Improving Performance of Hardware Adaptive Filter

Ankur
RV college of engineering
ECE Dept.
Bengaluru
Email: ankur.ec17@rvce.edu.in

Dr Veena Devi
RV college of engineering
ECE Dept.
Bengaluru
Email: veenadevi@rvce.edu.in

***Abstract*—Adaptive Filter (AF) is a digital filter that has a transfer function that changes based on changes in the surroundings. Adaptive filters can adjust their weights using cost functions similar to a neural network. An implementation of adaptive filter in hardware allows it to have higher speed (Consumes lesser number of clock cycles) and hence also saving on power. A regular Digital Signal Processor (DSP) may also be employed to do the same but it will never come close to performance of a dedicated hardware. An improvement in this said hardware will directly boost the performance of all use-cases.**

Simulation of the existing design gives an idea of the current data flow and architecture. Exploring different potential improvements in design and then weighing the outcome gain vs effort to add the functionality is done. An improvement is chosen and implemented. Once it does the intended functionality, It is profiled to see the improvement in performance.

A large Filter task is divided into multiple threads. These are executed sequentially. In the current design a thread has 3 status registers to indicate whether it's in progress, pending, next. A scenario in which a certain thread needs to be cancelled, it can only be done if the thread is not already in progress, which will lead to wasted clock cycles. Hence the ability to stop a thread executing midway will save those clock cycles.

***Index Terms*—AF, Audio Device, Universal verification Methodology (UVM), Register Abstraction layer (RAL), Reference Model**

I. Introduction

Adaptive filter an important area in digital signal processing and Digital Signal processing is important on an audio chip. In many audio devices, this filter has wide array of applications. [1]All audio devices will never be present in the same environment for too long given this world filled with mobile hardware. To provide quality experience regardless of the environment, Adaptive filters are the Go To approach. An Audio subsystem requires a lot of resources to perform Adaptive filtering. Hence it is suitable to offload that burden from the main Digital signal Processor A hardware accelerator consisting of specialized blocks for performing adaptive filtering can be utilized. Improving this will provide a substantial performance boost.

Using RAL makes it possible to access registers directly by name and address. This allows manipulation of thread related registers with ease. Once that option is available, It is then used to verify whether a potential improvement is already present. Various avenues can be chosen for improvement.

II. Theoretical Concepts

[2]Adaptive filters, because of their ability 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 use-cases are echo cancellation, adaptive beam-forming, 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. [3]The fundamental function of a filter is to get rid of unwanted component of a signal from those of interest. Obtaining the simplest design usually requires a priori knowledge of certain statistical parameters (such because the mean and correlation functions) within the useful signal. With this information, an optimal filter can be designed which minimizes the unwanted signals according to some statistical criterion. One popular measure is known as least mean squared error minimization, Here the the difference between the actual and the expected signal's statistical measure's difference is obtained and squared, This value is then minimized by giving a nudge to the weights in the right direction.

Adaptive Filters are FIR filter with coefficients that are able to change based on the changes in the environment around it. These filters are very versatile in applications where statistics of incoming signals cannot be reliably determined.

An adaptive filters have many use-cases, [4]Some examples are in channel equalization, echo cancellation or adaptive beam-forming. The basic function comes down to the adaptive filter performing a range of different tasks, namely, system identification, inverse system identification, noise cancellation and prediction.

Hardware Description Language (HDL)s like verilog, systemverilog are used as a base to synthesize hardware to create a functioning filter. Multipliers, Dividers, Multiply Accumulate (MAC) units are all coded in in HDL. Communication between different modules are 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.

[5]UVM is a methodology 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 SystemC at any appropriate abstraction level. UVM has versatility to use other featurees

like assertions and RAL to design a testbench, UVM Allows re-use of testbench by also allowing purchase of standard VIPs (Verification IP) from vendors like cadence and synopsys In UVM, there is a mechanism to be followed when a transaction is to be sent from sequencer to driver. Sequences are sent to the driver using TLM(Transaction Level Modelling), Individual signal bit flips are sent in the form of packet.

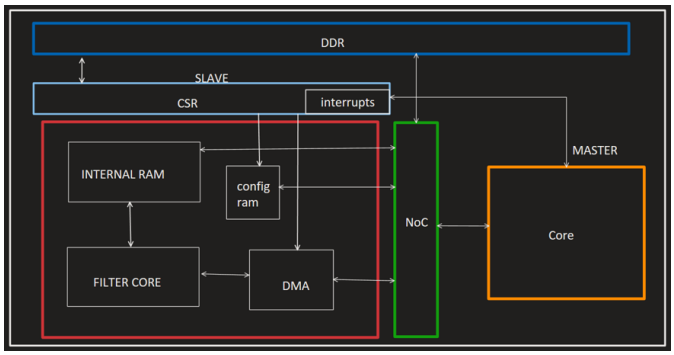

Fig. 1. Simplified Block Diagram

III. DESIGN METHODOLOGY AND IMPLEMENTATION

A. Current Block Design

Fig. 1 shows a simplified block diagram of the current design. The block consists of a system wide DDR, Slave interface, NoC, Internal RAM, DMA and core. The core is a processor on which software/firmware is supposed to run.

1) Once Software invokes Hardware Adaptive Filter (HW-AF) functionality, Data transfer between core and DDR takes place.
2) Once Data transfer is complete Core sends and interrupt signal to slave interface.
3) Slave will start transferring data from DDR to its own internal memory.
4) Processing begins in the filter core, once done, slave will send interrupt to master.
5) Master retrieves the processed data from the DDR

1) Features:

- Current design consists of debug registers
- One such debug register's thread is to count the clock cycles taken per thread.
- The core is allowed to submit 4 threads at a time
- Once a thread is submitted it can either be in pending, progress or done state
- Once 4 threads are submitted, The order of execution in thread1 $\longrightarrow$ thread2 $\longrightarrow$ thread3 $\longrightarrow$ thread4
- A thread can be cancelled only if it is in pending state
- Start the Actual thread by writing to GO/GO thread2/GO thread3/GO thread4
- threads can be submitted into queue if it is not In pending or progress state

The Fig. 2 shows a diagrammatic representation of design procedure followed in this project. Running all test cases gives a good idea about the design and how data flows during one thread cycle. Once the patterns are analyzed, It is necessary to see what parts of the design are not up to the mark, or have a potential to be improved. If said improvement requires less amount of changes in RTL that potential can be taken up for improvement. Once said improvement has been implemented through changes in RTL, A testbench is to be prepared to check the functional correctness of the applied change, Also one can again run the testcases to see if changes have not caused anything to break in the entire design.

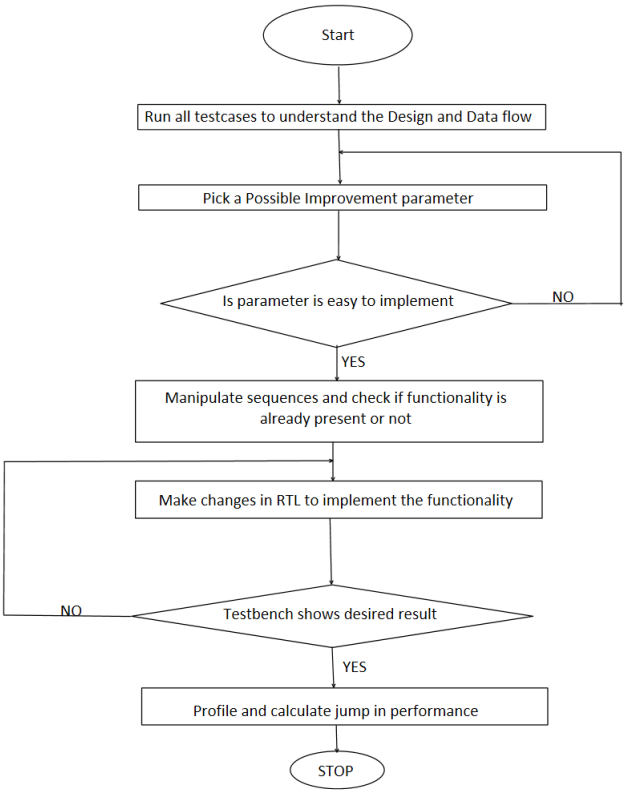

Fig. 2. Design Methodology Flow

B. Possible avenues to explore for improvement

- **Make it possible to interrupt thread midway**
- Possibility of running 2 threads simultaneously
- Possible compression of filter coefficients
- **Check clock gating(By using power analysis tools)**

If so, then a new possibility of improvement can be chosen. When Improvements are once performed, It can also be again verified by changing these registers appropriately. Once the design has been changed, New scenarios must be coded onto the reference design. However careful consideration must given to performance enhancement to implement, because if the enhancement requires alot of RTL changes, drastic changes must also be made to the reference model. Then the UVM monitor and scoreboard must also be modified accordingly. Hence a performance enhancement must be considered only if it is giving drastic improvement for small RTL changes. Which in turn lead to less changes to reference model.

Example, thread2 is in pending state (ie. It will be exercised soon after thread already in progress completes)Then new thread cannot be put onto thread2 buffer

C. UVM Register Model

UVM register model, also known as RAL(Register Abstraction Layer) The UVM RAL offers a standard base class from which users can inherit the functions that are able to access DUT registers and memories. RAL is not a necessity, however it makes testbench design a lot less cumbersome and more easily re-usable

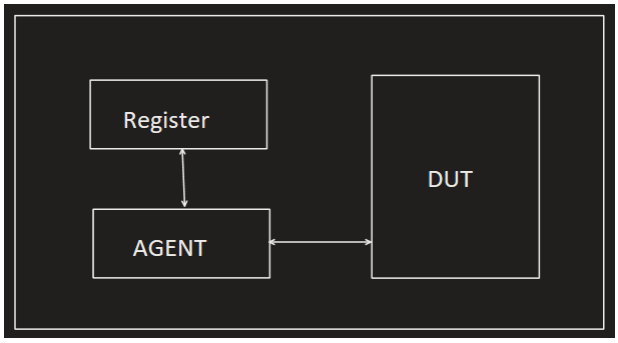

Fig. 3. Register Abstraction Layer

The advantages of UVM RAL Model are

- Provides high-level access to registers in the DUT by name, address or reference.
- UVM provides a sequence library to perform initial tests on the registers to verify that the correct registers are being read and written. A backdoor access is also provided with these classes. This can allow to corrupt the data in the DUT and see how it is handled.
- Read and write methods can be called on design registers, regardless of Bus interface state.
- Multiple threads are allowed to access the register model.
- A simple change in the blk file can allow a verification engineer to re use register model in next iteration of a project. Using RAL has become an industry standard, it has allowed for re-use of this model for almost all the commonly found IP cores like AHB-Bridge, AXI slave etc.
- Generation of a register model is simple because of the standardized approach, Open source scripts are readily available to help in aiding generation of a RAL model.

D. Reference Model

Reference Model as the name suggests is a standard to be set against which the DUT reesponse will be matched. The UVM testbench response is compared with the output of the reference model.

[6] The problem is that when 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.

Fig. 4. thread cancelled successfully for Use case

A reference model is supposed to be the golden standard for the application and hence it's accuracy is of utmost importance. It is becomes very difficult to code a reference model with a HDL for systems of higher level of complexity. A high level language's help may be used for such applications. A language like MATLAB or C/C++ can be used for coding a reference model. Different languages have different benefits like speed, ease of use etc. Applications like signal processing can benefit from using MATLAB because of it's tools.

Fig. 5. Cancellation Signal Received

E. Implementation

[7]Using RAL, It becomes possible to access registers directly by name and address. This allows manipulation of thread related registers with ease. Stopping or starting a thread just becomes a matter of assigning a few registers. Once that option is available, It is then used to verify whether a potential improvement is already present. If so, then a new possibility of improvement can be chosen. When Improvements are once performed, It can also be again verified by changing these registers appropriately. Once the design has been changed, New scenarios must be coded onto the C model or reference design. This involves deep understanding in the design and mathematical aspects of the system.

However careful consideration must given to performance enhancement to implement, because if the enhancement requires alot of RTL changes, drastic changes must also be made to the reference model. Then the UVM monitor and scoreboard must also be modified accordingly. Hence a performance enhancement must be considered only if it is giving drastic improvement for small RTL changes. Which in turn lead to less changes to reference model.

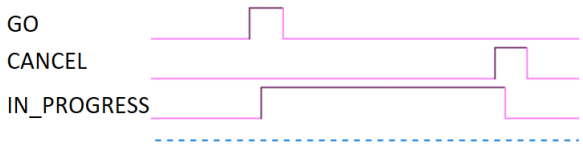

Fig. 6. thread Cancelled successfully

IV. RESULTS AND DISCUSSION

This section contains all simulation waveforms and their detailed explanations respectively. Cancellation of thread mid duration was successfully achieved. Which will be followed by a tabular representation of effects of said changes implemented. In terms of clock cycles saved.

A. Simulation results

1) Cancellation Signal through UVM Virtual Sequencer: The Fig. 5 shows go1, go2, go3 signals being triggered which then sets of thread1, thread2 and thread3 to sequentially execute.

Once thread1 starts running, It can be seen thread2 and thread3 are in pending state. A cancellation pulse is sent midway during thread execution. Which in this case doesn't affect thread1's progress, because RTL has not been changed and this functionality is currently not present.

TABLE I
THREAD CANCELLATION IMPROVEMENT

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

2) thread Midway Cancel Successful: In the Fig. 6, it can be observed that 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.

3) thread Cancellation Successful for Use-case: The RTL change done to implement the feature should've propagated and broken the whole system, however it didn't happen and now feature must be tested for a use-case, Hence it was then simulated on an existing use-case with appropriate modifications. This Fig. 4 shows a use case being cancelled midway during it's processing.

B. Performance Impact

The Table I 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 priority task. On an average it assumed thread is stopped right in middle, in this case it will save 50% of the clock cycles that the thread would've taken to finish itself.

V. CONCLUSION

It can be inferred that this improvement is very dependent on how frequently it is utilized, however the amount RTL that was needed to change was minimal, and hence the amount of changes in UVM testbench and reference design remained less. This improvement has the potential to save 1000's of clock cycles, which matters quite alot for a low power and high speed application. Even marginal gains are quite useful if the amount of RTL to be changed is less.

REFERENCES

[1] D.-S. Chen, P.-Y. Chen, and Y.-W. Wang, "Hardware/software co-design of nlms adaptive filters on fpga," in *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, Jun. 2011, pp. 442–445. DOI: 10.1109/ISCE.2011.5973866.

[2] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Springer Publications, 2008, ISBN: ISBN 978-0-387-68606-6.

[3] Y. Mollaei, "Hardware implementation of adaptive filters," in *2009 IEEE Student Conference on Research and Development (SCOReD)*, Nov. 2009, pp. 45–48. DOI: 10.1109/SCORED.2009.5443365.

[4] F. Nekouei, N. Z. Talebi, Y. S. Kavian, and A. Mahani, "Fpga implementation of lms self correcting adaptive filter (scaf) and hardware analysis," in *2012 8th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*, Jul. 2012, pp. 1–5. DOI: 10.1109/CSNDSP.2012.6292753.

[5] A. Moursi, R. Samhoud, Y. Kamal, S. El-Ashry, and A. Shalaby, "Different reference models for uvm environment to speed up the verification time," Dec. 2018, pp. 67–72. DOI: 10.1109/MTV.2018.00023.

[6] W. Ni and J. Zhang, "Research of reusability based on uvm verification," in *2015 IEEE 11th International Conference on ASIC (ASICON)*, Nov. 2015, pp. 1–4. DOI: 10.1109/ASICON.2015.7517189.

[7] A. Jain and R. Gupta, "Scaling the uvm$_r egmodeltowardsautomationandsimplicityofuse$," in *2015 28th International Conference on VLSI Design*, Jan. 2015, pp. 164–169. DOI: 10.1109/VLSID.2015.33.