# Using Matlab And Simulink In SystemC Verification Environment By JPEG Algorithm

Walid Hassairi, Moncef Bousselmi, Mohamed ABID
CES Laboratory
National Engineering School of Sfax
Email: hassairi1@yahoo.fr , moncef.bousselmi@fss.rnu.tn, mohamed.abid@enis.rnu.tn

Carlos Valderrama Sakuyama
School polytechnique of
Mons, Belgium
Email:
carlos.valderrama@fpms.ac.be

***Abstract*— This paper presents the co-simulation interface matlab and systemC methodology, the development and modeling of a JPEG algorithm using this co-simulation interface. This Hardware/Software co-design technique is applied to the JPEG compression algorithm which has many implementations in recent video real time application. Time to market is the very important factor for development of such applications. The results obtained by modeling the hardware and software JPEG modules using our approach, indicate that the execution time was reduced by 85%.**

## I. INTRODUCTION

Hardware/software [1, 2] co-designs consist of different aspects such as specification, modeling, partitioning, performance estimation, interface performance and co-simulation. This latter is the objective of our work [3]. It concerns essentially communicating and synchronizing two simulators such as matlab and systemC. Actually, these techniques are used in so varied applications such as the postproduction in television and cinema, the broadcasting and the video surveillance. Their speciality is the realization of solution of video coding with an objective of real time. SystemC [4], a modeling language is useful for representing the hardware and the software components of a system. SystemC is based on C++ and includes a set of classes to model hardware components and a simulation kernel. The design environment specifies software algorithmically as a set of functions located in modules. The MATLAB [5] environment is a high-level technical computing language for algorithm development, data visualization, data analysis and numerical computing. One of the key features of this tool is the integration ability with other languages and third-party applications. MATLAB also includes the Simulink graphical environment used for multi-domain simulation and model-based design. Signal processing designers take advantage of Simulink as it offers a good platform for preliminary algorithmic exploration and optimization.

This paper presents a structure of check based on a new interface of co-simulation between systemC and simulink of matlab environment to improve the bottleneck of check of material. We are going to test our platform on video compression JPEG algorithm.

In this paper, we first discuss the related work in section 2 and in section 3, we present a methodology of co-simulation. In section 4, we propose the application of compression video JPEG. We give the results of co-simulation in section 5. Finally, we conclude and suggest some recommendations for future works.

## II. RELATED WORK

DSP [5] applications regularly requires complex environment to competently simulate and verify the design. For example real world telecommunication stimuli are needed to truly exercise the design. In addition, the criteria used to evaluate performances are quantitative measures such as least-mean-square, bit-error rate, and others. It is little flexible and time consuming to implement these criteria directly into HDL or C++ testbenches. There are three main components for verification framework: SystemC verification standard, Transaction based verification and MATLAB and Simulink.

The principal core element of the verification platform is SystemC. It is important to understand that SystemC is the language to use whether for the conception or the check. Test code is written with SystemC to produce scalable and intelligent testbenches. Since the design below verification can be represented at multiple levels of abstraction, transactors are used to bridge the abstraction gap. Creating and maintaining testbenches in a higher abstraction level is naturally faster than HDL simulator and can be reused across abstraction levels.

Next, we use MATLAB and Simulink in the verification flow. The verification framework takes this into consideration and uses Mathworks tool to assist the SystemC testbench. By this way, each tool and language is used for its intended purpose.

## III. METHODOLOGIE

Using MATLAB and Simulink to assist SystemC verification framework relies on co-simulating the two environments. The co-simulation interface must give adequate capabilities and reasonable simulation speeds. Our solution is based on the MATLAB engine interface. The data

are directly exchanged by the memory on both sides. Both simulators obtain the optimal speed by guarding the interface and the protocol as simple as possible. This section presents the implementation details of this interface.

### A. *SystemC calls MATLAB*

The base of our platform is the transfer of data between SystemC and MATLAB [6]. For that reason, we use the 'engine' library that is available with MATLAB. The work described uses a similar interface. The difference is that we employ this interface in a verification context. In addition, we will show in the following section how we significantly improve the interface to communicate with Simulink.

The 'engine' library contains nine routines for controlling MATLAB computation engine from a C program. On Microsoft Windows, the engine library communicates with MATLAB using a Component Object Model (COM) interface.

A SystemC module employs these routines to remotely control MATLAB and exchange data back and forth between SystemC and MATLAB workspace.

### B. *SystemC calls Simulink*

To exchange data between a Simulink model and SystemC module, the co-simulation interface must integrate a bridge between the two simulators [6]. This bridge is built with two Simulink S-Functions. An S-Function is a computer language description of a Simulink block. It uses syntax of call thus we can interact with Simulink solvers. For our bridge, we create two C++ S- Functions. Figure 1 gives an overview of how a Simulink model and S-Functions are connected together.

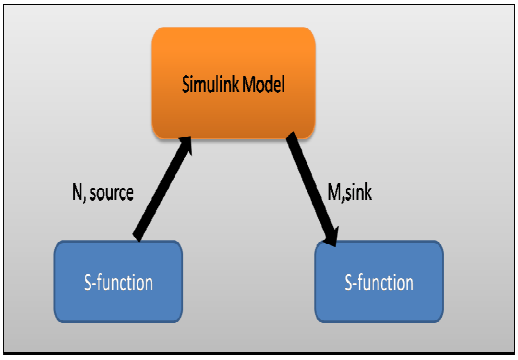



Fig .1. The model with n-input source and m-output sink in s-function blocks

The 'source' S-Function reads data from MATLAB workspace [4] written with SystemC and sends the corresponding signals in Simulink. On the other hand, the 'sink' S-Function reads its inputs and update the corresponding variables in MATLAB workspace so it can be read back by SystemC. Equally, S-Functions includes a configurable burst mode option to support variable width data burst transfers.

### C. *Simulator synchronization*

The representation of simulation time differs significantly from SystemC and Matlab. SystemC is cycle-based simulator and simulation occurs at multiples of the SystemC resolution limit. The default time resolution is 1 picoseconds; this can be changed with function sc_set_time_resolution. Simulink maintains simulation time as a double-precision value scaled to seconds. Our co-simulation interface uses a one-to-one correspondence between simulation time in Simulink and SystemC. The Simulink solver is set to discrete fixed-step type, this means that one time step in Simulink correspond to one tick in SystemC.

As mentioned previously, SystemC is the boss of the simulation and Simulink is controlled from SystemC by the interface of co-simulation. SystemC uses set_param to start, stop and continue the execution of Simulink. The simulation is suspended every time by S-Function. The same command set_param is used at the end of the S- Function, but with the argument 'pause'. On the other hand, SystemC asks the status Simulink with the command get_param to synchronize both simulators.

## IV. JPEG COMPRESSION ALGORITHM

The JPEG encoder consists of modules for color converting, DCT (Discrete Cosine Transformation), quantizing, and encoding [7]. The DCT processes two-dimensional still images as a combination of two-dimensional frequency components to compress image data.

The JPEG encoder requires repetitive DCT calculations which are time consuming. Therefore, DCT calculation was selected as a target for replacing by hardware. The JPEG software is used in the experiment and based on the source program distributed by the independent JPEG group (IJG) [8]. We extracted the minimum set of the JPEG encoder functions and prepared the SW modules by SystemC in advance. The RGB format images were used in this experiment.

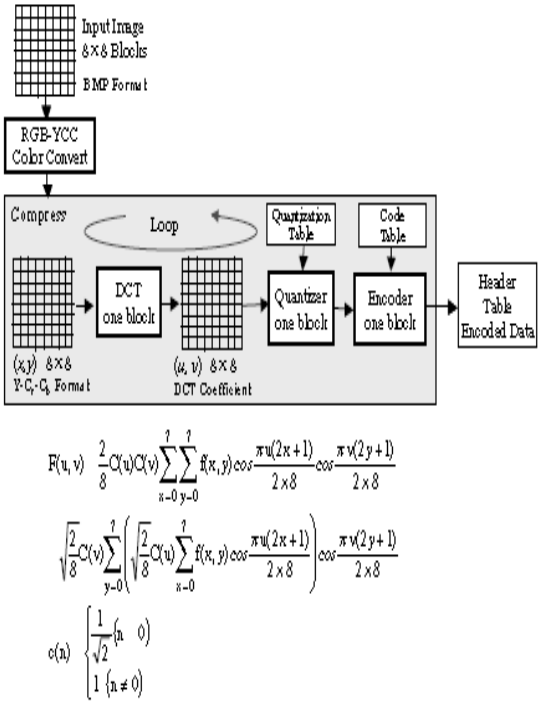
- RGB-YCbCr color conversion: A 16*16 RGB image is converted into four 8*8 blocks of Y (luminosity) and two 8*8 blocks of Cr and Cb (color difference).

- Calculation of DCT coefficient: Two-dimensional DCT coefficient F(u,v) corresponding to each Y, Cr and Cb are obtained using one-dimensional DCT calculation twice using the formula shown in Figure2.

- Quantizer: Each DCT coefficient is divided by the corresponding value in the quantization table to remove high-frequency ingredients.

- Huffman encoder: DCT coefficient information is compressed and the JPEG picture form is edited by adding a quantization table, frame header, etc.

In this chain, we have made many calculations.




$$F(u,v) \quad \frac{2}{8}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{\pi u(2x+1)}{2\times 8}\cos\frac{\pi v(2y+1)}{2\times 8}$$

$$\sqrt{\frac{2}{8}}C(v)\sum_{y=0}^{7}\left(\sqrt{\frac{2}{8}}C(u)\sum_{x=0}^{7} f(x,y)\cos\frac{\pi u(2x+1)}{2\times 8}\right)\cos\frac{\pi v(2y+1)}{2\times 8}$$

$$c(n)\begin{cases}\frac{1}{\sqrt{2}} & (n=0)\\ 1 & (n\neq 0)\end{cases}$$

Fig. 2. Configuration of JPEG encoder including DCT

## V. RESULTS

SystemC is an open source simulator . Its language represents an extension by classes of the directed language object C ++ for the description of digital systems. It allows the description level RTL, as it is the system level (SystemC 2.0 and the later versions) for the systems implemented in software, hardware or the combination of both. A SystemC model can be compiled, executed and debugged by using the standard tools of C++ programming. Compared with other languages of hardware description such as VHDL and Verilog, SystemC supports a large number of description domains which allow it to supply a specification of the system at high levels of abstraction and with a better speed of simulation. To create a configurable and customizable block, we are going to use S - Function. This latter supplies a powerful mechanism to spread the capacities of Simulink. The S- Function is then used as any block of the Simulink library.

The S - functions uses a special call syntax which allows interacting with the engine of resolution of Simulink equations. This interaction is very similar to the interaction between the engine and the other Simulink blocks.

With the built-in debugger, it is possible to attach the MATLAB process to the current SystemC simulation.

The mult16 function of implemented SW/SW takes 3753 clock cycles. It consumes a total of 1114641 clock cycles. These major accounts of clock cycle represent 82.89 % of the total time execution of the JPEG algorithm.

The figure 3 shows the implementation HW/SW of the JPEG proposed architecture. This implementation has a treatment unit to execute the software part of the DCT algorithm and a hardware module to execute multiplications of the JPEG compression algorithm. The processing unit has a microprocessor, RAM and ROM.

The modified portion of the DCT algorithm without multiplications is stored into the ROM.

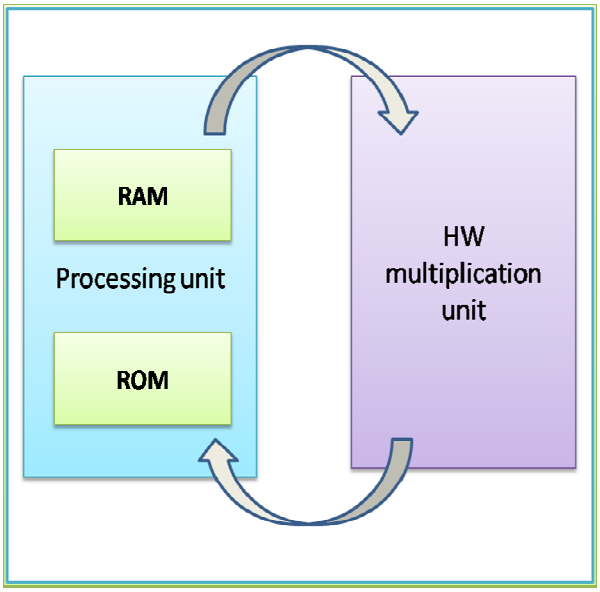



Fig .3. Block diagram of the proposed HW/SW jpeg implementation

The HW/SW implementation reduces the number of clock cycle from 1334722 to 200209.The implementation of the mult16 function in hardware reduced by 99.5% the number of clock cycles required to perform the 16-bit multiplication. The reduction on the total execution time of the JPEG algorithm is 85%.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a check structure based on a new co-simulation interface between SystemC, MATLAB and the Simulink environment. This platform can be used to help the check of DSP cores design. By using MALTAB and Simulink environment, we improved the SystemC capability of material check and the time to market decrease.

The proposed plateform is tested on the JPEG compression algorithm. The execution time of such algorithm is improved by 85% due to the hardware implementation of the Matlab mult16 Function using SystemC. As future works, we aim to test our platform with the whole video compression chain using MPEG4 modules.

## VII. REFERENCES


[1] A. Avila, “Hardware/Software Implementation of a Discrete Cosine Transform Algorithm Using SystemC” Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2005)

[2] M.Abid, A. Changuel, A. Jerraya,” Exploration of Hardware/Software Design Space through a Codesign of Robot Arm Controller” EURO-DAC '96 with EURO-VHDL '96 pp 17-24

[3] L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, M. Poncino, “SystemC Cosimulation and Emulation of Multiprocessor SoC designs,” Computer Magazine, April 2003 pp: 53 – 59

[4] The Open SystemC Initiative (OSCI) http://www.systemc.org

[5] J.F. Boland “Using MATLAB and Simulink in a SystemC Verification Environment”, Proc. of Design and Verification Conference & Exhibition, San Jose, Californie, Février 2005

[6] C. Warwick, “SystemC calls MATLAB”, MATLAB Central, March 2003, http://www.mathworks.com/matlabcentral/

[7] Hiroyasu Mitsui “A Student Experiment Method for Learning the Basics of Embedded Software Development Including HW/SW Co-design” 22nd International Conference on Advanced Information Networking and Applications – Workshops 2008 pp.1367-1376

[8] Independent JPEG Group, http://www.ijg.org