World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:7, No:2, 2013

SystemC Modeling of Adaptive Least Mean Square Filter

Kyu Han Kim, Soon Kyu Kwon, Heung Sun Yoon, and Jong Tae Kim

***Abstract*—**In this paper, we demonstrate the adaptive least-mean-square (LMS) filter modeling using SystemC. SystemC is a modeling language that allows designer to model both hardware and software component and makes it possible to design from high level system of abstraction to low level system of abstraction. We produced five adaptive least-mean-square filter models that are classed as five abstraction levels using SystemC proceeding from the abstract model to the more concrete model.

***Keywords*—**Adaptive Filter, Least-Mean-Square Algorithm, SystemC, Transversal Fir Filter.

I. Introduction

THE development of System-on-Chip(SoC) technology induced many high-end portable devices, such as mp3 player, smart phone and so on. Most portable devices are made more compact, but have high performance, and more increase complexity. With this increasing complexity of system design, it is one of the most important things to boost system design productivity to meet the time-to-market and reduce product design costs. Time-to-market is the factor that manufacturer put a premium on and under situation that increasing complexity of device, we must concern about system level design. Because the beginning stage of high level system design bring to its final development result, in other words degree of completion, which include performance, power consumption and development cost, etc. For this reason, many engineers employs SystemC language as the system modeling tools that are widely used to simulate architecture and based on C++ language easy to use. SystemC is a modeling language that allows designer to model both hardware and software component and this is possible to design from high level system of abstraction to low level system of abstraction. SystemC language is more suitable than Hardware Description Languages (HDLs) such as VHDL and Verilog at the beginning of product design stage.

This paper presents the result of adaptive least mean square (LMS) filter modeling for various models according to different level of abstraction using SystemC modeling language. The adaptive LMS filter is the digital filter module that frequently used in echo cancellation. We implement the adaptive LMS filter that is subdivided into 5 case models to inquire SystemC modeling characteristic according various abstraction level and simulate each model in SystemC testbench module and determine execution speed each other.

The remainder of this paper is composed as follows. Section IIdiscusses related work. In section III, we present implementation for adaptive LMS filter modeling. And section IV, we discuss and present model simulation results. Finally, we conclude this paper in section V.

Kyu-Han KIM is with the Department of IT Convergence, the college of Information and Communication Engineering, Sungkyunkwan University, Chun-chun-dong, Jangan-gu, Suwon, Gyunggi-do, Korea (e-mail:sgkh132@ skku.edu).

Soon-Kyu Kwon, is with the college of Information and Communication Engineering, Sungkyunkwan University, Chun-chun-dong, Jangan-gu, Suwon, Gyunggi-do, Korea (e-mail:caesar01@ skku.edu).

Heung Sun Yoon is with the college of Information and Communication Engineering, Sungkyunkwan University, Chun-chun-dong, Jangan-gu, Suwon, Gyunggi-do, Korea (e-mail:y11010s@ skku.edu).

Jong Tae Kim is the professor in the college of Information and Communication Engineering, Sungkyunkwan University, Chun-chun-dong, Jangan-gu, Suwon, Gyunggi-do, Korea (e-mail:jtkim@ skku.edu).

II. Related Work

A. Transversal Fir Filter

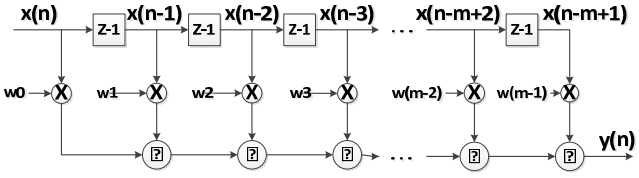

Fig. 1 Transversal Fir Filter

Fig.1 shows a block diagram of a transversal Fir filter, the input vectors are denoted by x(n), the filter order is denoted by M, and Z^{-1} is denoted by a delay [1]. Adaptive filter utilizes to minimize a value that the difference between a desired output and an actual output of FIR filter.

The characteristic of a transversal FIR filter can be expressed as tap weight, w(n). The tap weight represents the impulse response of FIR filter. The output of the FIR filter at time n is determined by the sum of products between w(n) and delayed input value x(n).

B. The Least-Mean-Square (LMS) Algorithm

The Least-Mean-Square (LMS) Algorithm is one of the most widely using algorithms in adaptive filtering. It is well known and is widely used due to its computational simplicity. With iteration of the LMS algorithm, the filter coefficient, filter tap weights, is updated. The LMS Algorithm is presented by "(1)".

$$W(n+1) = W(n) + \mu e(n) X(n) \quad (1)$$

The *X(n)* is the input vector of time delayed input, and *W(n)* is the coefficients of the adaptive filter tap weight vector at time

n. The μ is the parameter represented as the step size. This step size controls the influence of the updating factor.

C. Adaptive LMS filter

The adaptive Least-Mean-Square (LMS) filter operate in the manner of reducing error signal that difference between the desired signal and the output signal that passed through adaptive filter to converge specific value, zero, and update filter tap weights with iteration. Fig.2 shows the block diagram that represents adaptive LMS filter applied to the LMS algorithm [2]. The adaptive LMS filter generates the output signal (Y[n]) from the input signal(X[n]) and the error signal (e[n]) to subtract the output signal (Y[n]) from the desired signal (d[n]). After that, to use the error signal (e[n]) and the input signal(X[n]), the updated filter tap weights apply to the adaptive LSM filter repeatedly in coefficient update module.

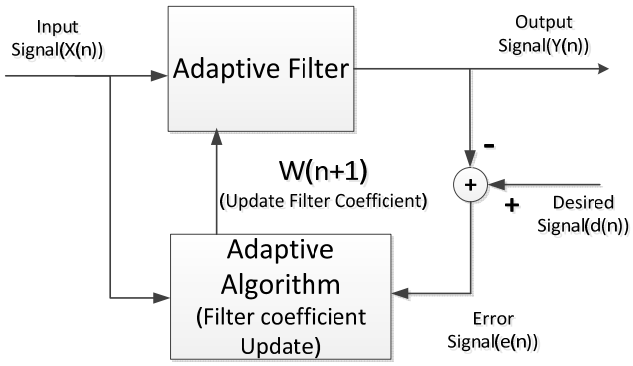

Fig. 2 Adaptive LMS filter block diagram

III. Implementation

A. Tools & Installation

SystemC language utilizes C++ language environment like Visual Studio. SystemC is a kind of library that easily imports and employs for systematic modeling. SystemC language has some characteristic proper to high level system development that do not have in C++ language and also has an advantage of C++ language, especially reuse based on C++ class. Because of the feature that we mentioned before, SystemC is the system modeling language that improves the system design productivity definitely. SystemC is a library of C++ language, so it can be easily used to download SystemC library and import library in C++ language program. SystemC environment that is used in this paper is SystemC version 2.3.0 and C++ language program is the Visual Studio 2005 that use widely in various purpose. Also, MATLAB can be used in modeling verification and simulation.

MATLAB is a high-performance language for the mathematical computation, and have many tool-box that predefined various digital filter to use without difficulty. And, also, there are many visualized functions and commands that are easy to utilize verification and simulation for SystemC modules in MATLAB as a graph to visualize the result data intuitively. Because of that reason, it is easy to apply the SystemC model functional verification using MATLAB. The adaptive LMS filter verification model in this paper is also applied to MATLAB visualizing functions.

B. Modeling

We tried to follow a typical model proceeding from the abstract model to the more concrete model. We produced five adaptive LMS filter models that are classed as five abstraction levels using SystemC [3].

Model 1 is a C language model case in SystemC environment. Model 2 is a SystemC module implemented by C language based on adaptive Least-Mean-Square algorithm. Model 3 was developed as more concrete model based on model 2, especially added input and output buffer module that save the data generated by testbench module as the input signal of adaptive LMS filter and collect the data passed adaptive LMS filter as the output signal of adaptive LMS filter. Model 4 is partitioning model using model 3, divided adaptive LMS filter module into data processing module and control unit. Model 5 utilizes arm core processor for simulation to investigate the testing result that run the adaptive LMS filter in arm processor environment.

We design the verification model about adaptive LMS filter by combining SystemC and MATALB to simply verify and visually simulate [4]. The signals, which input signal(x[n]) and desired signal(d[n]) using as an test input of adaptive LMS filter, is generated by MATLAB random number generate function.

That signals that are stored as the .mat file format data that use frequently in MATLAB are provided in a way that testbench module read the signals as test inputs. And, output signal (y[n]) data and error signal (e[n]) data that are exported by testbench module are re-stored as the .mat file format data. In MATLAB, the output signal file that was saved before is visualized in the 2D graph. It is easy to utilize reading and writing the .mat format file in SystemC testbench module by calling the MATLAB API functions and MATLAB provide the method that can use the MATLAB API function in C++ environment. We convert particular MATLAB API functions that are used in reading and writing the .mat format files to the C++ library that is able to apply in SystemC environment.

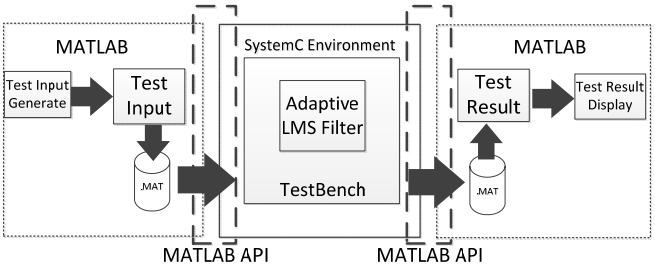

Fig. 3 SystemC model simulation flow

C. Model Description

Model 1 – C language modeling Model 1 is an output of adaptive LMS algorithm using C programming language to demonstrate functionality in SystemC environment. We

organize an adaptive LMS filter represented by Fig.1 and (1) based on behavioral modeling methodology following data flow before starting SystemC modeling.

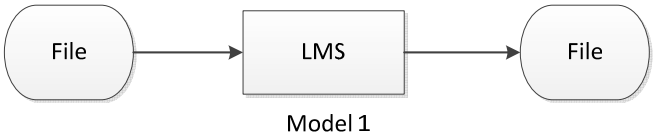

Fig. 4 Model 1 block diagram

Model 2 – SystemC Modeling (LMS Filter Module) Model 2 is an implement that designs an adaptive LMS filter as a SystemC module based on adaptive LMS algorithm that verifies functionality in model 1. The function of adaptive LMS filter module is designed by *Thread* that checks the positive edge of clocks and synchronizes with positive edge of clock. The tap size of adaptive LMS filter are 32tap, and the step size of filter parameter set up the value to 0.005. Because adaptive LMS filter using SystemC synchronize by the clock, we investigate the execution time of the designed filter module in clock unit. We simulate model 2 using SystemC testbench module and find the execution speed for model 2.

Model 2 consumes 6 clocks per one data sample that pass through and operate the adaptive LMS filter in one cycle.

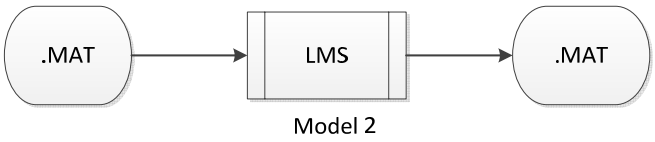

Fig. 5 Model 2 block diagram

Model 3 – SystemC Modeling (Partitioning Modules) Model 3 is more concrete model than model 2. Model 3 is a SystemC designed model that adds the input and output buffer module based on model 2 that was developed by only one adaptive LMS filter module. The input buffer is a temporal data storage that saves test signals from testbench module and the output buffer is also the temporal storage that save test result signal, y(n) and e(n), which hand over to testbench module.

The input and output buffers are implemented as *Thread* that detect and synchronize the positive edge of clock. In comparison with model 2, the execution speed increase about 2 clocks per one filter operation cycle, because of the input and output buffer module. But, as a consequence of adding buffer modules, we are able to get control of the test input and output used in the adaptive LMS filter and can simulate more detail and sophisticated manner for the operation of an adaptive LMS filter in various circumstances.

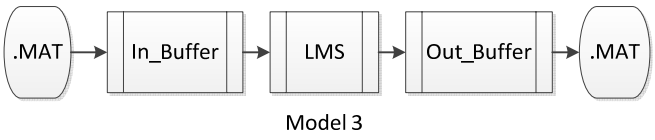

Fig. 6 Model 3 block diagram

Model 4 – SystemC Modeling (DU/CU Partitioning Model)

Model 4 is the existing model 3 to add control unit that can control the behavior of the adaptive LMS filter. The control unit control the behavior of the adaptive LMS filter to stop if error signal that the output signal of adaptive LMS filter converge within a certain range or radiate out of a certain range. When the adaptive LMS filter to work for the first time, the input buffer module sends the signal to the control unit, input data that it came. When the control unit receives the signal, it sends signals to each module that can control the behavior of each module to initialize the adaptive LMS filter to operate properly. And then, the control unit if the error signal, output of adaptive LMS filter, converge within a certain range or radiate out of a certain range that predefined in a control unit module as a fix value. The control unit controls the entire data flow by sending signals to control each module to stop the operation of the adaptive LMS filter.

The control unit is built as *Thread* that detects and synchronizes signals that was sent from input buffer and LMS filter module, and also operate according to the positive edge of clock. In comparison with model 3, the execution speed increase about 2 ~ 8clocks per one LMS filter operation cycle, because of control unit module and functions that control error signal. However, as a consequence of adding control unit module, we can have more controls for the operation of adaptive LMS filter. We are able to test more various and detail operation of an adaptive LMS filter in various circumstances, such as converging error signal within a certain range or diverging error signal out of a certain range.

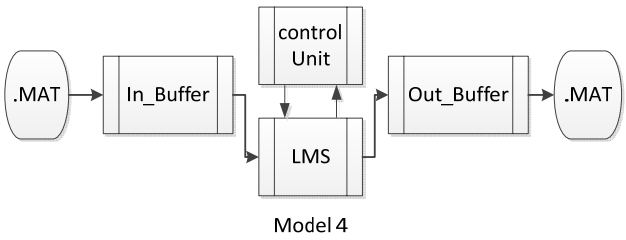

Fig. 7 Model 4 block diagram

Model 5 – Processor Model

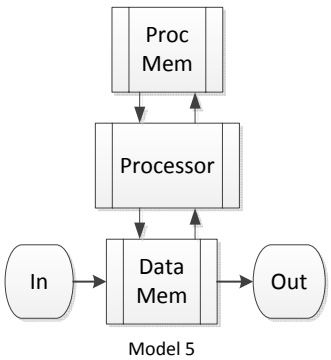

Fig. 8 Simulation result (Model 4 error signal)

Model 5 is the processor model having components such as data memory, processor memory and processor. ARM9 processor is used at the target processor. We have an object binary file made C source code of model 1 using arm-elf-gcc compiler. We analyze an instruction code how adaptive LMS filter behave at instruction code level in arm-core processor environment and calculate how many clocks are required to operate an adaptive LMS filter module.

The adaptive LMS filter actually requires 207 clocks per one adaptive LMS filter operation cycle and 57150 clocks per a thousand adaptive LMS filter operation cycle.

IV. Simulation Results

The purpose of adaptive LMS filter is made the output signal similarly to the desired signal as reducing the error. Accordingly, if the error signal converges to zero, we can consider that the adaptive LMS filter works properly.

We show graph that represent the error signal of model 4 on MATLAB and SystemC each other. The upper in Fig.10 is the error signal on MATLAB implement and the lower one, on SystemC implement. The upper is similar to the lower and both converge close to zero. This shows that the adaptive LMS filter is working properly and well designed.

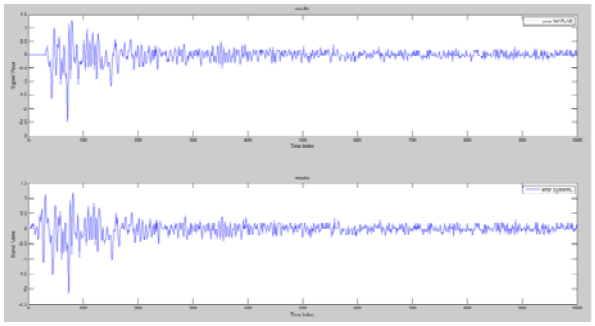

Fig. 9 Simulation result (Model 4 error signal)

As previously mentioned in model description, the adaptive LMS filter in model 4 was designed to stop filter operation, if the error signal converges within a desired range or diverges out of a certain range.

These pictures, Fig.10 and Fig.11, are the error signal in the case of convergence and divergence. When the error signal enter the range between +0.005 and -0.005 the control unit sends a signal to the LMS filter module to stop running, as you can see in Fig.10. In the same manner, the control unit send the stop signal to the LMS filter module when the error signal radiate beyond the range between +1.5 and -1.5.

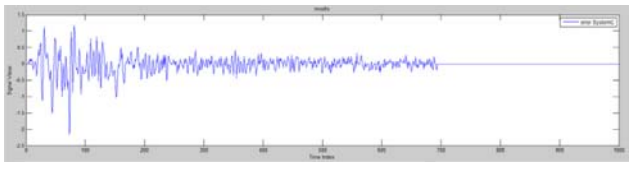

Fig. 10 Control convergence (Model 4 error signal)

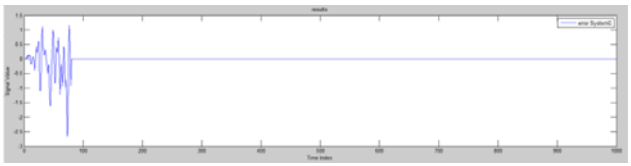

Fig. 11 Control divergence (Model 4 error signal)

TABLE I
Execution speed for each model

Model	1 cycle per execution time	1000 cycle per execution time
2	**6 clock**	**6000 clock**
3	**8 clock**	**8000 clock**
4	**12 clock**	**12000 clock**
5	**207 clock**	**57150 clock**

cycle : a cycle of adaptive LMS filter running

Table I shows the execution speed of adaptive LMS filter model 2, 3, 4, and 5. Data represent the number of clocks during 1 cycle and 1000 cycles of the adaptive LMS filter running, each model.

Model 2 implemented as a simple behavioral filter module just applied to adaptive least-mean-square algorithm shows fast execution speed than other models. Because it has a high abstraction level, the implementation is not so detail that clocks do not consume to do so. On the other hand, in the case of model 5 design a detail to the most, the adaptive LMS filter consumes 207 clocks per 1 cycle of filter running.

More simply designed model with high abstraction has short execution time tends to increase simulation speed. Even so, the case of a model with high degree of abstraction has only ability to verify the functionality of the system. On the other hand, the case of model 4 can be derived the clock accuracy simulation result in a specific behavior or situation, as well as verification of the functionality, because of partitioning that divide the adaptive LMS filter into the control unit and the data processing unit.

V. Conclusion

We design the adaptive least-mean-square filter according to five different level of abstraction using SystemC. From the upper level model to verify the algorithm to the lower level model to be able to control with clock accuracy, we could design adaptive LMS filter according to the level of a variety abstraction using SystemC.

While comparing the execution speed varied depending on each abstraction level with the degree of filter controllability, we can have an opportunity to try to take advantage of a variety of SystemC. SystemC can be used variously from only to verify the functionality of the system to models that can be simulated detail under certain circumstances and models based on a certain scenario in various aspects of verification. Through proper system design and simulation satisfied high level system spec, we will be able to obtain an important result to determine design direction used in low-level system design in the subsequent. There by, in terms of the overall system design, we will be able to lower the development cost and time-to-market

demand will be met.

REFERENCES

[1] Micheal Hutson, "Acoustic Echo Cancellation Using Digital Signal Processing", submitted for the degree of Bachelor Engineering, Nov. 2003
[2] Christian Feldbauer, Franz Pernkopf, and Erhard Rank, "Adaptive Filters", A Tutorial for the Course Computational Intelligence ,Signal Processing and Speech Communication Laboratory
[3] J.R.Armstrong and Yuval Ronen, "Modeling With SystemC : A Case Study"
[4] Xu Ningyi and Zhou Zucheng, "A Methodology for SystemC Algorithmic Model Verification Applying MATLAB" , *ASIC, 2003.Proceedings.International Conference On,* volume 1, pp.294-297, Oct. 2003