# ViP: A Practical Approach to Platform-based System Modeling Methodology

**Junhyung Um, Sungpack Hong, Young-Taek Kim, Eui-young Chung, Kyu-Myung Choi, Jeong-Taek Kong and Soo-Kwan Eo**

*Abstract*—**Research on highly abstracted system modeling and simulation has received a great deal of attention as of the concept of platform based design is becoming ubiquitous. From a practical design point of view, such modeling and simulation must consider the following: (i) fast simulation speed and cycle accuracy, (ii) early availability for early stage software development, (iii) inter-operability with external tools for software development, and (iv) reusability of the models. Unfortunately, however, all of the previous works only partially addresses the requirements, due to the inherent conflicts among the requirements. The objective of this study is to develop a new system design methodology to effectively address the requirements mentioned above. We propose a new transaction-level system modeling methodology, called ViP (Virtual Platform). We propose a two-step approach in the ViP method. In phase 1, we create a ViP for early stage software development (before RTL freeze). The ViP created in this step provides high speed simulation, lower cycle accuracy with only minor modeling effort.(satisfying (ii)). In phase 2, we refine the ViP to increase the cycle accuracy for system performance analysis and software optimization (satisfying (i)). We also propose a systematic ViP modeling flow and unified interface scheme based on utilities developed for maximizing reusability and productivity (satisfying (ii) and (iv)) and finally, we demonstrate VChannel, a generic scheme to provide a connection between the ViP and the host-resident application software (satisfying (iii)). ViP had been applied to several System-on-a-chip (SoC) designs including mobile applications, enabling engineers to improve performance while reducing the software development time by 30% compared to traditional methods.**

*Index Terms*— **System on Chip, Platform-based design, Transaction-level modeling**

## I. INTRODUCTION

Convergence is a recent trend in electronic system design and SoC is one of the hot issues to integrate various functions in a single silicon device. Compared to traditional application specific integrated circuits (ASIC), SoC has faced new challenges that cannot be addressed by traditional design methodologies. Among them, the pressure of reducing the time-to-market in the presence of exponentially increasing design complexity has become critical in SoC design. However, it is not easy to handle this problem by traditional design methodologies because they are based on relatively lower level design abstraction, which is appropriate for simple designs, such as traditional ASICs. New research activities around these issues have introduced the concept of design reuse, and platform-based design methodology [1-3] has emerged as a method by which designers can efficiently and systematically exercise this reusability concept.

Virtual platform [4,5] (highly abstracted system simulation model) based design methodology, which basically captures the concept of the platform-based design approach, has recently been gaining interest and

Manuscript received April 11, 2005; revised June 9, 2005.
CAE Center, SoC R&D, System LSI Div. Samsung Electronics, Korea
E-mail : junhyung.um@samsung.com

recognition among today's industry leaders. In this paper, we will address the virtual platform methodology aiming at covering all aspects of benefits from platform-based design. The key factors that we consider in our virtual platform methodology are as follows: (1) hardware and software co-verification[1], (2) development and verification of the embedded software in connection with external SW development tools, (3) quantitative analysis and exploration of system architecture, (4) executable specs for HW/SW/System designers, and (5) managing large scale SoCs which exceed 100 million transistors on a single piece of silicon.

There are many works related to virtual platform design methodology. For performance analysis and architecture exploration, [9-12] present the design approaches to find the optimal communication architecture focusing on the IP component modeling method in the transaction-level and the exploration strategy of various communication architecture; [16, 20-22] suggest several methods to adapt transaction-level modeling for architecture exploration with high simulation speed. Furthermore, several works extended the scope of the virtual platform to the embedded SW design in conjunction with hardware design. Y. Nakamura[13] suggests an HW/SW co-verification method based on the integration of a C/C++ simulator and FPGA simulator, and [14] presents a parameterizable HW/SW platform that is customized for the rapid prototyping of code compression. However, most of the previous works have focused on only one or two aforementioned requirements, targeted to specific usages, and cannot be applied to general SoC applications.

To overcome these limitations, we developed our virtual platform design methodology, called V*i*P, consisting of IP models written in SystemC/C++ at transaction level, to achieve simulation speed more than 1000 times faster than the conventional RTL level design method. As discussed in the case study (Section 4), we applied our method to a very complex mobile application design that includes CPU, DSP, and more than 60 masters and slaves connected through a multi-layer bus system. The experimental result shows that we achieved simulation times more than one thousand times faster than using conventional RTL design with 90% timing accuracy compared to the FPGA board, proving that V*i*P can be used effectively for analyzing the system performance of different architectures.

## II. Overview of V*i*P

Platform-based design is a design approach that emphasizes the systematic reuse for developing complex products based upon platforms and compatible hardware and software. The advantages offered by this methodology are becoming indispensable elements in today's SoC development, where the design complexity has been increasing exponentially.

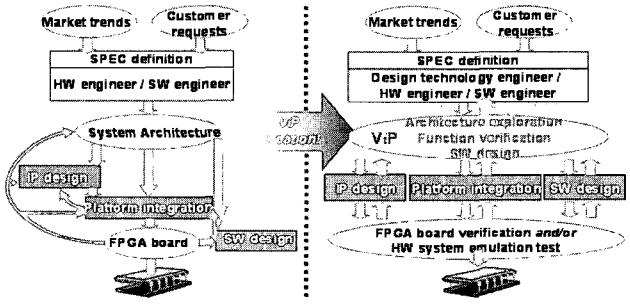


**Fig. 1.** The movement of design flow

The proposed HW/SW co-design methodology, V*i*P or a highly abstracted system simulation model consisting of a multiple processor and the IP component models described in the function-accurate and/or cycle-accurate level, will overcome the shortcomings apparent in the traditional design methodologies by enabling hardware/software co-design, including software development, analyses of the software and hardware architecture early in the design process, thereby preventing the likelihood of encountering problems in the latter stages of design. V*i*P maximizes the advantages of the platform-based design approach, and provides further benefits as follows:

- V*i*P is modeled at the transaction-level (TL) [6-8] (these models are developed using SystemC or C++) and, thus,

1) ViP can also serve as a verification model for the entire system. It can serve as a tool that facilitates the performance analysis at the system level, including the functionality and performance of the embedded software running on the system.

offers a very high simulation speed with reasonable cycle accuracy.

- Architecture exploration and software development can happen early in the design process.
- Software designers can prepare fully-optimized and error-minimized software before the development of RTL code. This is especially beneficial when multiple embedded software engineers in multiple locations collaborate because the simulation model does not have physical restrictions, unlike an HW board.
- ViP, or essentially the executable specification of the chip, can be used for marketing purposes before the real product or board is available.
- ViP provides a complete SW development environment - it is possible to connect with external SW development tools executed on the host system.

Therefore, this methodology considerably improves system design productivity and reduces SoC design and software development cycles, as shown in Fig. 1. We have applied this new ViP based methodology to our various SoC designs, and we are currently in the process of incorporating the new flow into our SoC design infrastructure. The development procedure, usage and features of the ViPs we have developed will be described in the next chapter. Then, we conclude this paper with the future roadmap of ViPs in SoC development.

## III. ViP Development

Let us begin with the requirements to satisfy the five factors suggested in Section 1. Then, we will describe the overall flow of the ViP development procedure, which was constructed based on those requirements.

The requirements for SW prototyping are *fast simulation speed* and the *early availability of the SW development environment*. On the other hand, for performance analysis and SW optimization, more emphasis should be placed on *cycle accuracy*, and *reusability* is also decisive factor to curtail the development period of derivative designs. Also, ViP sometimes needs to *communicate with the SW running on the host PC*, which is designed to control and monitor the SW ported on the embedded system for debugging and testing purposes. Finally, to be an executable system specification for RTL designers (they can reside in separated regions), the ViP should satisfy the *easy distribution* property. This is especially beneficial compared with the physical board when people in multiple locations are working together because there is no limitation to copy, distribute, and modify the ViP.

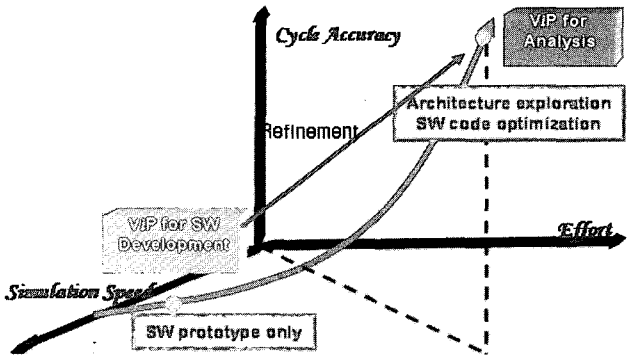


**Fig. 2.** ViP development trade-off

Our goal is to build our own ViP creation procedure which will enable us to satisfy all the previously mentioned requirements. Among these requirements, since *cycle accuracy* and *simulation speed* are in strong inverse proportion, and moreover, since *modeling effort* also has a co-relation with simulation *speed* and *cycle accuracy*, as shown in Fig. 2, it is important to find out the optimal trade-off depending on the ViP development purpose in our method. Moreover, to satisfy early *availability* and *reusability*, we build a four-step ViP creation flow - IP modeling, bus system modeling, integration and verification, and value added work like architecture exploration and software development. This flow will be discussed in detail in this section.

### 1. IP Modeling

Since our target is to achieve *more than 90% accuracy* and *fast simulation speed* for a SoC system which exceed 100 million transistors on a single piece of silicon, we adapted a transaction-level modeling [6] method, and we use a two-step cycle based simulation to increase simulation speed even more. Moreover, to reduce the *modeling effort* to develop a cycle accurate model, and to achieve the enhancement of *productivity* and *reusability*,

we created our own systematic IP modeling flow, which will be described in the following sections.

**Code reusability**: We separated the functional and communication parts of each IP model to achieve systematic code reusability, as shown in Fig. 3. The main functional behavior is described in the core block while communication is totally delegated to a dedicated routine called "the communication handler," which is prepared for various communication schemes with a unified programming interface. Therefore, all the IPs use the unified bus interface scheme, and this enables easy adaptation to possible changes in the environment, which include a system bus or even a whole new simulation environment.

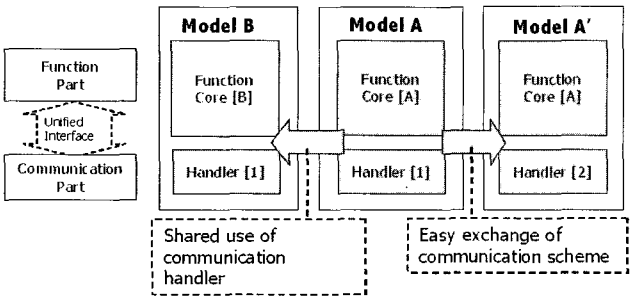


**Fig. 3.** IP modeling for reusability and productivity

**Classification of transaction-level IP models**: In our modeling procedure, IPs are developed according to the target of the whole ViP. If the goal of the project is fully satisfied by a function-accurate ViP, we use function-accurate IPs for that system. On the other hand, if the target requires cycle accuracy, we make all the IPs that will be integrated to the system as cycle accurate models. To support these IP modeling in the degree selection and development planning of the IP creation procedure, here we categorize IP models, as shown in Table 1, listing the features and the typical application areas for each TL model types.

**Table. 1.** Level of IP modeling

| | Description | Features | Application Area |
|---|---|---|---|
| TL0 | Functional Model | Functional description wrapped into a model. Delays for function and even communication are neglected. Fast to simulate, easy to deploy. | SW development |
| TL1 | Parameterized transaction model | Parametric function delay (input-to-output delay). Strict communication protocol delay. | Top down Design |
| TL2 | Cycle accurate model | Internal structures are modeled strictly. Aiming for cycle and pin accuracy. | Architecture Analysis |

The TL0 model aims for high simulation speed and early availability for SW development. In order to meet these criteria, we modeled IPs as function-accurate models and also zero-delay transactions for all communications. The TL1 model has parameterized functional cycle delays suitable for top-down design flow. Changing delay parameters under architecture exploration gives more chances for retail performance budget in the early design stage. Note that cycle delays for communication are strictly kept in this level. The TL2 model reflects the structural details of the target IP, especially for bottom-up design/verification flow. Keeping micro-architectural information enables the establishment of full cycle-accuracy and even pin-accuracy; however, this may cause simulation performance degradation, since the bottleneck of the total simulation speed is usually the slowest model.

**I/O interface block modeling**: Our modeling strategy to enhance simulation speed is to perform detailed modeling for analysis of the crucial parts only. Since the internal structure of the I/O interface blocks usually has little effect on the analysis result, we model I/O IPs as DMAs, which perform the same in/out activities with related HW IPs. In view of the whole system, since IP models created by our method produce the same results, our modeling strategy is sufficient for system analysis. However, if we want to optimize or obtain quantitative analysis results of the I/O IPs themselves, we should create a model on a more detailed level. Furthermore, we integrate the resources of the host PC directly in our ViP environment to provide more possible usages. For example, the image data stored in the LCD module of ViP can be displayed on the monitor of host PC, and moreover, we can connect UART of the ViP to the UART of the host PC so that the various legacy software using UART can still be applied without any modification.

## 2. IP Model Verification

The IP model itself should be verified against specifications or the existing HW IP. We define a three-step IP verification flow as follows.

**Unit testing**: In this step, functional verification is sufficient for the TL0 models, while timing verification is

also required for the TL1 and TL2 models. It is relatively easy to write test input vectors for transaction level models, but when the RTL test benches already exist, instead of rewriting them in TL, a co-simulation based approach is preferable. This requires a transaction-to-pin conversion routine for a communication handler.

**Verification of IPs integrated with a basic subsystem**: The subsystem consists of a CPU, a basic bus system, including functional memory and a bus, and a interrupt controller. We mainly focus on the verification of the interrupt controlling scheme, SFR behavior, and DMA operations between the IP and memory. We create simple software to run on the CPU and check the above behavior using the software.

**Full V*i*P verification**: All developed IPs are integrated and verified together. This step will be described in Section 3.4

## 3. Bus Subsystem Modeling

There are several works related to bus subsystem modeling. M. Caldari and S. Pasricha[16, 17] intensively addressed the general modeling issues related to the communication architecture in the transaction-level. In [15], AHB and APB bus of AMBA2.0 were modeled as TLM using SystemC2.0 language. They describe how to raise the abstraction-level to achieve higher simulation speed. In [16], a new abstraction level called Cycle Count Accurate at Transaction Boundaries (CCATB) was introduced. CCATB is placed in between the transaction level and bus cycle accurate level to trade off simulation speed and cycle accuracy. The authors modeled AMBA AHB and AXI at CCATB and a showed simulation speed improvement over the pin-accurate SystemC model, called AHB CLI, by 1.5 times.

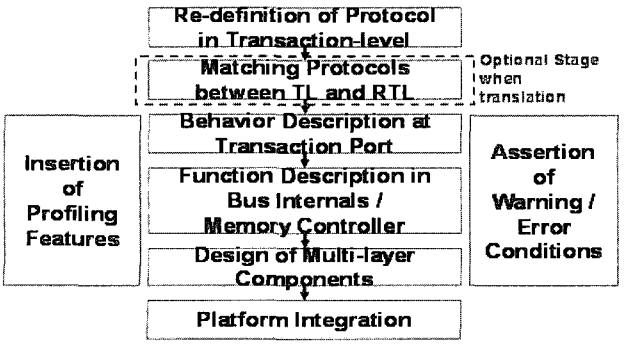


**Fig. 4.** Flow of bus architecture design

Unfortunately, they did not mention the cycle accuracy, which can be sacrificed by hiding some details in higher abstraction-levels, so it was not possible to understand the trade-off between simulation speed and cycle accuracy. V*i*P requires more than 90% cycle accuracy for the entire system with high speed, which means that bus cycle accuracy should be exactly the same as the specification or legacy RTL bus system. Having these necessities in mind, to increase the simulation speed, we use method-based modeling method rather than a thread-based method, and to speed up the simulation further, we use a two-step cycle-based simulation tool. Fig.4 shows our transaction-level modeling procedure.

**Table. 2.** Signal to transaction-level port mapping for AHB

| AHB RTL | AHB Transaction | AHB RTL | AHB Transaction |
|---|---|---|---|
| HCLK | global | HTRANS[1:0] | *ctrl[ACC][5:4]* |
| HRESET | global | HPROT[3:0] | *ctrl[ACC][9:6]* |
| HBURSREQ | *requestAccess()* | HSIZE[2:0] | *ctrl[ACC][12:10]* |
| HGRANT | return value of *checkForGrant()* | HWRITE | *read()* or *write()* call respectively |
| HADDR[31:0] | *read(addr, ..., ...)* / *write(addr, ..., ...)* | HWDATA[31:0] | *write(..., *data, ...)* |
| HLOCK | *ctrl[ACC][0]* | HRDATA[31:0] | *read(..., *data, ...)* |
| HBURST[3:0] | *ctrl[ACC][3:1]* | HREADY | return value of *read()* or *write()* |

To model bus architecture at the transaction-level, the AHB protocol needs to be redefined as a transaction-level protocol. Bus protocol in a design specification is usually described at the signal level, so it is necessary to map signal-level protocol into TL-level protocol, which is performed by transaction level ports (typically, transaction level ports are implemented as variables or functions). Table 2 is a mapping table of AHB. The left column lists the signals of AHB, and the right column represents the corresponding variable or function of each transaction-level port.

The next step is to model the behavior of each transaction-level port. Based on the definitions in Table 2, the behavior of each transaction-level port was modeled according to each burstX transaction scheme. Fig. 5 illustrates single transaction in AHB protocol. The procedure of sending "HBUSREQ" (bus request signal) and receiving "HGRANT" (bus grant signal) which is performed by RTL master can be represented as the

following. TL-master check bus grant by "checkforgrant()" function call and if the master cannot access bus grant, it sends bus request by "requestAcess" function call. Moreover, the procedure of sending an address, checking hready, and data read can be executed by "read(addr, *data, *ctrl)" function call. In this procedure, the TL-model checks hready by the return value of "read()" function.

After designing the behavior of transaction-level ports, internal functions of the arbiter were implemented. The internal function of AHB bus delivers the request from the master ports after performing arbitration at every clock cycle. Our AHB protocol supports a fixed priority and round-robin arbitration algorithm, and we can select one of them by the SFR setting.

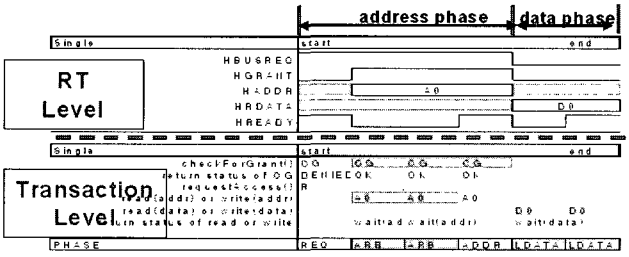


**Fig. 5.** Single transaction

Our modeling strategy to increase simulation speed is to perform detailed modeling for analysis crucial parts only. As for the memory subsystem, memory is modeled as a zero-delay functional model, but we model the memory controller to reflect accurate latency at the bus boundary. That is, we categorized the pattern of the access and burst type and gave a wait cycle so that the data movements between the bus and the memory controller could satisfy the cycle accuracy. This is valid if the usage of memory subsystem is limited to give an accurate bus access pattern to the whole system. On the other hand, if the main target of the analysis is memory structure itself, memory also should be modeled in a more detailed way.

In a complex system, for the bus system efficiency, we use multi-layer structure to distribute bus traffic and minimize the bus access wait of each master. In addition, to enhance the performance of memory access, we connect number of buses to the memory using a bus matrix and set the memory map to support concurrent memory bank access. We also modeled these features and integrated them into our bus subsystem.

## 4. Integration

Since cycle accuracy and simulation speed lies in a strong inverse proportion relationship, as shown in Fig. 3, we propose a two-step approach: in Phase 1, we create ViP for early SW development (before RTL freeze), and then in Phase 2, we refine the ViP to support performance analysis with more emphasis on cycle accuracy, while the SW is being developed using the model developed in Phase 1. In the case of ViP for early SW development (before the RTL freeze), the cycle accuracy can be sacrificed for a higher simulation speed. On the other hand, for the system designers, more emphasis is put on cycle accuracy and especially on accurate bus cycle simulations. Having these applications in mind, we decided on a two-phase approach where we first develop a function-accurate ViP for the SW developers, and while the SW is being developed, we would fine-tune the ViP to support cycle accurate simulations for the system designers. These ViPs will be described in detail in the following subsections.

**ViP for SW development**: We use TL0 IPs and the IA (Instruction Accurate) processor model to increase the simulation speed. In addition, we create a target-oriented ViP; that is, we integrate minimal set of target related components. For example, to make the ViP for codec-software development, we only integrate MPEG4 related HW IPs (ME/MC/DCTQ/VLX), a process core, memory, and a basic bus structure.

**ViP for architecture exploration**: On the other hand, for architecture exploration, we use cycle accurate models with less consideration of simulation speed to obtain the exact analysis data. We replace TL0 IP with TL2 IP models, the IA processor model with the CA (Cycle Accurate) processor model, and use a cycle accurate bus subsystem. These two ViPs are basically functionally equivalent, but they are different in terms of simulation speed and cycle accuracy. We also conducted experimentation to compare the simulation speed of the ViPs with RTL environment. Because a full chip RTL simulation environment like ViP is not available, we

measured the simulation speed of only one RTL IP, and Table 3 shows the results. It is noticeable that the cycle accuracy causes serious simulation speed degradation. Thus, it is necessary to consider the objective of virtual platform creation in the beginning of the development procedure.

**Table. 3.** Simulation speed comparison

| | RTL | RTL (One IP) | ViP (for SW) | ViP (for Analysis) |
|---|---|---|---|---|
| Simulation Environment | N/A | RTL (Post Processor) | Mobile ViP SW : MPEG4 Decoder | |
| Speed (cycle/sec) | | 400 | 500,000 | 70,000 |

## 5. A Connection Scheme between Host Application and ViP: VChannel

One of the main objectives of ViP is to provide an environment for software development prior to silicon. This environment includes various kinds of developing tools, like source-level debuggers, command shells and status monitors. In conventional test-board based methodologies, these tools are executed on the host machine and connected to the system via physical layers, such as JTAG, parallel port, UART, etc. To transparently migrate the SW development environment from test board into ViP, as shown in Fig. 6, we developed a generic scheme to provide a connection between the ViP and the host-resident application, named "VChannel." The major issues that we considered in VChannel development procedure are as follows:

- Adaptability: In the view of SW developers, the difference between the user interface of conventional approaches, like FPGA, and the ViP should be minimized to support easy and efficient adoption to the new debugging environment.
- Speed mismatch: Host SW is usually targeted to hardware boards, but it is simulated in the ViP, which runs about 1000 times slower than the host system. Some buffering schemes to recover this mismatch are mandatory.
- Data flooding: This problem is also caused from the speed issue between ViP and host system. The late reply of ViP causes unexpected repeated retransmission of data from the host, resulting the phenomenon known as "data flooding."
- Simulation speed degradation: The simulation speed of ViP can be degraded by connecting with host system.

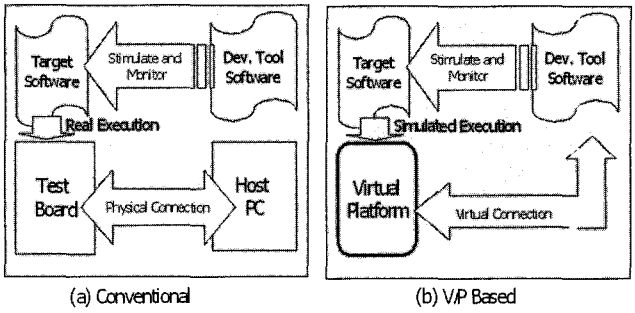


**Fig. 6.** Migration of the SW development environment

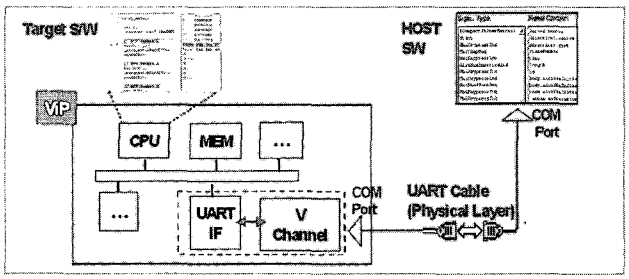


**Fig. 7.** Concept of the VChannel

Fig. 7 shows the concept of the VChannel. The behavior of the component in a ViP integrated with a VChannel is identical to the actual HW IP. For example, a UART component integrated with VChannel in the ViP uses the same memory map, interrupt scheme, and SFR settings as the actual HW device. As a result, modification of the target SW is not required. On the other hand, let us consider the interface between the VChannel and host system. The VChannel communicates with the host system through the same physical-layer interface as the test board environment. It handles the physical-layer hardware through the system call interface of OS, such as COM port API for UART in Windows® OS. Therefore, since the physical-layer interface is exactly the same, modification of the host application is also not required.

A VChannel is composed of two separate sub-modules, the Universal Host Port (UHP) and Universal Target Port (UTP), as shown in Fig. 8. The UTP communicates with other components in the ViP by signal interfaces, while the UHP communicates with host-side software through physical interconnection. Each sub-module has a two-level buffer. The buffer is located in the UHP store data from/to

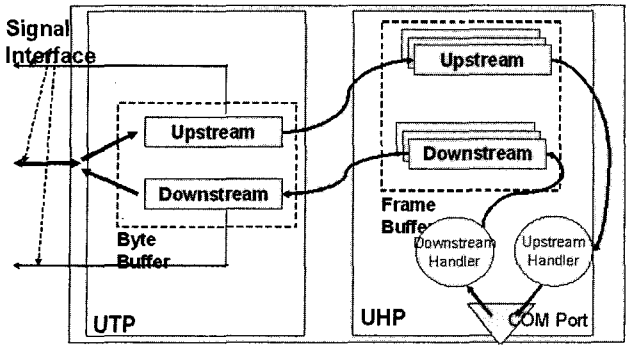


**Fig. 8.** Structure of buffers in a VChannel

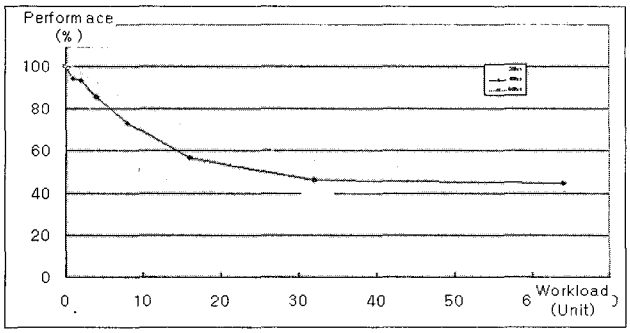


**Fig. 9.** Results for the various workloads

host in the unit of frame, and the UTP-side buffer is accessed byte-wise from interfacing components in the V*i*P. In each buffer, we separately allocate downstream buffers from upstream buffers to avoid the data flooding problem.

The data flooding problem, infinite data retransmission or link loss after a timeout produced by speed mismatch, cannot be avoided by simple approaches, such as an increase in buffer size. To solve this problem, we make the VChannel to perform its own communication protocol prior to the execution of the target SW. In other words, it inspects each data frame and returns a suitable answer to host SW long before the target SW processes the data frame. To support this sophisticated protocol buffering, UHP uses its own thread separated from the simulation engine. This threaded execution of the protocol enables quick response to the host SW, even when the V*i*P engine is stopped. We also conducted a set of experiments to analyze this effect by integrating a performance degrader that uses its own thread in a simple RISC-based system and watched the relationship between the amount of thread overhead cost and simulation speed degradation. The degrading component infinitely repeats a certain job to generate workload. Our experimentation was executed on a Pentium-4, 2.4GHz, 768MB machine with Windows XP OS. We used Maxsim [22] to simulate this environment. The result is summarized in Fig 9. The X-axis indicates the workload[2], and Y-axis indicates the simulation speed degradation with respect to original system.

The result indicates that the simulation speed decreases in proportion to the workload, up to 40%, and the degradation is negligible when the workload of the thread is relatively small. From the above experiment, we can conclude that the activation time of the thread in UHP should be minimized. Actually, this time depends on the type of data protocol and the volume of data treated by UHP, which is usually not significantly large. For example, if we use a UART connection of 115,200bps speed, at most about a hundred bytes of data are handled per second. This corresponds to about one workload unit in our experiment. We applied this scheme for creating the V*i*P of a GSM/GPRS system, where special command shell application is necessary to develop GSM/GPRS protocol stack software. Here, the speed degradation caused by the VChannel was less than 5%.

## IV. Case Study: V*i*P for a Mobile Device

We applied the suggested V*i*P methodology to a contemporary mobile design to illustrate how our V*i*P methodology has benefits the real world SoC design process. Our V*i*Ps are applied for performance analysis and SW development for multi-core and dual chip designs. Also, we constructed V*i*Ps with 20% effort for the next derivatives by reusing IPs in the mother version. We will now provide the details.

**Performance analysis**: We analyzed the performance of MPEG4 codec using a cycle accurate V*i*P. Performance was measured in terms of clock speed. Our main targets

2) We used the unit of workload as the amount of work to perform bubble sort for 1000 pieces of data. Since bubble sort is an $O(N^2)$ algorithm, to sort 2000 pieces of data, for example, corresponds to 4 units.

were: (1) to compare previously developed MPEG4 HW IPs with the improved ones; (2) to find the improvement factors of software through systematic quantitative analysis; (3) to optimize the performance of the software.

To compare the previous and current MPEG4 HW IPs, a general MPEG4 video reference SW, 11 I frames, and 6 P frames with Akiyo video sequences were used. Fig. 7 shows the flow diagram of the MPEG-4 video encoder. FIMV1.0/FIMV2.0 indicates the previous/current MPEG-4 hardware IPs. We created V*i*Ps for both cases, and Table 4 summarizes the cycle count ratio. We used the execution cycle count of ME as a baseline.

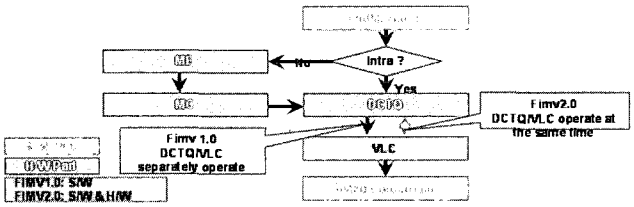


**Fig. 10.** Flow of MPEG-4 video encoder

**Table. 4.** The cycle count ratio of FIMV1.0/FIMV2.0

| MPEG4 HW | Initialization | ME | MC | DCTQ | VLC & Make Bitstream |
|---|---|---|---|---|---|
| FIMV1.0 (Previous) | 4.52 (7%) | 1 (1.5%) | 1.58 (2.3%) | 3.01 (4.3%) | 59.5 (85.5%) |
| FIMV2.0 (Optimized) | 4.52 (25.3%) | 1 (5.6%) | 1.58 (8.8%) | 1.75 (9.7%) | 8.98 (50.0%) |

From the above results, we confirmed that the next derivative design seriously outperforms the current design, and based on the quantitative result, we decided to adopt next derivative. The performance improvement is mainly caused from the use of VLC and a direct data pass from DCTQ output to VLC. Then, we were interested in finding the software optimization point. We performed profiling analysis on a more detailed level, and Table 5 shows the results. Based on the results, we found the improvement factors, performed software optimization, and we achieved a performance improvement of over 400% .

**SW development running on multi-core**: We developed an audio software using a V*i*P and shortened the software development time by more than 35%. Our target audio software runs on a CPU and a DSP subsystems. The CPU does most of control operations, while the DSP's job is to decode the audio. Short messages (commands and replies) are sent via communication box registers, and large data

**Table. 5.** Profile analysis - FIMV2.0 I/P Frame

| Frame | Operation | (%) | Bottleneck |
|---|---|---|---|
| I frame | Initialization | 68.7 | - Parameter Initialization<br>- Non-necessary parameter assignment<br>- Non-necessary memory allocation |
| | DCTQ/VLC | 13.3 | |
| | Making Bitstream | 18 | Writing bitstream (DCTQ to Memory) |
| | SW Part : 86.7%, HW Part : 13.3% | | |
| P frame | Initialization | 25.2 | - Parameter Initialization<br>- Non-necessary parameter assignment<br>- *Non-necessary memory allocation* |
| | ME<br>MC<br>DCTQ/VLC | 5.6<br>8.8<br>9.7 | |
| | Making Bitstream | 50 | - Motion vector VLC<br>- Writing Bitstream (Motion vector to VLC and DCTQ VLC to HW) |
| | SW Part : 75.2%, HW Part : 24.8% | | |

transfers are done through DMA. The most time-consuming part in our modeling process was the verification of the communication part of the CPU and DSP subsystem. In the verification process, we tested CPU subsystem first, which consists of the CPU core, dummy memory, and the basic bus subsystem; that is, we only integrated a minimal set of audio-related IPs. Using simple software running on the CPU, we checked the interrupt scheme and basic data movement though DMA. Then, to test the DSP subsystem alone, we used a self-checking program to verify the interface between peripherals. These tests gave us a clear indication of the functionality of the peripherals. The self-checking tests were "standalone." These are DSP programs that do not require external stimuli.

After testing two subsystems separately, we checked the interface between two subsystems using simple Inter-Processor Communication (IPC) software. The problematic parts were message handling IPs. Actually, we spend most of the time verifying these parts. As for the final functionality test, we ran public-domain audio decoding software to check the entire functionality. We could not hear the audio sound directly because simulation speed was too slow for real time audio processing. Instead, we saved decoded PCM audio data to a file and converted it to a wave file using an audio converter.

**Table. 6.** Results for cycle count comparison

| | V*i*P (for SW) | V*i*P (for Analysis) |
|---|---|---|
| Simulation Environment | CPU subsystem + DSP subsystem SW : Test Audio Software | |
| Total Cycle Count | 7,699,730 cycles | 35,748,518 cycles |

Finally, to develop a cycle accurate model, we refined the model, and Table 6 shows a cycle count comparison between the functional and cycle accurate models. The comparison indicates that the functional model significantly reduces the simulation time.

**ViP for the next derivatives**: We also developed ViP for the next derivative mobile applications. By maximizing reusability, the transaction-level design time for the derivative products was curtailed by more than 80%. The main differences from the mother version are as follows: (1) the codec uses a new parallel processing scheme for speed enhancement; (2) VGA images are supported; (3) bus topology is changed. Standard peripherals (such as a timer), UART, interrupt controllers are reused. In regards to the codec IP, because the scheduling scheme and all the register settings were changed, we could only reuse was the engine part. We already separated the engine part from the communication part of the mother IP; therefore, it[MIK1] takes only 40% more effort than the mother version. For the other IPs, only the sizes of the buffer and bus interface schemes were modified to handle bigger (VGA) images. Through the parameterized buffer size and bus protocol of the mother IPs, we could reuse the mother IPs immediately, and moreover, we could explore various HW parameters and predict the chip performance in advance. In conclusion, the reusability factor should be thoroughly considered in the IP development flow. We should separate the engine and communication part of IPs and maximally parameterize the features of IPs, such as the base address, internal memory size, interrupt related numbers (such as IRQ) of each IP, and the bus access scheme.

## V. CONCLUSIONS

In this paper, we developed a comprehensive and practical platform-based system modeling methodology, ViP, to target large scaled SoC designs. ViP contains a set of useful and valuable features that solve a few critical issues in platform-based system modeling. In particular, it is possible to prototype the SW on a ViP running more than 1000 times faster than RTL design and to reduce the overall design time by starting the SW before the RTL design is completed. Also, the accuracy of architecture exploration is improved compared to previous approaches thanks to the pre-developed SW, which eventually reduces the design re-spin due to performance issues. Furthermore, SW can be concurrently optimized while the architecture exploration is performed thanks to the cycle-accuracy guaranteed by ViP. It is also worthwhile to mention that our method does not aim to replace the FPGA based verification stage, but aims to shift many issues in earlier design stages to reduce the design iterations by lowering the critical issues in later design stages.

We applied our ViP method to a large scale mobile application design. With the ViP modeled at transaction-level, we analyzed the performance of the system and identified a bottleneck quantitatively. Based upon the analysis results, it was possible to improve the overall performance by more than four times. Also, we achieved a 30% reduction in software design time by developing the software using ViP before the RTL design was completed. Moreover, our method was highly effective based on the reuse concept when we built the derivative version of ViP. From our experience, it took only 20% of the effort to develop a derivative version of the ViP compared to when the mother version of ViP was created. Finally, we validated the cycle accuracy of the ViP by comparing the simulation results against the FPGA board measurement. The result shows that its accuracy is more than 90% compared to the FPGA board.

Such successful results will surely motivate the continuous enhancement of our ViP design flow in the future and widen its application area, including power estimation, embedded software optimization, and so on.

## REFERENCES

[1] K. Keutzer, et al., "System-level design: orthogonalization of concerns and platform-based design", *IEEE Trans. on CAD*, vol. 19, no. 12, Dec. 2000.

[2] A. Sangiovanni-Vincentelli, et al., "Benefits and challenges for platform-based design", in *Proc. of DAC*, pp. 409-414, 2004.

[3] G. Smith, "Platform based design: Does it answer the entire SoC challenge?", *in Proc. of DAC*, pp. 407-407,

2004.

[4] S. Brini, et al., "A flexible virtual platform for computational and communication architecture exploration of DMT VDLS modems", *in Proc. of DATE*, pp. 164-169, 2003.

[5] J. Notbauer, et al., "Verification and management of a multimillion-gate embedded core design", in *Proc. of DATE*, pp. 425-428, 1999.

[6] L. Cai and D. Gajski, "Transaction level modeling: an overview", *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis"* ,2003.

[7] I. Moussa, et al., "Exploring SW performance using SoC transaction-level modeling", *Proc. of DAC*, pp. 120-125, 2003.

[8] A.K. Deb, et al., "System design for DSP applications in transaction level modeling paradigm", *Proc. of DAC*, pp. 466-471, 2004.

[9] R. Jindal and K. Jain, "Verification of transaction-level SystemC models using RTL testbenches," in *Proc. of First ACM and IEEE International Conference on Formal Methods and Models for Co-Design*, 2003.

[10] N. Calazans, et. al., "From VHDL register transfer level to SystemC transaction level modeling: a comparative case study," in *Proc. of 16th Symposium on Integrated Circuits and Systems Design*, pp.355-360,. 2003.

[11] I. Moussa, et al., "Exploring SW performance using SoC transaction-level modeling," in *Proc. of the DATE*, pp.120-125, 2003

[12] K. Lahiri, et al. "LOTTERYBUS: A new high-performance communication architecture for system-on-chip designs," in *Proc. of DAC*, 2001

[13] Y. Nakamura, et al., "A fast hardware/software Co-verification method for System-on-a-Chip by using a C/C++ simulator and FPGA emulator with shared register communication", in *Proc. DAC*, 2004

[14] H. Lekatsas, et al., "Coco : A hardware/software platform for rapid prototyping of code compression technologies", in *Proc. of DAC*, 2003

[15] M. Caldari, et. al., "Transaction-Level Models for AMBA Bus Architecture Using SystemC 2.0", in *Proc. of DATE*, 2003

[16] S. Pasricha, et al., "Extending the transaction level modeling approach for fast communication architecture exploration," in *Proc. of DAC*, 2004.

[17] M. Bombana, F. Bruschi, "SystemC-VHDL co-simulation and synthesis in the HW domain", in *Proc. of DATE*, 2003

[18] A. Sayinta, et al., "A Mixed abstraction level co-simulation case study using SystemC for System on Chip verification", in *Proc. of DATE*, 2003.

[19] X. Zhu et al, "A hierarchical modeling framework for on-chip communication architecture", *Proc. ICCAD*, 2002

[20] O. Ogawa et al, "A practical approach for bus architecture optimization at transaction-level", *Proc. DATE*, 2003

[21] AHB CLI Specification www.arm.com/armtech/ahbcli

[22] Maxsim, AXYSDesign Inc., http://www.axysdesign.com

**Junhyung Um** She received the M.S. and Ph.D. degree from the Korea Advanced Institute of Science and Technology(KAIST) in 1999 and 2003. She is currently working for Samsung Electronics on next-generation system desigh technology. Her research interests include platform-based design methodologies, C-based HW design, and arithmetic opti-mization in behavioral and logic synthesis.

**SungPack Hong** He received the B.S. degree and the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 1999 and 2003. From 2001 to 2003, he worked at Eletronics and Telecommunication Resarch Institute (ETRI) where he was involved in research and development of bluetooth system. Currently, he is working for Samsung Electronics on the design techonology of digital system design on a higher level of abstraction and embeded software optimization for low-power system.

**Young-Taek Kim** He received the B.S and M.S degree in electronics engineering from Hanyang University, Korea in 1998 and 2000, respectively. His research interests include system architecture and interface optimization, prototyping methodology of System-on-chip, and system-level design verification. He is currently an engineer in SoC R&D center of Samsung Electronics, Kiheung, Korea.

**Eui-Young Chung** He received the BS and MS degrees in electronic and computer engineering from Korea University in 1988 and 1990, respectively. He also received the PhD degree in electrical engineering from Stanford University in 2002. Dr. Chung's research interests are design technologies for digital systems and VLSI design with special emphasis on system architecture, low power design, and embedded software optimization. He is currently a prinicipal engineer in SoC R&D center of Samsung Electronics, Kiheung, Korea.

**Kyu-Myung Choi** He received B.S. and M.S. degrees from Hanyang University, Seoul, Korea, in 1983 and 1985, respectively and Ph.D degree in electrical engineering from the University of Pittsburgh, Pennsilvania, USA, in 1995. His Ph.D research focused on the development of algorithms for CAD tools on system-level design automation. Since 1985, he is with Samsung Electronics as an engineer, senior engineer, and principal engineer of CAE(Computer Aided Engineering) team. He studied his Ph.D course as a Samsung scholarship. Now he is a Vice President of CAE center and the chief technologist of Design Technology TU(Technology Unit), System-LSI Division, Samsung Electronics. Also, he has served as the technical program committee members for a lot of local conferences and IEEE conferences including ISLPED 03, 04 and IEEE International SOC Conference 03, 04. His role in Samsung is to develop the design methodology for non-memory devices including SOC devices.

**Jeong-Taek Kong** He received the B.S. degree in electronics engineering from Hanyang University, Korea, in 1981, the M.S. degree in electronics engineering from Yonsei University, Korea, in 1983, and the Ph.D. degree in electrical engineering from Duke University, Durham, NC, in 1994. From 1983 to 1990, he was with Samsung Electronics Co., Ltd., as a VLSI CAD manager. From 1990 to 1994, he was at Duke University granted by a Fellowship from Samsung Electronics Co., Ltd. Currently, he is with Semiconductor Business, Samsung Electronics Co., as VP of CAE Team. He has authored and coauthored more than 90 technical papers in international journals and conferences and coauthored a book titled Digital Timing Macromodeling for VLSI Design Verification (Norwell,

MA: Kluwer, 1995). His research interests focus on various VLSI CAD tools and design methodologies. He has served on the program committees of IEEE International Workshop on Statistical Metrology, International Conference on VLSI and CAD, International Symposium on Quality of Electronic Design, International Conference on Simulation of Semiconductor Processes and Devices, International Workshop on Behavioral Modeling and Simulation, and International Symposium on Low Power Electronics and Design. He is a Member of Nanoelectronics and Giga-Scale Systems (NaGS) Technical Committee and serves as a Distinguished Lecturer for the IEEE Circuits and Systems Society. He was an Associate Editor of IEEE Transactions on Circuits and Systems-II and IEEE Transactions on Very Large Scale Integrated (VLSI) Systems and is a member of IMEC Scientific Advisory Board in Belgium.

**Soo Kwan Eo** He has been senior vice president of Samsung Electronics' SoC R&D Center focusing on the system design technology development in semiconductor business for 4 years. He currently directs the System Design Technology team that deals in SoC communication interconnect, hardware/software co-design, embedded software optimization and low power architecture exploration. Prior joining in Samsung Electronics in 2002, he had worked in the various Silicon Valley companies for 20 years as a director and Sr. system architect in the design groups including wireless and broadband communications applications. He received his MS degree in Physics from Sogang University in 1977 and MSEE from University of Arizona in 1986.