Hardware/Software Partitioning and Simulation with SystemC

RICHARD GALLERY, DEEPESH M. SHAKYA
School of Informatics & Engineering
Institute of Technology, Blanchardstown
Blanchardstown Road North, Dublin-15
IRELAND

Abstract: - This paper gives an overview of how the speed of simulation of Video/Graphics subsystem can be increased with SystemC [1][5][6]. Also, an idea of partitioning hardware and software at instruction level and the simulation framework to support this partitioning has been introduced. The simulation of each design space is conducted at the transaction level and the partitioning decision, which is effectively a selection of suitable design space, is made on the basis of the number of instructions, resource wastage factor and the hardware cost involved in the design space.

Key Words: - SystemC, Transaction Level Modeling, Codesign, Hardware/Software Partition, Simulation

1 Introduction

SystemC[2][7][8] is an industry led initiative to provide a modelling platform that promotes and accelerates system models which can then be proven using simulation tools before transfer to silicon.

The simulation of the system at the RTL (Register Transfer Level) level is very time consuming. In a very complex design this can be a major obstacle. There is a need of simulating the system at higher level of abstraction than the RTL level to boost the simulation speed and effectively decreasing the design time. SystemC provides the facility of simulating the system at the transaction level where the communication between the design modules takes place through interface methods. This significantly increases the simulation speed.

The design of a graphical subsystem like the video mixer can have multitude of design space. Each design space if simulated at the RTL level consumes significant amount of time. If a complex graphical subsystem design is considered, the simulation of large number of design space may easily be impractical. If each design space is simulated at the transaction level, it gives the designer drastic increase in simulation speed and thus makes it easy to verify the correctness of the design for each design space the designer chooses. The simulation framework which has been described in this paper is used to simulate the design space.

The suitable design space can then be chosen based on the instruction analysis, resource wastage factor analysis and the cost analysis.

2 SystemC Model of the Video Mixer

Typical video/graphics functionality includes video encoding/decoding, streaming, scaling, scrolling, vertical and horizontal filtering, multiple video overlays, video grabbing etc. Within our research we have chosen to concentrate on the use of SystemC to design, model and simulate a video overlay or video mixing engine.

The SystemC model of the mixer hardware has been divided into three major sections-Control Unit (CU), Data Addressing Unit (DAU) and the Arithmetic Unit (AU). The CU is microprogrammed i.e. it consist of a set of instructions, stored in the memory, to perform mixing operation. The DAU generates the address of the input data for the mixer and the address of the output data

where the output of the mixer is stored. Finally, the AU does the mixing operation.

3 RTL Level Simulation of Multiple Overlay Image Mixer

Initially an RTL based model of an image mixer was developed. This allowed the generation of composite images formed from multiple overlay images (the mixer has the capability of overlaying more than one image on the primary image) with the addition of the software layers that controls the mixer hardware. The communication between the software layers is performed at the transaction level. The software layers consist of an application layer (AL), presentation layer (PL) and the driver layer (DL). The application layer and presentation are hardware independent. The only software layer that interacts directly with the mixer hardware is the driver layer. The inclusion of the application and presentation layers allows a more complete system level model to be developed.

Fig.1 gives an overview of the mixer subsystem simulation where the mixer hardware has been simulated at the RTL level.

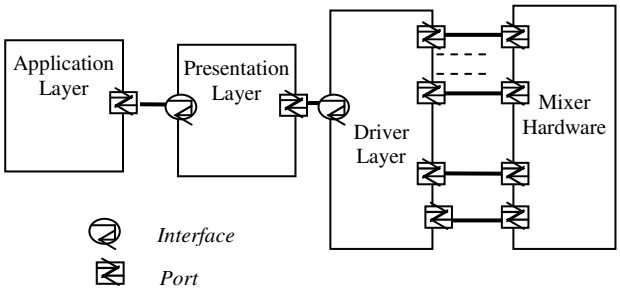

Fig.1 RTL level simulation of multiple overlay image mixer

Although the design constraints are more visible at RTL simulation but its efficiency soon diminishes due to high simulation time and difficulty in the design.

4 Transaction Level Simulation of Multiple Overlay Image Mixer

The communication between the different modules in the transaction level modelling [3][4] is implemented using a hierarchical channel. The hierarchical channel is a module which implements interface methods. Interface methods are defined in the interface but do not implement these methods. They are pure virtual objects without any data in order not to anticipate implementation details.

In order to improve the speed of simulation, the mixer hardware is simulated at the transaction level. To simulate the mixer at the transaction level, we have created a simulation framework which operates entirely at the transaction level.

4.1 Simulation Framework in SystemC

Fig.2 gives an overview of the simulation framework we have created for the simulation of the multiple overlay image mixer.

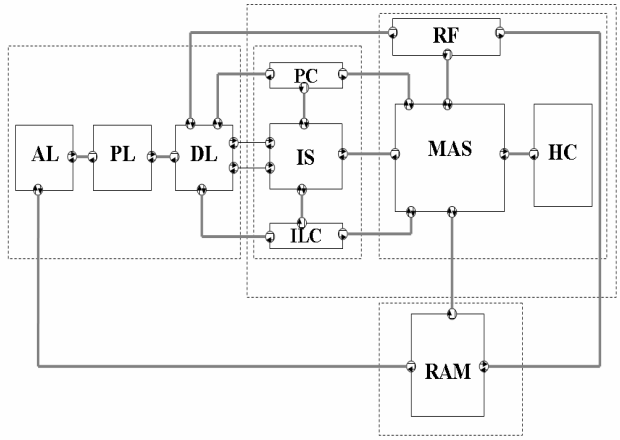

Fig.2 Simulation framework in SystemC

In our simulation framework, we have a control unit which comprises of program counter (PC), Instruction Store (IS), and Image Line Counter (ILC). The IS is where the microprogram is stored. The ILC keeps count of the number of image lines remaining to be processed. The other section is the software layer with AL, PL and DL. The hardware section consists of Register File (RF), Mixer Arithmetic Simulator (MAS) and the Hardware Components (HC). All these modules communicate at the transaction level.

When the system is run, the first instruction is executed. The IS passes the instruction ID of the first instruction to the MAS. The MAS is built of switch statements depending on the instruction IDs for e.g. if the first instruction is the initialization of the parameter registers (for e.g. image width, image height etc.) then the MAS after receiving this ID from the IS invokes the corresponding interface method in the RF and initializes the required parameter registers. If the instruction ID tells to add two image values then the MAS communicates

with the RF (to read input values for addition) through the interface method. The MAS passes the register ID and the mode (READ or WRITE) to the interface method in the RF and the RF returns the value of the register with the ID sent by the MAS. This is shown in the source code given below:

```
reg_img_wd_val=
    reg_file_port->reg_val_rd_wr
              (REG_IMAGE_WD,READ,NULL);
```

Having read both the values required for the addition, the MAS now calls the adder interface method in the HC. The HC consists of the number of hardware components (for e.g. 2 adders, 3 multipliers etc.) to be implemented in the design space under consideration. While calling the adder interface method, the MAS passes the two input values read from the RF. The output obtained after the addition operation is again stored in the RF. The MAS now passes the register ID where the output value is to be stored and the mode passed now will be 'WRITE' along with the value to be written instead of NULL in the code shown above. This is depicted in the code shown below:

```
reg_file_port->reg_val_rd_wr
            (REG_MIX_OP, WRITE, temp_3);
```

If we intend to use two adders in a design space then the HC will consists of two adder interface method emulating the adder module. If two add operations are to be performed in a single instruction, the MAS calls these adder interface methods one at a time. As described this may appear to imply sequential operation, but in practice all operations within a switch case are intended to execute in parallel. When the output image value is obtained then the MAS writes the output image value to the RAM.

5 Hardware/Software Partition

The partitioning decision adopted here is at the instruction level (instructions in this context are microcode instructions which then control the hardware directly). In the conventional hardware/software partitioning approach, dedicated hardware (for e.g. ASIC) is used for the hardware portion while the microprocessor is used for the software portion. But in our implementation software portion is represented by the microprogram in the control unit which controls the hardware selected for the mixing operation.

First an analysis is performed to investigate the design space topology. As we are processing *8* pixels of image at a time, we can perform *8* subtractions and *8* multiplications at one time i.e. in a single instruction. This means may be a maximum of *8* adders and *8* subtractors in our design selection. The total number of design space with this configuration will be *64* (for e.g. *1* subtractor & *2* multipliers, *3* subtractors and *3* multipliers and so on) Thus, in principle, *64* design spaces should be analysed and the design space which best suits our requirement selected. All these design spaces are simulated using the simulation framework described above. Once a design space has been simulated, the correctness of the design may be verified through the comparison of the output of the design (overlay image) with a reference output. The analysis of the suitability of the design spaces is based upon three major factors: the number of instructions, the resource wastage analysis and the cost analysis.

For each element of the design space, an instruction set is constructed which corresponds to the microprogram in the IS of the CU. With each set of instruction (and the appropriate Hardware Components) a simulation of the subsystem is performed using our simulation framework. After receiving desired result from the simulation, the total number of instructions required for that design space is analysed. Fig.3 shows a graph of the number of instructions in each of the possible design space. For simplicity, the design space has been restricted to be the function of the number of subtractors and the number of multipliers.

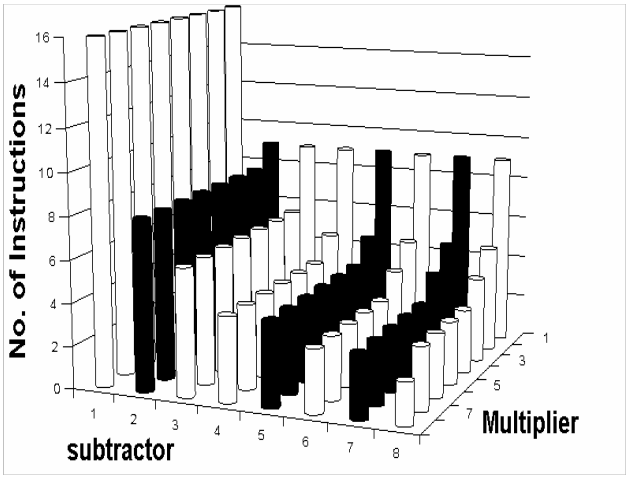

Fig.3 Instruction analysis chart

In Fig.3, it is clear that the design space with *8* subtractors and *8* multipliers has least number of instructions. The selection of this design space is not feasible due to resource wastage and the cost of the hardware components.

The resource wastage factor (RWF) is calculated for subtractor and multiplier for each design space. The RWF indicates how much the resource (subtractor and multiplier) is used (or remains idle) during the entire instruction set execution. We have created a formula to calculate the RWF as shown below.

$$RWF_S = \left(\sum_{i=1}^{NS} \frac{NS - xsi}{NS} + (N_I - NI_S) \right) \mathbf{x}\ (\mathbf{NS/NS_{max}})$$

$$RWF_M = \left(\sum_{i=1}^{NM} \frac{NM - xmi}{NM} + (N_I - NI_M) \right) \mathbf{x}\ (\mathbf{NM/NM_{max}})$$

Here, NS is the total number of subtractors used in the particular design space; NSmax is the maximum number of subtractor used in the entire design space. NI is the total number of instructions in the instruction set and NIs is the total number of instructions with subtraction operation. Xsi is the number of subtraction in the instruction *i*. The same denotation applies for the RWF calculation of the multiplier as well. The RWF for each design space are calculated using above equations.

After RWF analysis, a cost analysis is performed. The cost analysis is performed by calculating the cost of the multiplier and the subtractor if implemented in silicon. It is intended to implement the subsystem using an FPGA, and the cost it then arrived at by calculating the number of logic blocks required to implement the subtractor and multiplier in FPGA. The corresponding cost can then be easily calculated.

By making comparative study of all the analysis i.e. instruction analysis, RWF analysis, cost analysis and the other factors the most suitable design space is chosen which will be the compromise of the different factors and satisfies the requirement of the design.

6 Conclusion

We discussed the implementation of the SystemC based simulation framework and the partitioning technique in the mixer but the similar concept can be implemented for the design of other graphical subsystems like video encoding/decoding, streaming, scaling etc. The simulation framework and the partitioning technique we presented here can be enhanced to develop even the complex graphical subsystems efficiently at lesser design time.

References

[1] GrÖtker T Liao S, Martin G and Swan S, *System Design with SystemC*, Kluwer Academic Publishers, 2002.

[2] Official web site for Open SystemC, www.systemc.org.

[3] Parischa S, Transaction level modelling of SoC with SystemC 2.0, *Technical Papers from www.systemc.org* ,2002.

[4] Preeti Ranjan Panda, A modeling platform supporting multiple design abstractions, *Proceedings of the international symposium on Systems synthesis,* Volume 14, September 2001.

[5] Guido A, SystemC Standard, *ASP-DAC'00. Asia and South Pacific*, 2000, pp.573-577.

[6] Mueller W, Ruf J, Hoffmann D, Gerlach J, Kropf T and Rosenstiehl W, 2001, The Simulation Semantics of SystemC, *Design, Automation, and Test in Europe ,DATE '01*, 2001, pp.64-70.

[7] Gerlach J and Rosenstiehl W, 2000, System Level Design Using the SystemC Modelling Platform, *Workshop on System Design Automation, Rathen, Germany SDA'00*, 2000, pp.185-189.

[8] G.Economakos,P.Oikonomakos, I. Panagopoulos, I. Poulakis, and G. Papakonstantinou, Behavioral Synthesis with SystemC, *Design, Automation, and Test in Europe (DATE '01),*2001**.**