Improvement in Speed of Hardware Adaptive Filter

Ankur, Veenadevi S. V.[*]

Dept. of Electronics and Communication Engineering
RV College of Engineering, Bengaluru

Abstract

This paper deals with design modification of adaptive filter to enhance the performance of an audio chip which is widely adopted for digital signal processing. The Register Abstraction layer was implemented to provide access to registers directly by name and address. The Universal verification methodology was adopted to simulate the functional verification of digital hardware. The simulation resulted in reduction of 50% of redundant clock cycles which prevents the stopping of jobs midway in a low power and high speed application. Performance and debug registers present in the existing design provided number of clock cycles per job. Even marginal gains were quite useful when the amount of changes to Register Transfer Logic were decreased.

Keywords: Audio Chip Device, Universal verification Methodology, Register Abstraction layer

1.0 Introduction

Adaptive filter is an important component of audio chip for digital signal processing (DSP) and has wide variety of applications [1]. To provide a quality experience regardless of the environment, adaptive filters require a Go To approach. An Audio subsystem requires a lot of resources to perform adaptive filtering and offload the burden from the main digital signal processor. Hence A hardware accelerator consisting of specialized blocks for performing adaptive filtering can be utilized to provide a substantial performance boost.

Adopting Register Abstraction Layer (RAL) makes it possible to access registers directly by name and address. This allows easy manipulation of thread related registers which can verify whether a potential enhancement leads to less wastage of clock cycles. Adaptive filters, because of their ability [2] to operate satisfactorily in mobile environments, have become an important part of DSP applications where the characteristics of the incoming signals are unstable. A few examples include echo cancellation,

---

[*]*Mail address: Veenadevi S V, Associate Professor, Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru – 59*
Email: veenadevi@rvce.edu.in, Ph: 8762206268

adaptive beam-forming and channel equalization. The basic functionality comes down to the adaptive filter performing a range of use-cases, namely, inverse system identification, system identification, noise cancellation and prediction. Adaptive filters, because of their ability to operate satisfactorily in non-stationary environments, have become an important part of DSP applications where the statistics of the incoming signals are unknown or changing [3, 4]. Obtaining optimal design usually requires prior knowledge of certain statistical parameters (mean and correlation) within the useful signal. One of the popular measures is known as least mean squared error minimization in which the difference between the statistical measures of actual and the expected signal is obtained. Hardware Description Language (HDL) like verilog, system verilog, etc. is used as a base to synthesize hardware to create a functioning filter. Multipliers, Dividers, Multiply Accumulate (MAC) units are all coded in HDL. Communication between different modules is also coded in HDL. HDLs are also used in implementation of Finite State Machine (FSM), which makes up an integral part of most Register Transfer Logic (RTL) based designs.

UVM (Universal verification Methodology) is a methodology [5] for the functional verification of digital hardware, primarily using simulation. The hardware or system to be verified would typically be described using Verilog, SystemVerilog, VHDL or System C at any appropriate abstraction level. It may be behavioral, register transfer level, or gate level. UVM is explicitly simulation-oriented, but UVM can also be used together with assertion-based verification, hardware acceleration or emulation. In UVM, there is a mechanism to be followed when it sends to the transactions from the sequencer to the driver in order to provide stimulus to the DUT. A particular sequence is directed to run on a sequencer which in turn breaks down into a series of transaction items and these need to be transferred to the driver where these transaction items are converted into cycle based signal/pin level transitions.

2.0 Design Methodology and Implementation

2.1 Current Block Design

Fig. 1 shows a simplified block diagram of the currentdesign. It consists of a system wide DDR (Double Data Rate memory), Slaveinterface, NoC, Internal RAM, DMA and core. The processor is a core on which software or firmware is supposed to run. The following steps are adopted for the design:

i) The software invokes Hardware Adaptive Filter (HW- AF) functionality and data transfer takes place between core and DDR (Double Data Rate memory).

ii) The core transmits and also interrupts the signal to slave interface.

iii) The slave transfers the data from DDR to its own internal memory.

iv) The processing begins in the filter core and slave transmits to master.

v) The master retrieves the processed data from the DDR.

Some of the features of the current design include:

- Debug registers and one of the threads of debug registers is to count the clock cycles occurred per thread.
- The core is allowed to submit four threads at a time and once a thread is submitted, it can be in the state of pending, progress or completed. After the submission, the order of thread execution is thread1 → thread2 → thread3→ thread4
- A thread can be cancelled only if it is in pending state
- Initiates the actual thread by writing to GO/GO thread2/GO thread3/GO thread4
- Threads can be submitted into queue if it is not in the state of pending or progress.

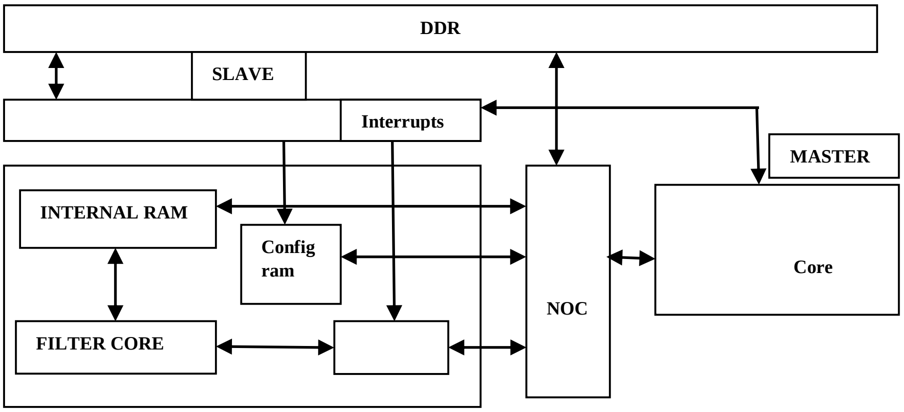

Fig. 1. Simplified Block Diagram

A schematic representation of design procedure is as shown in Fig. 2. After running all test cases, the design and data flow during one thread cycle can be ensured. Once the patterns are analyzed, it becomes necessary to visualize the parts of the design that need to be improved. If the performance enhancement reduces, the number of clock cycles requires less amount of changes in RTL. The statistical measures depends on the number of files edited and number of modules, and debugging effort time. After implementation of improvement through changes in RTL, a test bench is prepared to check the functional correctness of the applied change. Also one can again run the test cases to see if changes have not caused anything to break in the entire design.

Some of the possible avenues to explore for improvement include interruption of midway thread, running two threads simultaneously, compression of filter coefficients, and checking of gating using power analysis tools. When the improvements are performed, they can be again verified by changing the registers appropriately and new scenarios must be coded onto the reference design. Also some combinational logic is altered, like conditions for interrupts to trigger, certain enable and disable signals that were being used to track the status of jobs. Hence, performance enhancement must be considered only if it is providing a drastic variation for small changes in RTL.

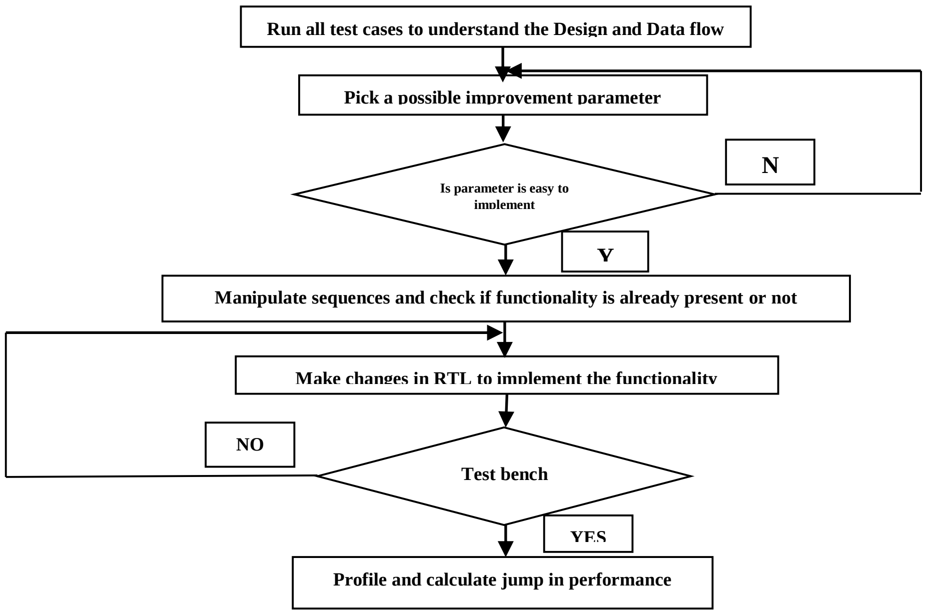
Run all test cases to understand the Design and Data flow
Pick a possible improvement parameter
Is parameter is easy to implement
N
Y
Manipulate sequences and check if functionality is already present or not
Make changes in RTL to implement the functionality
NO
Test bench
YES
Profile and calculate jump in performance

Fig. 2. Filter Design Methodology

2.2 UVM Register Model

The UVM Register layer provides standard base class libraries that enable users to implement the object-oriented model to access the DUT (Design under Test) registers and memories. UVM Register Layer is also referred to as UVM Register AbstractionLayer (UVM RAL) as shown in Fig. 3.

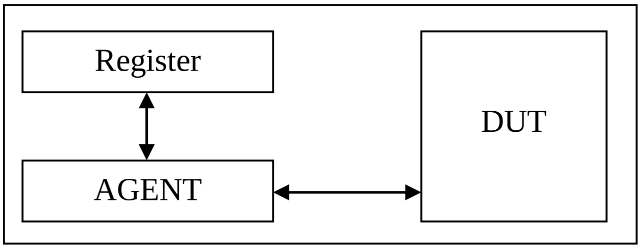

Fig. 3. Register Abstraction Layer

Some of the advantages of UVM RAL Model are:

Provides high-level abstraction for reading and writing DUT (Design under Test) registers. i.e. registers can be accessed with its names, address or reference.

UVM provides a register test sequence library containing predefined test cases and these can be adopted to verify the registers and memories. Register layer classes support front-door and back-door access.

Design registers can be accessed independently of the physical bus interface. i.e by calling read/write methods.

The register model can be accessed from multiple concurrent threads. It internally serializes the access to the register.

RAL packages can be directly reused in other environments. Defines the set of rules or methodology on register access, which can be followed across the industry.

Automated RAL model generations, tools or open-source scripts are available for RAL model generation.

2.3 Reference Model

Reference model can be considered as vital against which the UVM test bench compares values and determines functional correctness. One of the most critical parts of any verification environment is the expected output calculation [6]. UVM output is calculated by the reference model, which is normally implemented in System Verilog. The problem arises during designing a UVM environment for complex designs as the wireless baseband digital systems. The implementation of the reference model would be a bottleneck since it is complicated. The reference model also must be coded with a high degree of accuracy which might be very difficult to achieve using Hardware Description Languages (HDL) especially with bulky designs that involve sophisticated algorithms. Hence, the implementation of the design would not be coded in a straight forward readable manner using this approach. Also the Design Under Test (DUT) is already written in HDL, so a high-level language is more convenient to be used in a wide variety of applications. For instance, the communication and signal processing communities utilize MatLab for prototyping and delivering the abstraction models for the designs, while the video coding and image processing utilize C/C++ coding language.

3.0 Results and Discussion

The cancellation signal through UVM virtual sequencer is as shown in Fig. 4. The go1, go2, and go3 signals get triggered which then sets of thread1, thread2 and thread3 to execute sequentially. Once thread1 starts running, the other two, thread 2 and thread 3 are in pending state. A cancellation pulse is sent midway during thread execution. In this case, progress of thread 1 does not get affected because RTL (Register Transfer Logic) do not get and the functionality is currently not present.

Table 1. Thread Cancellation Improvement

	Before	After	Performance Jump
Clock Cycles per thread	2000000	2000000	0%
Clock Cycles Wasted	1000000	10	50%

The Thread Midway Cancel Successful case is as shown in Fig. 5. Once the go signal is given, thread goes into progress state. Once the cancel Pulse is given, the thread that was in progress is disabled. Thread Cancellation Successful for Use-case: The RTL change done to implement the feature have propagated and broken the whole system, however it did not happen and now feature must be tested for a use-case, Hence it was then simulated on an existing use-case with appropriate modifications. A use case being cancelled midway during its processing is as shown in Fig. 6.

Table 1 shows the amount of jump in performance. It can be seen that if a thread is not cancelled midway, this doesn’t offer any improvement, However if software chooses to preempt the thread for any reasons, Example higher prioritytask. On an average it assumed thread is stopped right inmiddle, in this case it will save 50% of the clock cycles that the thread taken to finish itself.

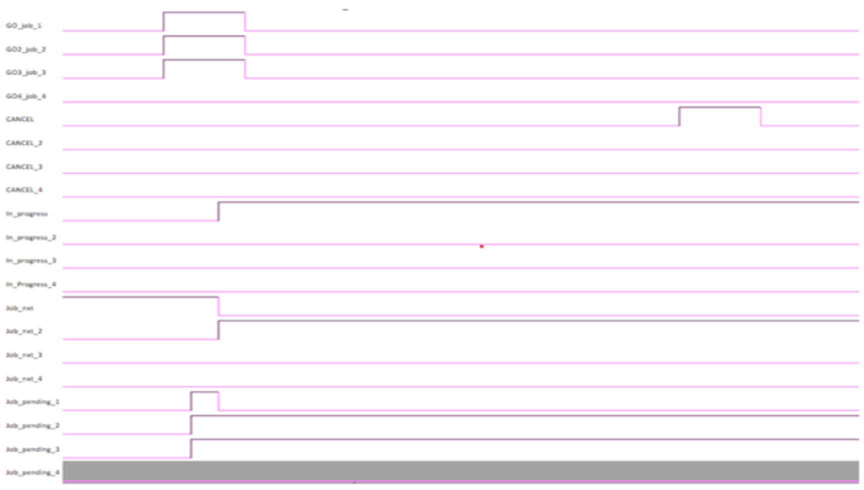

Fig. 4. Cancellation Signal Received

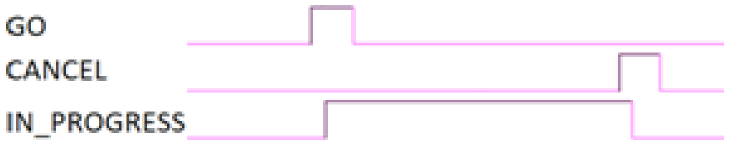

Fig. 5. Thread cancelled successfully

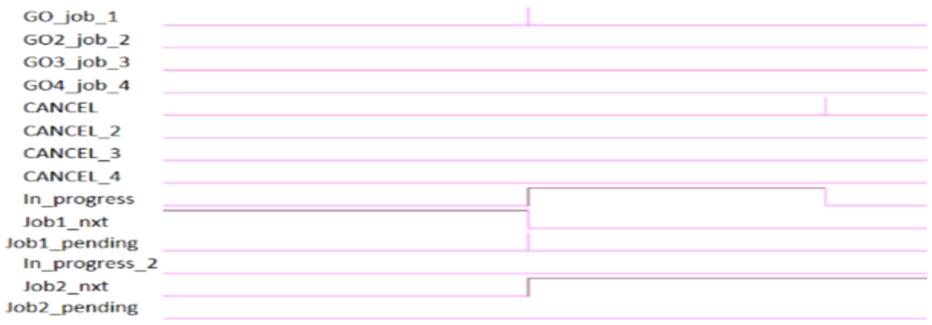

Fig. 6. Thread cancelled successfully for Use Case

4.0 Conclusion

The performance analysis of a newly designed adaptive filter was carried out. It can be inferred that the performance enhancement reduces the number of clock cycles and changes for RTL. Hence the amount of changes in UVM would require UVM test bench which mostly revolves around changes in the monitor and scoreboard. This improvement had the potential to save huge number of clock cycles, which matters a lot for a low power and high speed application. Performance and debug registers present in the existing design give the number of clock cycles taken per job. Even marginal gains were quite useful when the amount of changes to RTL were reduced.

References

1. D S Chen, P Y Chen, Y W Wang, Hardware/software co-design of nlms adaptive filters on FPGA, *IEEE 15th International Symposium on Consumer Electronics (ISCE)*, 442–445, 2011
2. P S R Diniz, Adaptive Filtering: Algorithms and Practical Implementation, *Springer Publications,* ISBN: ISBN 978-0-387-68606-6, 2008
3. Y Mollaei, Hardware implementation of adaptive filters, *IEEE Student Conference on Research andDevelopment (SCOReD),* 45–48, 2009
4. F Nekouei, N Z Talebi, Y S Kavian, A Mahani, FPGA implementation of LMS self correcting adaptivefilter (SCAF) and hardware analysis, *8th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP),* 1–5, 2012
5. A Moursi, R Samhoud, Y Kamal, S El-Ashry, A Shalaby, Different reference models for UVM environment to speed up the verification time", *IEEE Conference*, 67–72, 2018
6. W Ni, J Zhang, Research of reusability based on UVM verification, *IEEE 11th International Conference on ASIC (ASICON),* 1–4, 2015
7. A Jain, R Gupta, Scaling the UVM model towards automation and simplicity of use, *28th International Conference on VLSI Design*, 164–169, 2015