IP devt, FPGA prototyping with SystemC/TLM

By D. Singh, N. Sharma,
V. Upadhvava, A. Hazra,
A. Jain, A. Goel and R. Hakhoo
STMicroelectronics

With the advent of System-on-Chip technology, designs are becoming bigger in size and thus highly complex, time-to-market is becoming critical, and at the same time, RTL methodologies are generally becoming insufficient to fit into this new role. These factors are driving designers to explore new methodologies for early verification of complex IPs (HW as well as SW) as well as complete system.

This article discusses a design flow that starts with highly abstracted models to cycle accurate or RTL models of IP. While moving to lower levels of abstraction, the modelling becomes complex and so does the verification of the IP. Our approach is suited to this scenario because it permits us to run same test benches/test scenarios in similar environments throughout, hence permitting the reuse of all the test cases and environments across the complete development cycle efficiently.

In the semiconductor industry, the first step in product realisation is the development of a prototype of the specification at a higher level of abstraction, commonly in C/C++. Here, SystemC, a C++ library, comes to the rescue. It facilitates a conceptualisation of the coexistence of HW and SW designs together. Along with the TLM transport library, which permits interfacing between transaction-level models, SystemC speeds up the overall verification. Another important aspect is the enhanced portability across differently abstracted architectures. The same test set up can be used with different abstraction levels of design.

This paper discusses one such methodology. The ultimate goal was to design and implement UWB MAC (Medium Access Layer) IP. For the purpose of architectural exploration, the whole IP was to be implemented in SystemC. Various architectures with varying degrees of abstraction level were explored. The overall effort involved was less, simulation speed obtained was good, and the actual SW implementation started early in the design cycle. The RTL Implementation of this IP was ported to FPGA available with the SPEAr (Structured Processor Enhanced Architecture) family. The SPEAr provides, in addition to ARM Core and corresponding set of IP, a configurable logic block that allows an incomparable level of flexibility to the user for implementing his/her logic functions. This shortens time to market and also results in unprecedented cost savings.

Methodology

The methodology, shown in Figure 1 keeps Transaction Level Modelling (TLM) in the core of developments.

TLM is a high-level approach to modelling digital systems where details of communication among modules are separated from the details of the implementation of functional units or of the communication architecture. Communication mechanisms such as busses or FIFOs are modelled as channels, and are presented to modules or components using SystemC interface classes. Transaction requests take place by calling interface functions of these channel models, which encapsulate low-level details of the information exchange.

While modelling at transaction level, there is:

- More emphasis on functionality of the data transfers—what data are transferred to and from what locations
- Less emphasis on their actual implementation—that is, on the actual protocol used for data transfer

This approach makes it easier for the system-level designer to experiment, for example, with different bus architectures (all supporting a common abstract interface) without having to recode models that interact with any of

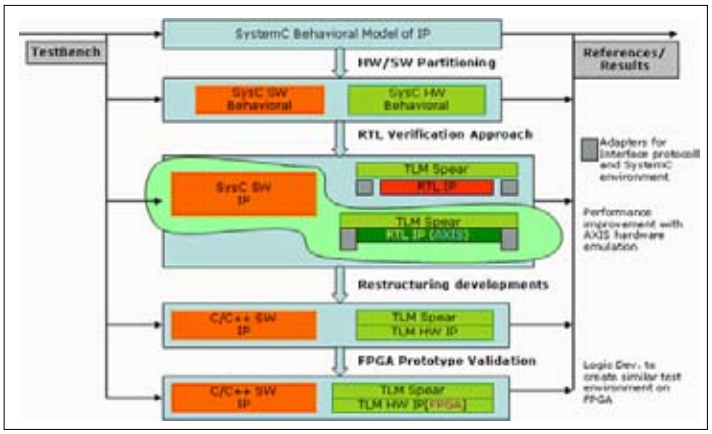

Figure 1: A methodology flow for I{P development.

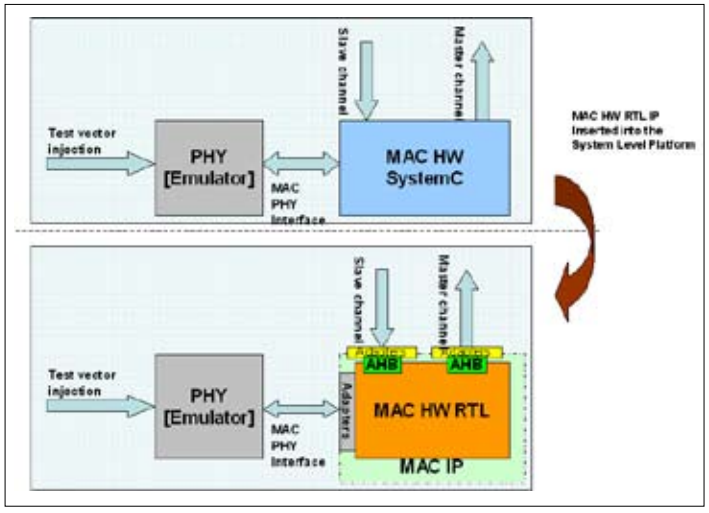

Figure 2: Transition from SystemC MAC HW to VHDL RTL MAC HW adapters.

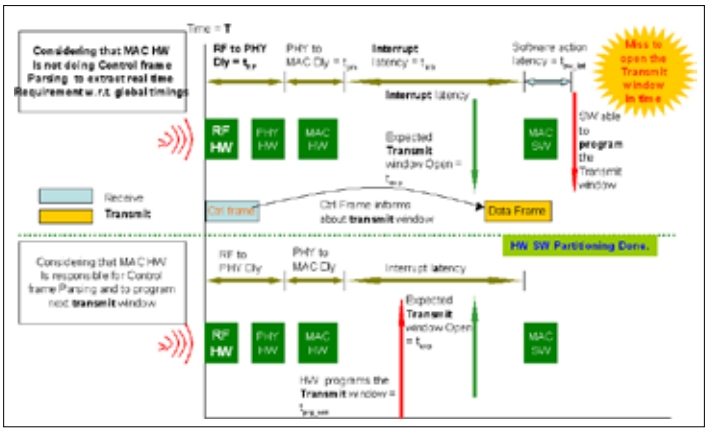

Figure 3: A scenario to highlight HW support needs in a system.

the buses, provided these models interact with the bus though the common interface.

In our methodology, the start point was to model the complete functional system platform. This is done with SystemC using the sc_fifo interfaces. Various structures are used to depict a flow of data between communication interfaces. These structures are basically parameters and frame format information which need to abide by the protocol. A test environment surrounding the IP is created where test benches are developed to flow inputs from either of the sides i.e. respectively for transmit or receive. In both the cases, expected results or references are generated with such a set up. At this level of abstraction, it's easier to play with the platform and make modifications, and to do experimentations quickly and efficiently though accuracy is compromised.

Figure 1 shows the set up platform, which is input for next level of development. The idea here is to identify the bottlenecks of the system and to perform a hardware/software partitioning. This approach is makes it safe to perform HW/SW partitioning as the platform may provide raw statistics that could be used to identify the bottlenecks in the complete system. In this step, the functional model of IP is modelled with detailed interfaces with functionality embedded. A section on HW/SW partitioning will detail the approach deployed in our methodology. Another addition to this platform is a TLM model of DMA-PL080 as HW/SW partitioning analysis suggested the use of an internal DMA controller.

In the next step, the complete MAC HW SystemC functional model is replaced with MAC HW RTL, as shown in Figure 2. The complete surrounding environment is the same, and hence the test injection is the same as it is injected in other steps. The change with previous environments is the use of adapters, which are used for transaction to signal conversion. Since our system was ARM

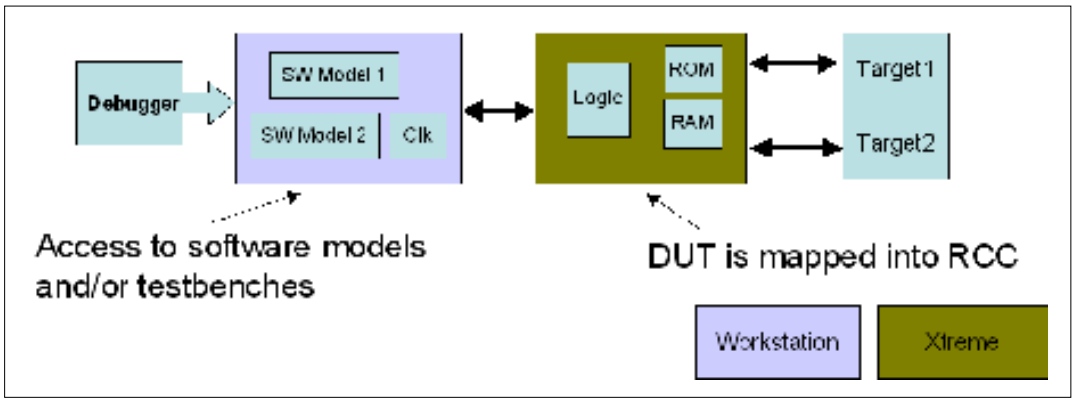

Figure 4: Xtreme server box setup with software.

based, adapters written had to comply with AHB bus interfaces at the signal level. Practically, this platform represented the same environment as a real system, but at the same time, we began facing the problem of simulation performance. Obviously, we were not able to perform extensive debugging/verification with this set up though simpler tests [with small simulation time] were run.

Due to the bottlenecks identified in the current simulation environment, we evaluated the hardware emulation XTREME server based platform, which basically provides FPGA chunks for hardware and provides integration of software with a complete environment. The porting of previous platforms in XTREME server based platform setup gave 5x of simulation speed up with respect to a ncsim based simulation environment. It permitted us to debug and perform verification of VHDL RTL implementation, which would have been too time consuming otherwise. At the same time, the Xtreme server based platform gave the same level of debugging capabilities as well.

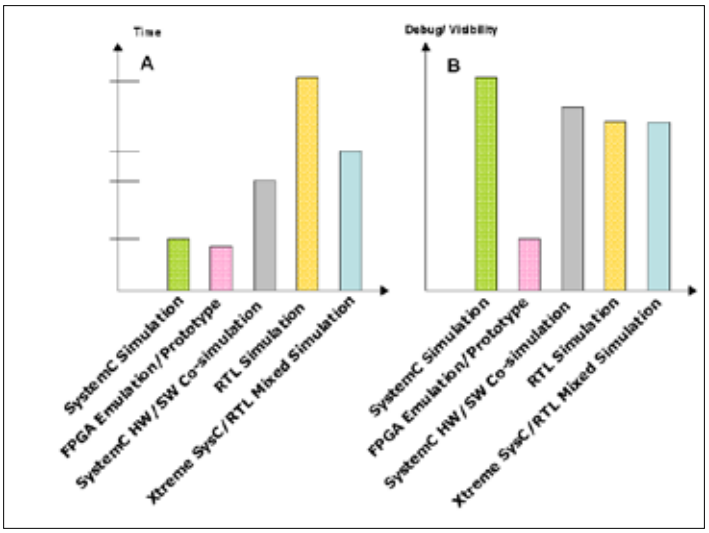

Figure 5: A) Simulation performance in different platforms. B) Debugging complexity in different platforms.

HW/SW partitioning

Decision making in partitioning of hardware and software in a system is the most important aspect. HW/SW partitioning becomes critical due to the factors e.g. the real time processing needs of a system, or memory limitations for application software, or other reasons. Many times, the decisions of the design space exploration phase are taken intuitively or on the basis of past experiences. However, there is a risk involved if something goes wrong. With the increased complexity of systems and costs involved for silicon, such decisions may prove to be blunders. These are the reasons that emphasise the need for a methodology which could help in making better decisions for hardware/software partitioning.

In the case of UWB MAC System development, there were time constraints that must be well respected as the application layer is totally dependent on global timings on the air i.e. RF Antenna. Our approach for reaching a decision was based on experimentation that we performed with our detailed system level platform. We were able to analyse the data flow in a pipelined data path, and we could see if there might be any bottlenecks in the system. Typically, the data flow in the system was transmitted where data frames had to be sent towards PHY from MAC and receive, where data frames were generated towards MAC from PHY and stored in memory for further analysis by software. During analysis of simulation scenarios, we are able to identify the need to perform some protocol parsing in hardware to take timely actions.

An example of decision-making is detailed in Figure 3. As per protocol needs, there is the reception of control packets, which informs about the next transmit event global timing i.e. when the next data packet has to be transmitted. Considering the MAC HW is a typical data path and passes the control frame in memory, the SW processed the frame and decided to open a transmit window. We could see a bottleneck with this approach where the opening transmit window would be missed. The system platform's results were used to confirm this un-

derstanding, and we were able to make better decisions for a more efficient system. Another scenario in Figure 3 shows the results after HW/SW partitioning is done.

In the first case, when the SW is processing the control frame, the global timings are as follows:

Time to program the window = T + tRP + tPM + tintr + tsw_lat > T + texp, hence in this system, SW fails to program the instruction to open the transmit window in time.

In the second case, when MAC HW is processing the control frame, the global timings are:

Time to program the windows = T + tprg_win < T + texp, hence in this system, HW programs the instruction to open the transmit window in time.

At the same time, the performance of the AES-CCM engine was measured on the SPEAr board. We were able to conclude AES-CCM to be in hardware as AES CCM software algorithm is not able to give the required performance needed.

Challenges

Testing of Design Under Test (DUT) or Unit Under Test (UUT) is one of the major concerns of any design methodology. During initial development i.e. architectural evaluation, a high performance simulation environment is a necessity. Behavioural TLM platforms fulfil this requirement and permits functional checks to be performed. When we go for lower level abstracted design, simulation performance goes down significantly, and it becomes an issue to verify the IP efficiently.

The System level verification of hardware and software is performed with co-simulation of hardware and software together. We had our own platform, which is a mixed platform consisting of hardware (RTL) models and software written in SystemC (Figure 2). The bottleneck of this platform was the RTL of IP introduced in the environment, and we noticed significantly decreased performance. As expected, this is the limitation which we encountered, and we evaluated a possibility to make it run faster than the host simulation. This approach is based on Xtreme server based hardware emulation which permits us to run the simulation at least 10X faster than using NCSIM simulations.

Xtreme server

This technology, shown in Figure 4, was useful for us in first time evaluations and without requiring any major efforts in environment set up. The idea is to run the RTL IP in FPGAs available with Xtreme technology. Initially, the clock introduced was a software clock, but the results are quite encouraging and facilitates system verification/debugging of RTL. In the setup, the complete simulation environment is similar and the only change is the replacement of SysC IP with VHDL RTL IP. Our experimentation resulted in 10X faster simulations. The most important aspect of the technology was to be able to have the same level of debugging capabilities as with ncsim. It also provided integration with the SystemC environment.

Debugging capabilities

One more challenging issue of hardware design is debugging. Whenever any self test case fails, a debug of that test case is needed. To verify that test case, one has to look all the major signals while guessing at some causes of that fault. So there is a need to dump the signals, verify the signals, and find out the exact cause. All these features can easily be performed using the XTREME SERVER BASED platform without extra effort. Improving simulation speed can be achieved by putting the real hardware into separate FPGAs, but this method provides fewer debugging capabilities. Hence, the Xtreme server based platform improves the speed of simulation. Figure 5 shows the results of our analysis.

FPGA prototyping

The next logical step in our functional verification methodology was to test the design in real time. Though modelling the hardware at a high level of abstraction can provide results at high speed, potential problems in the HW/SW integration cannot be augmented solely by this. Also, running a design on FPGA prototypes with real stimuli permits far more exhaustive and realistic functional coverage, as well as early integration with the software.

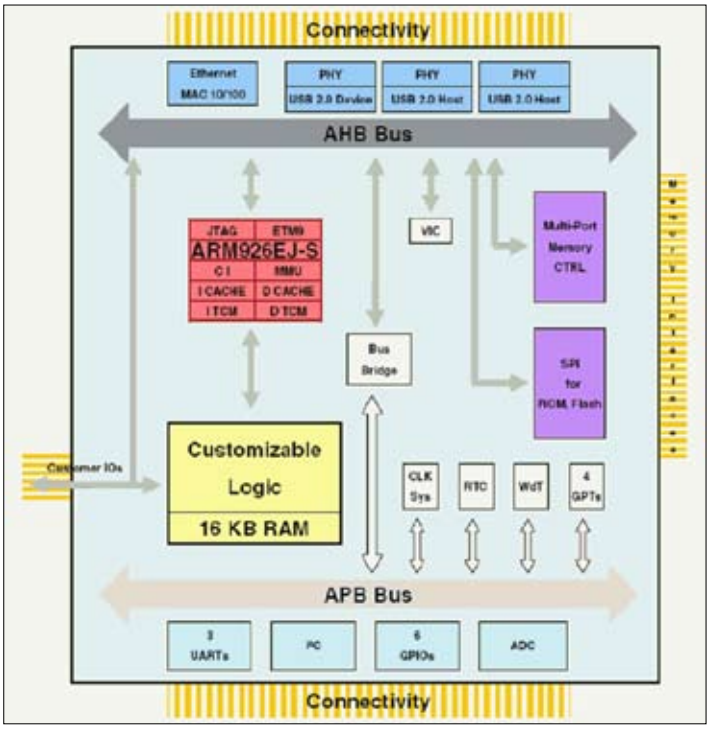

Figure 6: A general SPEAr (SPEArHead) SoC architecture.

SPEAr supplies a digital engine that offers the possibility of designing special user functions in a fraction of the time and with low investment (Figure 6). SPEAr consists of one or two ARM926 processor cores with 16k (data) + 16k (instruction) of cache memory running at 333MHz (worst case conditions). It also provides 600,000 gates (ASIC-equivalent) of embedded configurable logic, complemented with memory interface supporting DDR/DDR2 memories, and a large connectivity IP (intellectual property) portfolio.

Our targeted IP (UWB-MAC) was partitioned into two SPEAr boards: MAC RTL into one board and PHY Emulator code into the other. Both the boards were connected together with a connector board, which emulated the MAC-PHY Interface. Both the boards were controlled by software running on a PC through respective ethernet interfaces. The FPGA present on the board has three interfaces namely AHB, DMA, and Interrupt.

The custom logic (MAC RTL & PHY Emu, in our case) along with the glue logic (logic required to connect to three interfaces) was ported onto the FPGA. Software developed in the previous step was successfully run on the ARM platform provided with SPEAr. The same test-suite was integrated and functionality was seen to be in unison with the results of other architectures.

Future work

Currently, we have reached a stage where we have a System Prototype with a large set of test vectors. In order to further improve the simulation performance, some activities have been identified that we might be working on as ongoing research activity. Mixed simulation for larger test cases is the major bottleneck. . We are trying to overcome these limitations by separating out HW and SW functionalities on

the Xtreme server based platform (including test environment), as shown in Figure 7. The partitioning needs to be such that HW is running at speed and interacting with test environment, and SW is running independently. This is the ideal case when the technology could be exhaustively deployed and would give maximum simulation performance. We expect to achieve a 10-15x simulation performance with this strategy, which is 20-30x faster than pure RTL simulations.

Another activity in the near future is validation of our system prototype with DBB and RF chips. Since we have already conceptualised the board preparation for MAC and PHY chips separately, we have a lot more flexibility to directly plug the real PHY chip and run the tests. Another important task is to introduce a Lecroy protocol MAC PHY interface to check compliance of protocol for frames received and transmitted.

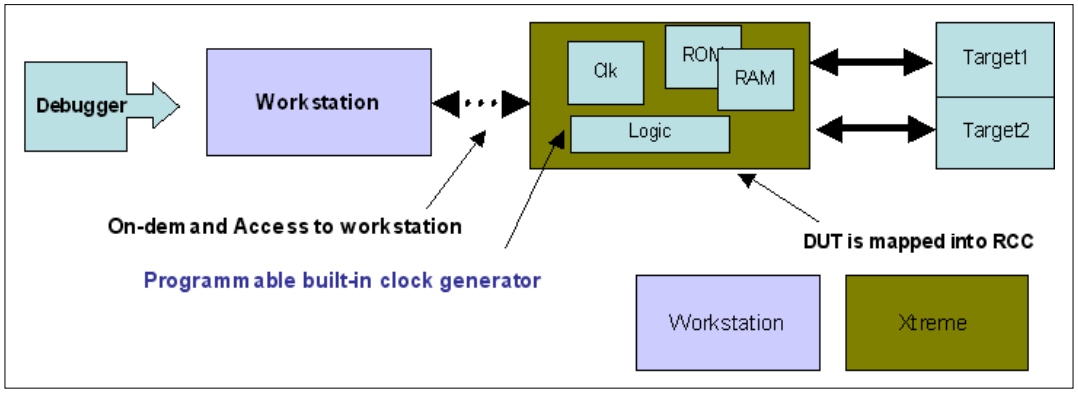

Figure 7: Xtreme server box set up.

The same approach will be used for further development of specific IPs that would use UWB technology as the underlying layer e.g. Wireless USB, WiNet etc.

Conclusion

This methodology helped us to achieve our goals at the end of the first phase of the project. We were able to not only develop a functional prototype but also set up a methodology that could be of help in the long term. There had been challenges in hardware/software partitioning, improving simulation performance for RTL verification etc., and with the methodology we adapted/improved, we were able to cope with the critical requirements of the project.

The System Level platform for UBW MAC is currently being exchanged with other teams to share our views on overall protocol checking. At the same time, using the platform raises the possibility of running software on it and developing test suites. Concurrently, we have the same test benches/test vectors across the design flow. It has been helpful to us to be able to run verifications of the DUT.