RESEARCH ARTICLE OPEN ACCESS

Functional and Assertion Based Verification of Audio Echo Effect Unit

Sreevani Nanjuri N[1] , Nagesh .K.N[2], Yaseen Basha[3]
[1]Department of ECE, PG Student, NCET, Bengaluru, India,
[2] Department of ECE, Professor & Hod, NCET, Bengaluru, India,
[3]Department of ECE, Assistant professor, NCET, Bengaluru, India,
Corresponding Author: Sreevani Nanjuri N

ABSTRACT
The aspect ratio of MOS (Metal oxide semiconductor) Transistors are scaling down, designer are able to put more circuit with various functionality on a single die. This made design and verification process complex. If we consider today's system on chip (SOC) design, it is impossible to check all possible combination of input on design. To verify complex design successfully various verification techniques are exists. Successful verification is very much required for design signoff. There are various verification techniques like functional verification, equivalence checking, model checking, code and functional coverage, Assertion based verification are employed in verification process. In this paper, the sub modules such as Counter, Subtractor, Multiplexer, Memory unit and a Multiplier is designed and verified. Using these sub modules the top module for audio echo effect unit is designed and verified with test benches (functional). The Assertion based verification is performed on the top module.

Keywords-: Audio echo effect unit, Functional Verification, Assertion Based Verification, Verification Approaches.

Date of Submission: 11-07-2018 Date of acceptance: 25-07-2018

I. INTRODUCTION

Verification[1] is a procedure used to exhibit that the goal of configuration saved in it's execution. Today, the period of multi-million-gate Application Specific Integrated Circuits (ASIC's), reusable intellectual property (IP) and system on-chip (SoC)[2] plan check expends 70% of outline endeavors. Because of this number of verification architects can be double the quantity of Register Transfer level (RTL) Designers. Verification moment can be lessened through parallelism. Verification time can be decreases through automation.

II. DIFFERENT TYPES OF FUNCTIONAL VERIFICATION APPROACHES

There are three complementary functional verification approaches.
BLOCK-BOX verification
WHITE-BOX verification
GRAY-BOX verification

2.1 BLOCK-BOX Verification

In this confirmation, without any understanding of the real realization of the design the functional verification[1] can be performed. The benefit of block-box verification is that it is independent on any exact implementation whether the implemented in a single ASIC, RTL code. It is hard to observe and control precise features in block-box verification. Critical functions, deep into the design will be complicated to manage and monitor.

2.2 WHITE-BOX Approach

This approach has intimate information of the internals of a plan and also has control over it. The advantage of this approach is being able to add any interesting arrangement of states and inputs quickly, or to separate a desired function based on requirement.

2.3 GRAY-BOX-Verification

It is understand between White-box verification and block box verification. This means, block- box may not fully use all parts while the white box is not convenient. A gray-box approach commands and notices a plan completely through its top level interfaces (block-box).

III. FORMAL VERIFICATION

It is a method of verifying whether the design fulfills the specific requirement or not (properties).Formal verification[3] does not remove the requirement to write test-benches. Once you follow what the conclusion points of the formal

verification reconvergent paths are, you be familiar with what perfectly is being established. The main application of formal verification falls under two categories, they are
1) Equivalence checking
2) Model checking

3.1 Equivalence checking

Equivalence checking differentiates two models. The most common advantage of equivalence checking is it balance two net lists to make sure that some net list post-processing, for example clock-tree synthesis or physical alteration, chain insertion, did not modify the process of the path. In the synthesis software it can find bugs, another general use of equivalence can find bugs, another general use of equivalence checking is to find that the net list properly perform the original RTL code.

3.2 Model checking

The most recent application of the formal verification technology is model checking. It confirms assertions about the performance of the design. A most influential model checker may be capable to detect if deadlock condition can arise. In it design assertions or characteristics are formally verified or disproved.

IV. ASSERTION BASED VERIFICATION

Assertions[4] institutionalization accomplishments hold the guarantee of enhancing verification proficiency and enabling formal check to work with simulation.

4.1 System Verilog Assertions

Statements are basically used to approve the conduct of a plan and they may also be utilized to give useful scope in development to an outline. Affirmations can be checked powerfully by recreation, or statically by a different property checker apparatus, formal confirmation instrument that demonstrates regardless of whether a plan meets its specification. There are two kinds of statements characterized in the system verilog language.[5]

4.2 Concurrent assertions

Based on clock cycles and test articulation is assessed at clock edges in light of the inspected estimations of the factors included. Inspecting of factors is done in the preponed area and the assessment of the articulation is done in the watched locale of the scheduler. These can be put in a procedural block, a unit, an interface or a program explanation. Concurrent declarations be able to utilized with together static and dynamic confirmation devices.

4.3 Immediate assertions

Immediate assertions are procedural proclamations and are mostly utilized as a part of simulation. An assertion is essentially a statement that something must be valid, like the If statement. Test articulation is assessed simply like some other extremely log articulation inside a procedural block. These are not worldly in nature and are assessed instantly and must be put in a procedural square definition. Quick declarations utilized just with dynamic simulation[6].

4.4 Assertions

It is a description of a property of the plan, If the property that is being checked for a reproduction does not carry on the way we guess, the assertion comes up short. The property that is not allowed from occurring in an outline occurs amid recreation, the statements falls flat. A list of properties can be taken from the useful detail of a plan and can be changed over in to assertions. The assertions can be constantly checked amid functional simulation. It is likewise called as screens or checkers. The Assertions formally written in System verilog, so it is typically called System verilog Assertion (SVA). It doesn't written in verilog in light of the fact that verilog has few detriments, they are Verilog is a procedural dialect and henceforth, does not have great control after some time. It is a verbose dialect. it implies as the assertions builds, it is extremely hard to keep up the code. Verilog has no worked in system to give utilitarian scope information. Verilog checkers may not catch all the activated occasions. The real distinction between the model checking and Assertion based Verification[7] is all the more ground-breaking then the model checking.

V. AUDIO ECHO EFFECT DESIGN

An audio echo effects unit that works by delay the samples of an acoustic signal indicated as a flow of 16-bit 2s-complement binary- coded standards. The sample rate is 50kHz.Appearance of a original input trial is represented by a control input, audio_in_en, being 1 for the clock cycle in which the model arrives. The component should point out accessibility of an productivity model using an yield control sign, audio_out_en, in the similar method.

The holdup time is found by an 8-bit unsigned input illustrating the number of milliseconds of holdup. We can delay the incoming acoustic model values by saving them in a memory until they are essential at the yield. The highest delay articulated by 8-bit unsigned input is 255ms. because samples appear at a speed of 50 kHz (that is, 50 per millisecond), we require to stock up up to 12,750 samples. A 16K X 16-bit memory, with 14-bit addresses, will be sufficient. A figure of the data path additionally the memory and additional

mechanism to calculate addresses revealed in the figure1.
We require to utilize a 14-bit counter to maintain track of wherever sample arrives, we keep it at the subsequently accessible memory location, whose address is specified by the counter. We after that read from the memory the value written d milliseconds in the precedent (where d is the value of the delay input) and give it at the yield, then increase the counter to refer to the next position in memory. The value written d milliseconds formerly is stored 50 X d locations earlier to the existing location specified by the address counter. Therefore we can calculate its address by multiplying d by 50 and subtracting the end product from the value of the address counter. The counter will rise to utmost address value then wrap around to 0, efficiently augmenting modulo 16K. Thus, formerly the memory is overflowing, older locations will be over written with recently inward samples. When we complete the subtractor will yield the distinction

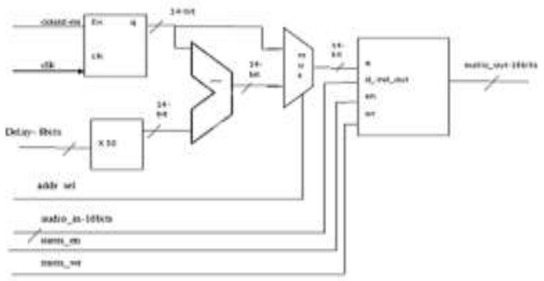


Fig 1: Functional diagram of Audio Echo Effect Unit modulo 16K, and require to be provide the correct address of the necessary delayed sample.

VI. RESULTS & DISCUSSIONS

6.1 14-Bit Counter

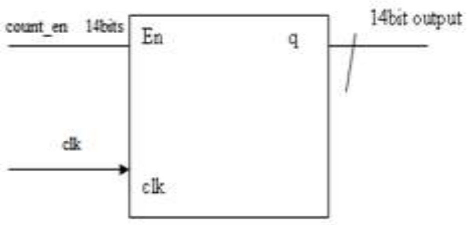


Fig 2: Block diagram of counter

Fig 3: Simulation wave form of 14-bit counter

When clock & count_enable is high, the output of q increases q= 14'b 0000 0000 0000 00 to q=14'b 0000 0000 00 1111.
Once complete the 14 bits it falls to zero and again starts increases upto 14 bits.

Table 1. Power analysis of 14 bit counter

Power analysis	**45nm**	**180nm**
Leakage power(nW)	10.165	40.213
Dynamic power(nW)	8344.540	46857.704
Total power (nW)	8354.704	46897.918

6.2 Multiplier

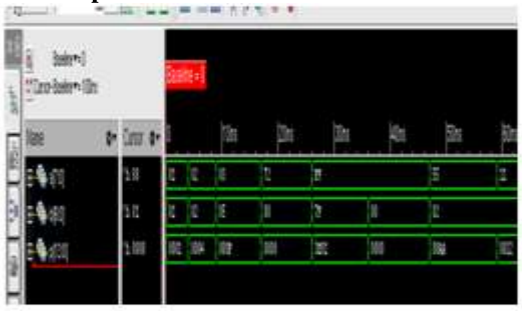


Fig 4: Simulation wave form of multiplier

At #0 ns the input a=00000001 and input b=0000001 the product of the a & b is p=00000000000001;
At #4 ns the input a=00000010 and input b=0000010 the product of the a & b is p=00000000000100;
At #9 ns the input a=00000011 and input b=0000101 the product of the a & b is p=0000000000 1111;

6.3 14-Bit Subtractor

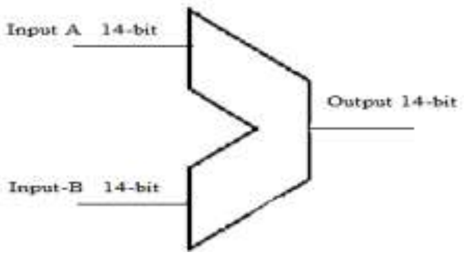


Fig 5: Block diagram of 14-bit Subtractor

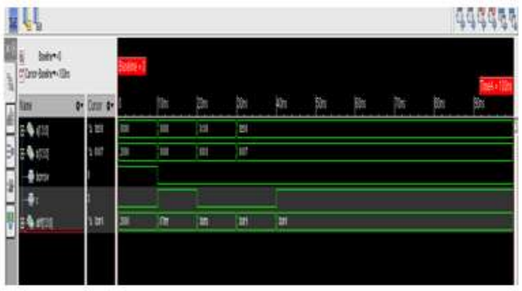


Fig 6: Simulation wave form of 14-bit Subtractor

When a=0000 0000 000000, b=10000000000000, borrow=1, difference=10000000000000;
When a=11100000000000, b=00000000000000, borrow=0, difference=00011111111111;
When a=11111000000000, b= 00000000000001, borrow=0, difference=11110111111000;

Table 2. Power analysis of 14 bit Subtractor

Power analysis	**45nm**	**180nm**
Leakage power (nW)	8.044	38.630
Dynamic power (nW)	8259.791	46817.159
Total power (nW)	8267.835	46855.789

6.4 2-1 Multiplexer

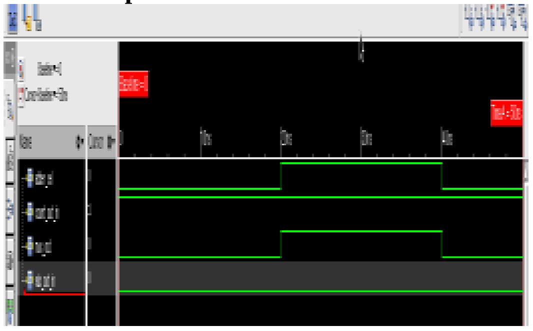


Fig 7: Simulation waveform of 2X1 multiplexer

When mux_out=0,count_in=1,sub_out_in=0,adder_sel=0
When mux_out=1,count_in=1,sub_out_in=0,adder_sel=1
When mux_out=0,count_in=1,sub_out_in=0,adder_sel=0

Table 3. Power analysis of 2-1 Multiplexer

Power analysis	**45nm**	**180nm**
Leakage power (nW)	0.495	1.34
Dynamic power (nW)	303.199	667.708
Total power (nW)	303.694	669.022

6.5 16-Bit memory

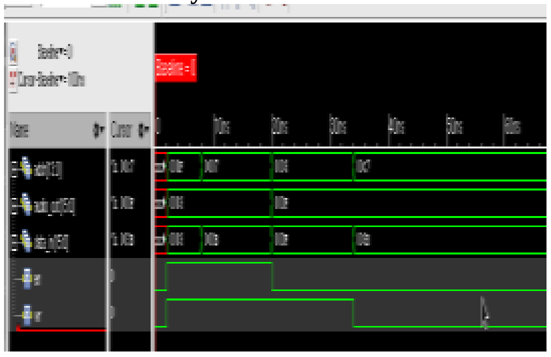


Fig 8: Simulation waveform of 16-bit memory

At#2ns
data_in=0000000000000011,addr=00000000001111
,
write=high, enable=high, audio_out=0000000000000011.
At#8ns
data_in=0000000000001011,addr=00000000000111
,
write=high,en=high,audio_out=0000000000000011

Table 4. Power analysis of 16-bit memory

Power analysis	**45nm**	**180nm**
Leakage power(nW)	630.635	1147.807
Dynamic power(nW)	22572.603	243192.060
Total power(nW)	23203.238	244339.867

6.6 Audio echo effect Unit

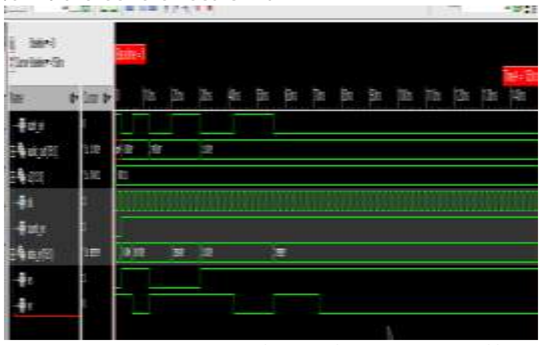


Fig 9: Simulation waveform of Audioecho effect unit

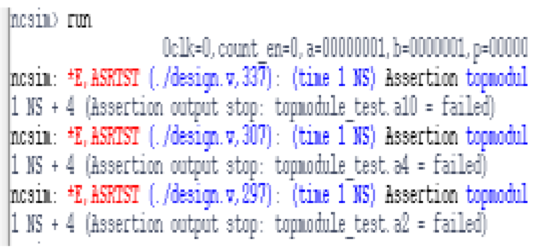
From the waveform at at 2ns, count_en is high, c2=0000 0000 0000 0001 and addr_sel is zero the inputs data_in =0000 0000 0000 1111; write is high and enable is high , the output of the audio_out is data_in.

From the waveform at at 10ns, count_en is high, c2=0000 0000 0000 0001 and addr_sel is one the inputs data_in =1111 0000 0000 1111; write is low and enable is high , the output of the audio_out is previous output.

VII. ASSERTION BASED VERIFICATION PROPERTIES & RESULTS

```
Property p1;
@ (posedge clk)  ! (addr_sel);
Endproperty
  a1:assert property(p1);
Property p2;
@ (posedge clk)  (en&!wr);
Endproperty
  a2:assert property(p2);
Property p3;
@ (posedge clk) (en&wr);
Endproperty
```

a3:assert property(p3);



```
ncsim> run
                  0clk=0,count_en=0,a=00000001,b=0000001,p=00000
ncsim: *E,ASRTST (./design.v,337): (time 1 NS) Assertion topmodul
1 NS + 4 (Assertion output stop: topmodule_test.a10 = failed)
ncsim: *E,ASRTST (./design.v,307): (time 1 NS) Assertion topmodul
1 NS + 4 (Assertion output stop: topmodule_test.a4 = failed)
ncsim: *E,ASRTST (./design.v,297): (time 1 NS) Assertion topmodul
1 NS + 4 (Assertion output stop: topmodule_test.a2 = failed)
```

Fig 10: Assertion based verification simulation

clk=0,count_en=0,a=00000001,b=0000001,p=00000000000001,addr_sel=1,data_in=0000000000000011, wr=1,en=0,audio_out=xxxxxxxxxxxxxxxx.
ncsim: *E,ASRTST (./design.v,337):(time 1 NS) Assertion topmodule_test.a1 has failed
ncsim: *E,ASRTST (./design.v,307): (time 1 NS) Assertion topmodule_test.a3 has failed
ncsim: *E,ASRTST (./design.v,297): (time 1 NS) Assertion topmodule_test.a2 has failed

At 1ns the property p1, property p3 and property p2 are failed because its not satisfy the property rules.
When en&wr both are high ,the property p3 satisfy or else it fails.
When en is high and wr is low, the property p2 satisfy or else it fails.
These three conditions are failed at 1ns.
clk=0,count_en=1,a=00000011,b=0000101,p=00000000001111,addr_sel=1,data_in=1111000000001111, wr=0,en=1,audio_out=0000000000001111
ncsim: *E,ASRTST (./design.v,337): (time 7 NS) Assertion topmodule_test.a1 has failed.
at #7ns the above property 1 failed. because addr_sel is 1 at 7ns.

VIII. CONCLUSION

Audio echo effect unit is designed and verified successfully. Initially basic functionality of the audio echo effect unit is verified using test bench. After verifying basic functionality, the different properties of assertions are verified. Functional and assertions results are presented in this paper. Functional verification is applied on different sub modules of Audio echo effect design such as, 14-bit Subtractor, 14-bit counter and multiplier, multiplexer as well as memory unit. Assertion based verification is applied for main module audio echo effect unit. System very log Assertions are used to apply this verification technique. Simulation results are shown for both functional and assertion based techniques.

FUTURE SCOPE OF WORK

Today's SoC/ASIC designs are more complex, it is impossible to verify the functionality of the designs using test benches. So assertion based technique plays a very important role in finding the bugs in the design. Before manufacturing any design such as SoC/ASICs it is very essential to verify the design using formal verification technique.

REFERENCES

[1]. A.Fedeli,F.Fummi,and g. Pravadelli. "properties Incompleteness Evalution by Functional Verification," IEEE Trans. Computers, vol.56, no.4 Apr 2007, pp.528-544
[2]. Prakash Rashinkar and peter Paterson, system-on-chip-verifictaion Kluwer academic publishers,2002,ISBN 0-306-46995-2.
[3]. D.W.Currie,A.J.Hu,S.Rajan,M.Fujita,"Automatic Formal Verification of DSP Software", Proc.Design Automation Conference,pp.130—135,jun.2000
[4]. Nicola Bombieri, Franco Fummi, and graziano Pravadelli, "Hybrid, Incremental Assertion-Based Verification for TLM Design Flows" IEEE CS press, 2007.
[5]. A.Dahan et al., "Combining System Level Modeling with Assertion Based Verification ," Proc. 6th int'l Symp. Quality Electronic Design (ISQED 05),IEEE CS Press, 2005,pp.310-315.
[6]. Chandy,K.M.,Misra, J.1979. Distributed simulation: A case study in design and verification of distributed programs. IEEE Trans. On Software. Eng. SE-5, 5(sep 1979).
[7]. Srikanth Vijayaraghavan and Meyyappan, a practical guide for system verilog assertions, Springer, 2005

Sreevani Nanjuri N "Functional and Assertion Based Verification of Audio Echo Effect Unit "International Journal of Engineering Research and Applications (IJERA) , vol. 8, no.7, 2018, pp.04-08