# VIRTUAL PROTOTYPING OF EMBEDDED DSP SYSTEMS

Vijay K. Madisetti, T. Egolf, S. Famorzadeh, L-R Dung

Electrical and Computer Engineering
Georgia Tech, Atlanta, GA 30332-0250, USA
vkm@eedsp.gatech.edu

## ABSTRACT

*The Rapid Prototyping of Application Specific Signal Processors (RASSP) program initiated by the Advanced Research Projects Agency (ARPA) has proposed a design process that is based on a new design methodology called Virtual Prototyping, wherein VHDL models of hardware components are integrated with application, control and diagnostic software to rapidly prototype complex embedded DSP systems. This paper discusses this new methodology, and compares the RASSP process with current design practice (circa 1993).*

## 1. INTRODUCTION

The ARPA-sponsored RASSP program is targetted towards the design and prototyping (from concept to product) of large embedded DSP systems. Examples of systems of interest range from efficiently packaged single-board embedded DSP systems (as found in high-performance workstations using MCM-based chassis) to large multi-chassis radar signal processor systems which typically have *performance* requirements ranging between 20-1000 BFLOPs (billions of floating operations per second) of computational intensity at pixel rates of 10 Mhz, within the form *constraints* of size, weight, and power of 2-50 $ft^3$, 100-1400 lbs, and 1-10 KW, respectively. Boards represent subsystems, while multiboard configurations can represent complete systems, and involve hardware fabrication, assembly, and integration with application, control and diagnostic software. Clearly, the RASSP program is of strategic importance to industrial and military competitiveness [2].

RASSP promotes a new design methodology for rapid systems prototyping that differs from current design practice. This difference is first described in a manner so as to introduce a new design methodology for rapid prototyping known as "virtual prototyping".

## 2. CURRENT DESIGN PRACTICE

Figure 1 represents a high level depiction of current design practice (circa 1993) as captured in [4]. The design process flow diagram starts at the level of the representation of the application (algorithmic) requirements. The algorithm to be implemented (e.g., a STAP/SAR radar signal processor system) is specified in an executable form (a

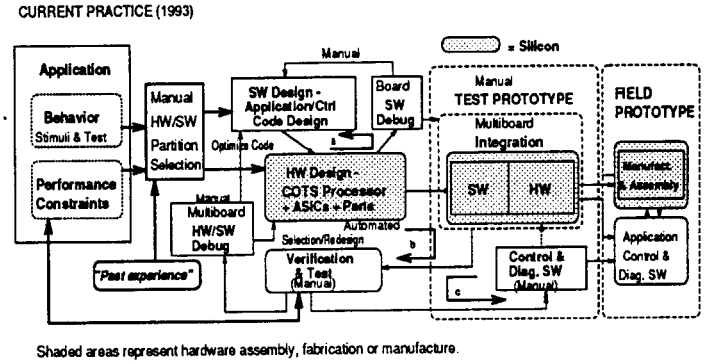


Figure 1: The current practice design model [4] showing dependence between the software and hardware design and development cycles, with *silicon fabrication and assembly/test* (shaded regions) incorporated into three design loops (*a*, *b*, *c*). This interdependence greatly slows down the design and prototyping process, especially if ASICs are to be designed on the critical path.

VHDL/Ada, or a C/Matlab program) together with stimuli and test benches. In addition, the system has certain performance characteristics and constraints that must be met by the prototype (representative values being given in the preceding paragraphs).

After an appraisal of the application characteristics is completed, a partitioning of the application onto hardware (HW) or software (SW) is carried out manually by an experienced hardware system designer, and is to some degree *ad hoc*. Those portions of the systems that are to be cast as ASICs are selected,and commercial-off-the shelf (COTS) components such as processors and memories are chosen as targets for mapping SW components, and inital estimates as to the allocation of these parts are drawn and reviewed. The application is partitioned into subsystems (boards) so that each of these boards executes a portion (in SW or HW) of the algorithm, and their ensemble (the multiboard system) will hopefully prototype the required radar system with satisfactory performance.

Since software (SW) *cannot* execute without target hardware, application and control software developed for each of these boards can *only* be tested and debugged *after* the

Research sponsored by ARPA (ESTO), US Dept of Defense: RASSP contract F33615-94C-1493, 1994-1997.



0-7803-2431-5/95 $4.00 © 1995 IEEE

hardware fabrication (assembly) and test of the board is completed (which can take $2-4$ months per board). After the hardware board is fabricated, application and control code is debugged in an iterative design cycle *a* of Figure 1. After successful design and test of the board-level HW/SW subsystem, the multi-board system is integrated manually, wherein the software and the hardware are merged and tested via diagnostic software and input from the application (stimuli and test). This integration is done manually and involves silicon fabrication, manufacture and assembly/test, and is iteratively refined until an acceptable *test* prototype is synthesized. The three software design loops *a*, *b*, and *c* all include hardware fabrication, and the final *field* prototype is realized after satisfactory integration and test, often involving a total concept to prototyping delay of $3-4$ years, at the cost of $20-30$ man-years. In one representative operational (late 1992) STAP radar signal processor system (studied by these authors), the final HW count was about 150 boards incorporating a total of about 25,000 LSI/COTS components including an ASIC front-end filter and a 64-processor TMS320C30-based processing engine. The final SW count in *lines of source code* (LOSC) was 5K LOSC for the DSP/radar signal processor application, 30K LOSC for control, 60K LOSC for diagnostics, and 25K LOSC for functional and performance verification, representing a 20 : 1 ratio of system-level design code size to DSP application code size [1].

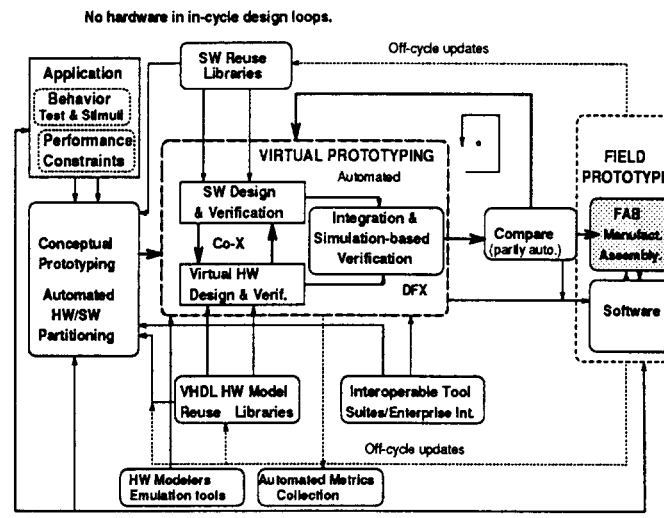


Figure 2: The mature RASSP Design Process (Fall 1997-98) with *hardware-less* in-cycle HW/SW co-design loops, enterprise integration, interoperable tool suites, automated metrics collection, and an additional stage for rapid early algorithm, functional, architectural, and automated HW/SW partitioning — "conceptual prototyping" [4].

## 3. A CANDIDATE RASSP DESIGN FLOW

In a candidate RASSP design flow of Figure 2, the primary difference from Figure 1 is that the hardware fabrication and assembly at the subsystem and system-level is eliminated from the in-cycle design loop [4]. *The software is executed (and designed) on virtual hardware (in the form of VHDL models or hardware modelers and emulators) long before any HW fabrication and assembly is begun.* This **"virtual prototyping"** environment significantly speeds up the HW/SW co-design and co-verification cycle through the use of models at multiple levels of design abstraction in the constituent VHDL libraries. The board-level and the multi-board integration is simulated and tested, additional control and diagnostic software is developed and debugged entirely in a user-friendly software environment. *If the model libraries were accurate, the next stage could itself be that of the field prototype.* However, at least one RASSP prime has planned to include the actual hardware test prototyping stage within the RASSP design process to validate and improve upon the process of virtual prototyping.

In addition to virtual prototyping, Figure 2 introduces an additional stage called **conceptual prototyping** which involves *early* design, replacing the manual HW/SW partitioning block of the "current practice" of Figure 1. Conceptual prototyping utilizes automated tools that allow rapid estimation and evaluation of algorithmic, functional, architectural and enterprise-related tradeoffs early in the design process. A few candidate conceptual prototypes are then culled from the dozen or so generated at this stage, and then passed on to the virtual prototyping stage. Here, extensive evaluation and detailed design is done in virtual hardware and software leading to successful and rapid integration, again through the use of HW/SW reuse libraries, interoperable tools, and enterprise integration.

## 4. VIRTUAL PROTOTYPING

Two types of models are defined for the purposes of this paper — *bus-interface (BIM)* and *fully-functional (FFM)* [3]. The BIM models the timing behavior of a component's interface (with respect) to its surroundings. Thus timing and format of output bus drivers for data, control/ addresses, etc. are modeled as accurately[1] as possible, while internal structure, state, or algorithmic values are not necessarily modeled. Thus BIMs are more suitable for providing rough estimates of system performance during conceptual prototyping. The FFM models *all documented* complexity of a physical component in a VHDL behavioral description. All timing and internal register states are modeled, in addition to full functional emulation of hardware behavior. In the virtual prototyping example of the next section, a FFM of the i860XP (developed at Georgia Tech), and BIMs of the memory, memory controller, FIR chip with buffer, and decoders were used. The board-level system was designed and debugged completely within a software-only environment using a VHDL simulator [2] within four man-days.

[1]While BIMs for SSI and MSI parts are often accurate in terms of timing, complex VLSI parts (i.e., RISC processors, DSP chips, or communication routers) whose timing is both data- and internal state-dependent (cache, interrupts, resource contention, etc) can seldom be accurately modeled by BIMs. Therefore, the authors recommend the use of FFMs for complex VLSI parts.

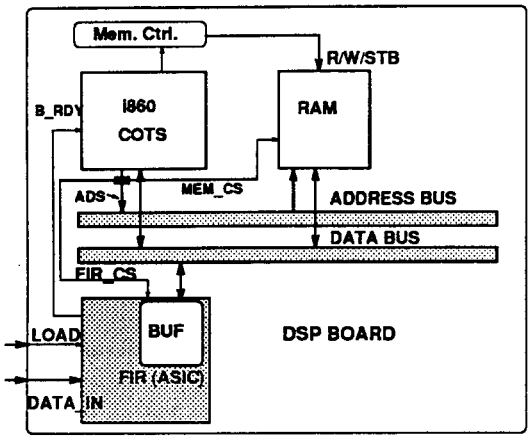


Figure 3: A Board-Level Virtual Prototype

### 4.1. HW/SW Co-Design of a DSP Board

A high-level description of the target system is shown in Figure 3. An equalizer is implemented as a FIR filter. Its design is to be finalized during the virtual prototyping effort. The FIR is loaded with incoming data and stores its output in a 128 word buffer. The i860 [2] should read the data from the FIR's buffer (by selecting its buffer for a read at an appropriate time). The buffer writes the data onto the data bus which loaded by the i860XP into its appropriate internal registers. After processing the data (the design must allow the capability to change this processing software easily at the user's discretion), the i860 selects its local memory (RAM) (and de-selects the FIR's buffer). The memory controller provides the appropriate control signals that allow the i860 to interface with a static RAM. The lower significant address bits are used to store the data in the RAM. The entire cycle then begins again, with the FIR processing new data in the meanwhile.

This example is typical of the functionality required of board level designs, and was chosen to be specific enough to highlight the different trade-offs that are available during the virtual prototyping process.

### 4.2. Model Integration

All the VHDL models (FFM or BIM) were integrated to virtually prototype a DSP board with an-off-the-shelf RISC processor, memory, a custom memory controller, and a FIR filter ASIC with internal buffer. The operations of the virtual prototype (to be verified and evaluated) can be described as follows — the FIR filter receives data from an external source, filters the result and upon request will provide the data to the i860 in a first-in-first-out (FIFO) manner. The i860 processes the data and will store the result back to its local memory. The FIR filter is mapped off the memory space of the i860, and should be capable of directly connecting to the i860 with no extra glue hardware. In order to test the board, some software was written for the i860 as a compiled program that would perform a read operation to the address where the FIR filter was mapped, and then store the result in the local memory immediately. After the load and store operations, the i860 idles for 30 cycles, before starting over. The FIR and the i860 operate at 2.667 MHz and 40 MHz, respectively.

[2] Nothing in the process of virtual prototyping precludes the use of any other COTS processor at any point during the design cycle, as long as its FFM is available.

### 4.3. Timing Verification

Once the models were integrated together, the virtual prototype was simulated and debugged for correct operation.

#### *4.3.1. Proper Timing*

Two snapshots of the signal drivers on our virtual prototype is shown in Figure 4. The timing diagrams show the state of all related signal drivers at the time the i860 is trying to read data from the FIR. The i860 initiates a read by placing the FIR's address on the bus and activating the address strobe signal, *ADS_N*. This can be seen in the timing diagram at time 11227.5 ns. The address lines are fed to a decoder that selects either the memory or the FIR filter by asserting either *FIR_CS* or *MEM_CS*, respectively. After a delay of 5 ns (due to the propagation delay of the decoder), the *FIR_CS* signal is asserted by the decoder at time 11232.5 ns, After an additional delay of 5 ns (due to the propagation delay of the chip select logic in the FIR filter) at time 11237.5 ns the FIR filter begins driving the bus. Valid data is put on the bus by the FIR filter 5 ns after address strobe, *ADS_N*, is deactivated, at time 11257.5 ns. At the same time the *BRDY_N* signal of the i860 is pulled low by the FIR filter to notify the i860 that the data is ready on the bus. The i860 will then read the data on the next rising edge of its clock (or 11266 ns) and the read cycle is completed. The next operation after the read is a write to the memory, where the FIR data is put in the local memory of the i860 (Fig. 4, upper half).

#### *4.3.2. Improper Timing*

A common design error on any bus-based system is that of signal contention, when more than one source is driving the bus. In our virtual prototyping example, the *hold* time of the FIR filter was increased *intentionally* to create a bus conflict. Previously, the FIR chip would release the bus 5 ns after the chip was *deselected* (when FIR_CS was deasserted). In a new experiment, the hold time was increased to 40 ns. With identical application code executing on the i860XP, the processor will try to write the value (just read from the FIR filter) to the local memory. Concurrently (due to the increased hold time) the FIR filter is still driving the bus and a signal contention occurs. These points are also illustrated in the lower part of Figure 4 with the FIR_CS being deasserted at time 11266 ns. Adding a 40 ns hold time of the FIR will result in 11306 ns as the time at which the bus is released by the FIR filter. Concurrently, the i860 tries to perform a write to the memory at time 11277.5 ns as indicated by the *ADS_N* signal going low. The 16 bit data that is to be written to the memory

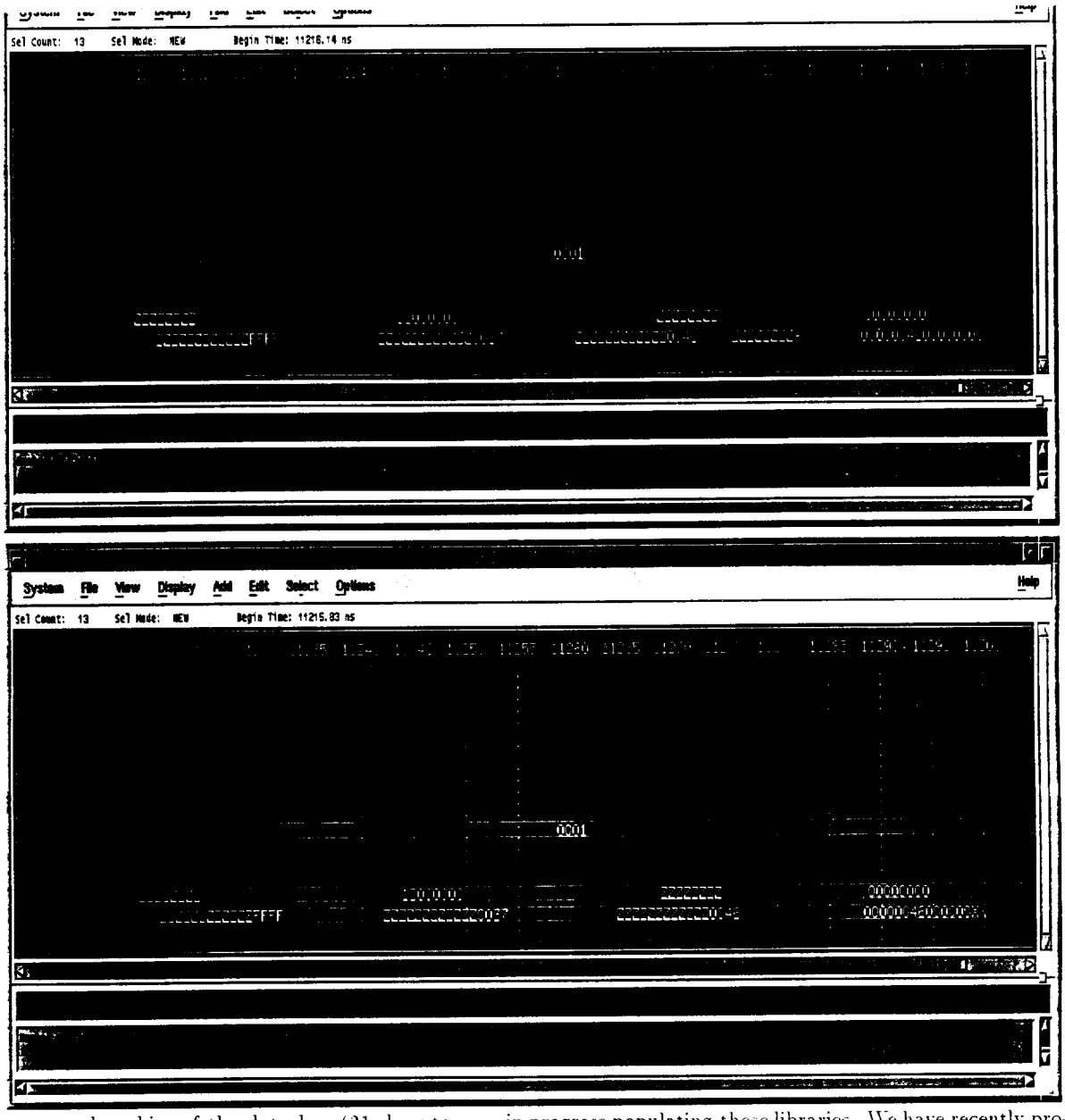


Fig 4:

is put on the upper data bits of the data bus (31 downto 16). At this time the lower 16 bits of the data bus are being driven by two different sources, i860 and FIR chip — one driving it with a value of 0x0000004600000000 and the other with 0x0000000000000046, resulting in the value of 0x00000046000000XX (Fig. 4, lower half).

## 5. CONCLUSIONS

This paper demonstrates how VHDL models of components, whether they be COTS or custom VLSI parts, can be integrated together to design and debug embedded systems in their entirety (with application, control and diagnostics SW) using a new RASSP-based *hardware-less* design methodology, called virtual prototyping. Virtual prototyping relies on the availability of a rich set of VHDL models for components, and a number of RASSP-funded efforts are in progress populating these libraries. We have recently prototyped larger multi-board multiprocessor RASSP systems with encouraging results in collaboration with a RASSP prime.

## REFERENCES

[1]. D. Martinez, J. MacPhee, "Real-time Testbed for Space-Time Adaptive Techniques," Proc. IEEE Adaptive Antenna Systems Symp., November 1994.

[2]. *Proc. of the First RASSP Workshop*, August 1994. Advanced Research Projects Agency (ARPA), US Dept. of Defense, Arlington, VA.

[3]. IEEE 1076-93 Standard VHDL Reference Model, ISBN I-55937-376-8.

[4]. V. Madisetti, "*Vive La Difference*: RASSP vs Current Practice (1993)," *The RASSP Digest*, Vol. 1, No.1, November 1994.