level of the Simulink block diagram in figure 10.

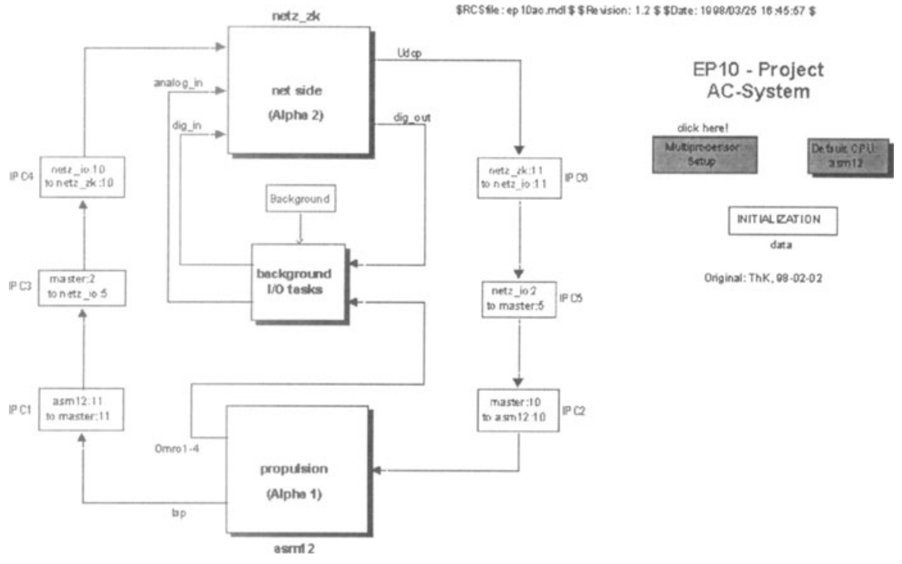


Figure 10. RTI-MP model of the HIL Simulator.

Adtranz has applied the described multiprocessor system successfully to a number of HIL simulation projects. Meanwhile there exist 5 copies of the simulator in different departments of Adtranz. Thus system integration tests by HIL simulation have become a standard part of the development process.

8. Conclusions

RTI-MP provides an intuitive graphical programming of multiprocessor systems based on Simulink block diagrams. All code for I/O operations, communication between CPUs, interprocessor interrupts, and the underlying real-time frame is generated fully automatical. This enables control engineers to apply multiprocessor simulators to very practical problems in domains where big mainframe or expensive analog computers were required before. The distribution of a control application over the available processors is still a remaining task for the user.

References

Kiffmeier, U. (1995). Automatic Code Generation for Multi-DSP Networks on the Basis of Simulink Block Diagrams. EUROSIM Congress '95, Wien, Austria, Sep. 11-15, 1995

Otterbach, R. and Kiffmeier U. (1996). Eine neue Generation hochleistungsfähiger Echtzeitsimulatoren auf der Basis des DEC Alpha Prozessors" ASIM '96, Dresden, Germany, 1996

Kiffmeier, U. (1997). Real-Time Simulation of a 3-D Vehicle Dynamics Model on the DEC Alpha Processor." 15th IMACS World Congress, Berlin, Germany, August 24-29, 1997

Keller, Th., Scheiben, E., and Terwiesch, P. (1997). Digital Real-Time Hardware-in-the-Loop Simulation for Rail Vehicles: A Case Study." European Power Electronics Conference EPE '97, Trondheim, Norwegen. Sep. 7-9, 1997.