Increasing Functional Coverage by Automation limits manufacturing reruns on expensive **FDSOI technology** for **Zetta-Hz High Speed CDMA Transceiver**

Karthik Bandaru[a], Priyanka Gharat[b] & Past Dean & Prof Sastry Puranapanda[c]
Silicon Interfaces{R}
info@siliconinterfaces.com
http:www.siliconinterfaces.com

Abstract: - Fully Depleted Silicon on Insulator or FDSOI, is an European technology using planar process technology for reduced silicon geometries. Since the deployment is still not so much at production level, it is important that the ASIC works the first time around, since it is very expensive.
The efforts to randomize and constraints the test cases is based on the developer or verification engineers perception of test vectors required and can easily lead to the hidden bug being overlooked. Traditionally, the coverage goals would have been reached by writing more test cases with unpredictable time lines (particularly keeping time to market guidelines in view). Functional Coverage defines critical states and Constrained Randomization test those states in unpredictable ways. Often Constrained Randomization necessarily repeats states or worse catastrophically misses the coverage point which has the "hidden" bug since the coverage points are written by the verification engineer by using coverage bins. Other shortcomings of Functional Coverage are that we define critical states, which can be targeted first with fewer test but provided we know which the critical paths are. To meet coverage goals, the verification engineer would need to write more test cases.

This paper explores the shortcomings of standard Constrained Randomization techniques to attain coverage and uses coverage automation tools, using intelligent routines and algorithms and analyzes the coverage matrix and does the maximum possible coverage by traversing the paths to detect undetectable bugs to increase functional coverage and applies it to a Zetta-Hz High Speed CDMA transceiver.

Key-Words: - Constrained Randomization, Functional coverage, CDMA Code Division Multiple Access, Coverage Automation, UVM (Universal Verification Methodology) SystemVerilog, Verilog.

[I] Purpose

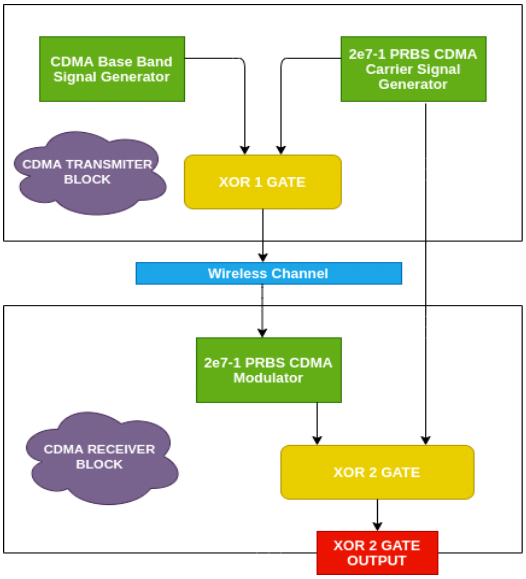
To show that using coverage automation tools, verification engineers, can identify hidden bugs and reduce time for verification and meet time to market guidelines.

This paper will methodically look at coverage reports generated by standard simulation, analyze it in terms of meeting coverage goal matrix, and then run automation tools to again higher coverage goals in lesser simulation cycles and time.

For the purpose of our proof of concept we are using a **Zetta-Hz High Speed CDMA Transceiver** and running standard simulation with coverage and than comparing with results attained using automation.

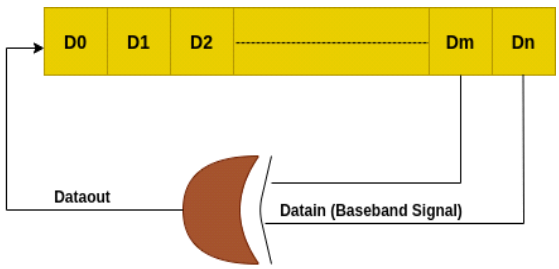
[II] Description of the DUT

This Zetta-Hz High Speed CDMA Transceiver is designed for multiple access for multiple users at a time by spreading and de-spreading of codes with very high frequency bandwidth spectrum based ultra high speed networking communication using different CDMA techniques DSSS(Direct sequence spread spectrum), frequency hopping, Chaotic etc. The design has data transmission and reception done in parallel for various data interface cards of different high speed data transfer. In this Design transmission and reception done by different PRBS data pattern sequences like 2e7-1, 2e10-1, 2e15-1, 2e23-1, 2e31-1, 2e48-1, 2e52-1, 2e63-1 etc as per ITU standard.



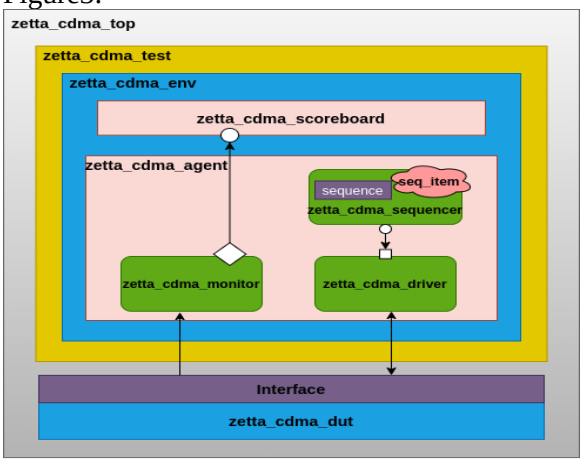
Figure(1) Zetta-Hz High Speed CDMA Transceiver

The zetta hertz transceiver consists of zetta clock frequency oscillator/generator, channel encoder, modulator, wireless channel, demodulator, decoder, PRBS-pseudo noise carrier frequency wave generator. The zetta hertz oscillator generates the one complete zetta hertz clock cycle with 2^70/2 low and 2^70/2 for high clock pulses. Total clock frequency cycle is 2^70 clock pulses. This clock frequency is input to all above CDMA blocks. This generate high frequency baud rate in terms of zetta hertz clock frequency. CDMA modulation is done by multiplying multiplexed comparing with channel encoder based CDMA baseband signal codes and carrier wave codes of PRBS pseudo noise generated by pseudo noise carrier noise generator and demodulation detection done by de-multiplexing with original CDMA transmitter output PRBS. All these blocks synchronized with zetta hertz clock frequency rate.



Figure(2) N-bit Pseudo random binary sequence generator

To verify the Zetta Hertz CDMA transceiver, we use a UVM based verification environment. The hierarchical structure of UVM test bench as shown in Figure3:



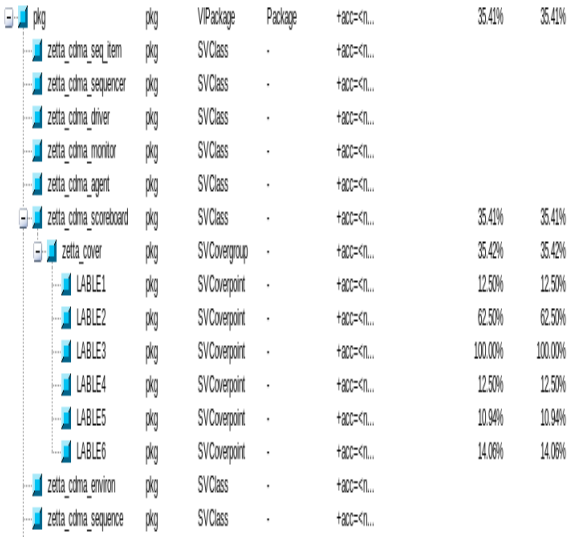
Figure(3) Hierarchical structure of UVM

[III] Constrained Random Verification and Functional Coverage

Using the standard simulation technique only the critical states can be hit. Constrained Randomization hits these test in unpredictable ways. Often constrained randomization repeat states and miss the coverage points. Bugs may be in the hidden/uncovered nodes and each may not be detected by the constrained random verification.

Figure (4) Constrained Randomization and Functional coverage

Now, the coverage matrix is shown in Fig 4 with standard simulation which shows percentage as 35.41%.



pkg	pkg	VlPackage	Package	+acc=<n...	35.41%	35.41%
zetta_cdma_seq_item	pkg	SVClass	-	+acc=<n...		
zetta_cdma_sequencer	pkg	SVClass	-	+acc=<n...		
zetta_cdma_driver	pkg	SVClass	-	+acc=<n...		
zetta_cdma_monitor	pkg	SVClass	-	+acc=<n...		
zetta_cdma_agent	pkg	SVClass	-	+acc=<n...		
zetta_cdma_scoreboard	pkg	SVClass	-	+acc=<n...	35.41%	35.41%
zetta_cover	pkg	SVCovergroup	-	+acc=<n...	35.42%	35.42%
LABLE1	pkg	SVCoverpoint	-	+acc=<n...	12.50%	12.50%
LABLE2	pkg	SVCoverpoint	-	+acc=<n...	62.50%	62.50%
LABLE3	pkg	SVCoverpoint	-	+acc=<n...	100.00%	100.00%
LABLE4	pkg	SVCoverpoint	-	+acc=<n...	12.50%	12.50%
LABLE5	pkg	SVCoverpoint	-	+acc=<n...	10.94%	10.94%
LABLE6	pkg	SVCoverpoint	-	+acc=<n...	14.06%	14.06%
zetta_cdma_environ	pkg	SVClass	-	+acc=<n...		
zetta_cdma_sequence	pkg	SVClass	-	+acc=<n...		

Figure(5) Functional Coverage with standard simulation

Coverage matrix using Automation tool :

Using this method we will see how the coverage percentage is increased. First, the work library of UVM is imported by the tool, and then the sequence item and scoreboard where the coverage is written in the classes added into the tool for cover groups and stimulus items.

```
class zetta_cdma_seq_item extends
uvm_sequence_item;
`uvm_object_utils(zetta_cdma_seq_item)
reg reset;
rand reg [2:0] prbstype;
rand reg [3:0]prbs_cdma_channel_sel;
rand reg [9:0] serin;
reg cdmatxserout;
reg [9:0]cdmarxserout;
reg [63:0] cdmatxparout;
reg [63:0] cdmarxparout;
function new (string name =
"zetta_cdma_seq_item");
super.new(name);
endfunction
function void do_print(uvm_printer printer);
super.do_print(printer);
-
-
-
endfunction
-
-
-
endclass
```

Figure (6) Zetta_sequence_item code

```
class zetta_cdma_scoreboard extendsuvm_scoreboard;
`uvm_component_utils(zetta_cdma_scoreboard)

uvm_tlm_analysis_fifo#(zetta_cdma_seq_item) zetta_imp;
zetta_cdma_seq_item xtn;
covergroup zetta_cover;
option.per_instance=1;

LABLE1: coverpoint xtn.serin;
LABLE2: coverpoint xtn.prbstype;
LABLE3: coverpoint xtn.cdmatxserout;
LABLE4: coverpoint xtn.cdmarxserout;
LABLE5: coverpoint xtn.cdmatxparout;
LABLE6: coverpoint xtn.cdmarxparout;
endgroup

function new(string name,uvm_component parent);
super.new(name,parent);
zetta_imp=new("zetta_imp",this);
zetta_cover =new();
endfunction

function void build_phase(uvm_phase phase);
super.build_phase(phase);
endfunction

task run_phase(uvm_phase phase);
forever begin
zetta_imp.get(xtn);
zetta_cover.sample();
end
endtask
-
-
-
endclass
```

Figure(7) zetta_scoreboard code

After several steps, the tool sequence generates in the format of UVM sequence. Source code of the generated tool sequence is shown in Fig 8.

```
`define  INCLUDED_zetta_cdma_sequence_gen_SVH
import uvm_pkg::*;
`include "uvm_macros.svh"
`include "zetta_cdma_seq_item_gen_scheduler.svh"
`include "zetta_cdma_seq_item_gen_items.svh"
`include "zetta_cdma_seq_item_gen_zetta_cover.svh"
`include "zetta_cdma_seq_item_gen_callback_closures.svh"
typedef zetta_cdma_sequence zetta_cdma_seq_item_gen_base_t;
typedef class zetta_cdma_seq_item_gen_cb;
class zetta_cdma_sequence_gen extends zetta_cdma_seq_item_gen_base_t;
typedef zetta_cdma_seq_item_gen zetta_cdma_seq_item_gen_t;
`uvm_object_utils(zetta_cdma_sequence_gen)
zetta_cdma_seq_item_gen_scheduler                m_scheduler;
function new(string name="");
super.new(name);
endfunction
virtual task body();
m_scheduler = new(get_full_name());
register_actions();
create_cov_strategies();
m_scheduler.run_all();
begin
te = m_scheduler.zetta_cdma_seq_item_gen();
te.delete();
end
endtask

-

-

-

endclass
```

11Figure(8) Generated tool code

Base sequence class (zetta_cdma_sequence) will be overridden with the tool generated sequence using UVM overriding methods while simulating the code. Then we can observe that generated coverage reached 99.41% where using normal simulation technique it was 35.41%.

uvm_pkg	uvm_pkg	VIPackage	Package	+acc=<n..		
pkg	pkg	VIPackage	Package	+acc=<n..	99.47%	99.47%
zetta_cdma_seq_item	pkg	SVClass	-	+acc=<n..		
zetta_cdma_sequencer	pkg	SVClass	-	+acc=<n..		
zetta_cdma_driver	pkg	SVClass	-	+acc=<n..		
zetta_cdma_monitor	pkg	SVClass	-	+acc=<n..		
zetta_cdma_agent	pkg	SVClass	-	+acc=<n..		
zetta_cdma_scoreboard	pkg	SVClass	-	+acc=<n..	99.47%	99.47%
zetta_cover	pkg	SVCovergroup	-	+acc=<n..	99.48%	99.48%
LABLE1	pkg	SVCoverpoint	-	+acc=<n..	100.00%	100.00%
LABLE2	pkg	SVCoverpoint	-	+acc=<n..	100.00%	100.00%
LABLE3	pkg	SVCoverpoint	-	+acc=<n..	100.00%	100.00%
LABLE4	pkg	SVCoverpoint	-	+acc=<n..	98.44%	98.44%
LABLE5	pkg	SVCoverpoint	-	+acc=<n..	98.44%	98.44%
LABLE6	pkg	SVCoverpoint	-	+acc=<n..	100.00%	100.00%
zetta_cdma_environ	pkg	SVClass	-	+acc=<n..		

Figure(9) Functional coverage using Automation tool

Figure(10) Constrained Randomization and Automated Functional Coverage

Using the coverage automation maximum coverage can be reached in the fewest tests. Since it covers all the nodes the bugs hidden in the corner areas will be detected.

[IV] Conclusion

Automated Functional Coverage is an important aspect of ASIC/SOC verification for checking the functionality of a particular design. The design Zetta-Hz high speed CDMA transceiver can achieve 100% functional coverage1 indicates that all areas of designs may be tested. Here we use automation and achieve coverage goals 10X to 100X faster and scales of existing coverage of Designs.