1 Introduction

The project 3 DTMF receiver module, TDSP versions, DSP behavior algorithm are implemented using Verilog and TDSP assembly language. Finite State Machine is implemented using Verilog behaviorally. The output dma_grant is 1 in GRANTD_DMA and tdsp_grant is 1 in GRANT_TDSP and CLEAR with GRANT_TDSP being priority except in DMA_PRA state. The finite state machine test bench is created to test the TDSP_GRANT and DMA_GRANT being 1 within 1 clock period of TDSP_BREQ or DMA_BREQ. The assembly language TDSP test is implemented for # spectrum with quiet tones at the start and end.

2 Memory Access Arbiter ARB modules

(A)A Brief Module Description

Memory Access Bus Arbiter ARB module allows DMA and TDSP access to Data Sample memory. The ARB state machine diagram is implemented with IDLE, GRANT_TDSP, DMA_PRI, CELAR and GRANT_DMA states. TDSP priority is granted if both devices of tdsp_breq and dma_breq happen at the same time except in DMA_PRI state. Tdsp_grant is 1 in CLEAR and GRANT_TDSP, dma_grant is 1 in GRANT_DMA and they are zeros in other states. The bus arbiter has 25 MHz frequency, 25 MHz positive edge triggered clock and active high reset which is asynchronous.

(B)Arb .v RTL source code

```
/*
 *
 * Author:  Zeyar Win
 *
 *
 */

module arb (
        reset,
        clk,
        dma_breq,
        dma_grant,
        tdsp_breq,
        tdsp_grant,
        scan_in0,
        scan_en,
        scan_out0
      );

/*
```

```
 *
 * DMA/ TDSP bus arbiter
 *
 */

input
   reset,                  // system reset
   clk,                    // system clock
   dma_breq,                 // dma controller bus request
   tdsp_breq ;             // tdsp bus request

output
   dma_grant,                // dma controller bus grant
   tdsp_grant ;            // tdsp bus grant

input
   scan_in0,                // test scan mode data input
   scan_en;                // test scan mode enable

output
   scan_out0;               // test scan mode data output

`include "./include/arb.h"

reg [2:0] state;
reg [2:0] nextstate;
reg    dma_grant;
reg    tdsp_grant;

//clock state change
always @(state or tdsp_breq or dma_breq)
          case(state)
          `ARB_IDLE: if (tdsp_breq == 1) begin

                    nextstate = `ARB_GRANT_TDSP;
                    end
          //        tdsp_grant = 1;
                    else if(dma_breq == 1 && tdsp_breq == 0)
                    begin
                    nextstate = `ARB_GRANT_DMA;

                    end else begin
                    nextstate = `ARB_IDLE;
                    end

          `ARB_GRANT_TDSP:
          if (tdsp_breq == 0)begin
```

```
                nextstate = `ARB_CLEAR;
                end else begin
                nextstate= `ARB_GRANT_TDSP;
        end
        `ARB_GRANT_DMA:
        if (dma_breq == 0)begin
                nextstate = `ARB_CLEAR;

        end else begin
                nextstate= `ARB_GRANT_DMA;
        end
        `ARB_CLEAR:
        if (dma_breq == 1)begin
                nextstate=`ARB_DMA_PRI;
        end else if (tdsp_breq == 1) begin
                nextstate=`ARB_GRANT_TDSP;
        end else begin
                nextstate=`ARB_CLEAR;
        end
        `ARB_DMA_PRI:
        if( dma_breq ==1) begin
                nextstate=`ARB_GRANT_DMA;
        end
        else if (tdsp_breq == 1 && dma_breq == 0) begin
                nextstate=`ARB_GRANT_TDSP;
        end
        else if (dma_breq != 1 && tdsp_breq != 1) begin
                nextstate=`ARB_IDLE;
        end
        else begin
                nextstate=`ARB_DMA_PRI;
        end

        default: nextstate= `ARB_IDLE;
        endcase


// seq
always @(posedge clk or posedge reset)
begin: FSM_SEQ
        if (reset == 1'b1)
        begin
                state <= `ARB_IDLE;
        end
        else
        begin
```

```
                state <=  nextstate;
        end

end
// output
always @(posedge clk or posedge reset)
begin
        if (reset == 1'b1)
        begin
                tdsp_grant = 1'b0;
                dma_grant  = 1'b0;

        end
        else
        begin
                case(nextstate)

                `ARB_IDLE:begin
                tdsp_grant = 1'b0;
                dma_grant  = 1'b0;

                end

                `ARB_GRANT_TDSP:begin
                tdsp_grant = 1'b1;
                dma_grant  = 1'b0;


                end

                `ARB_GRANT_DMA:begin
                tdsp_grant = 1'b0;
                dma_grant  = 1'b1;

                end

                `ARB_CLEAR:begin
                tdsp_grant = 1'b1;
                dma_grant  = 1'b0;

           end
                `ARB_DMA_PRI:begin
                tdsp_grant = 1'b0;
                dma_grant  = 1'b0;

           end
        endcase
```

```
        end
end

endmodule
```

(c)arb_test.v test bench source code

```
/*
 *
 * Author: zeyar win
 *         Rochester, NY, USA
 *
 */


`timescale 1ns / 1ns

module test;

wire dma_grant, tdsp_grant;
wire scan_out0;

reg clk, dma_breq, reset, tdsp_breq;
reg scan_in0, scan_en;

arb top(
       .reset(reset),
       .clk(clk),
       .dma_breq(dma_breq),
       .dma_grant(dma_grant),
       .tdsp_breq(tdsp_breq),
       .tdsp_grant(tdsp_grant),
       .scan_in0(scan_in0),
       .scan_en(scan_en),
       .scan_out0(scan_out0)
   );


reg [4: 0]
   dma_wait,
   tdsp_wait ;

integer
```

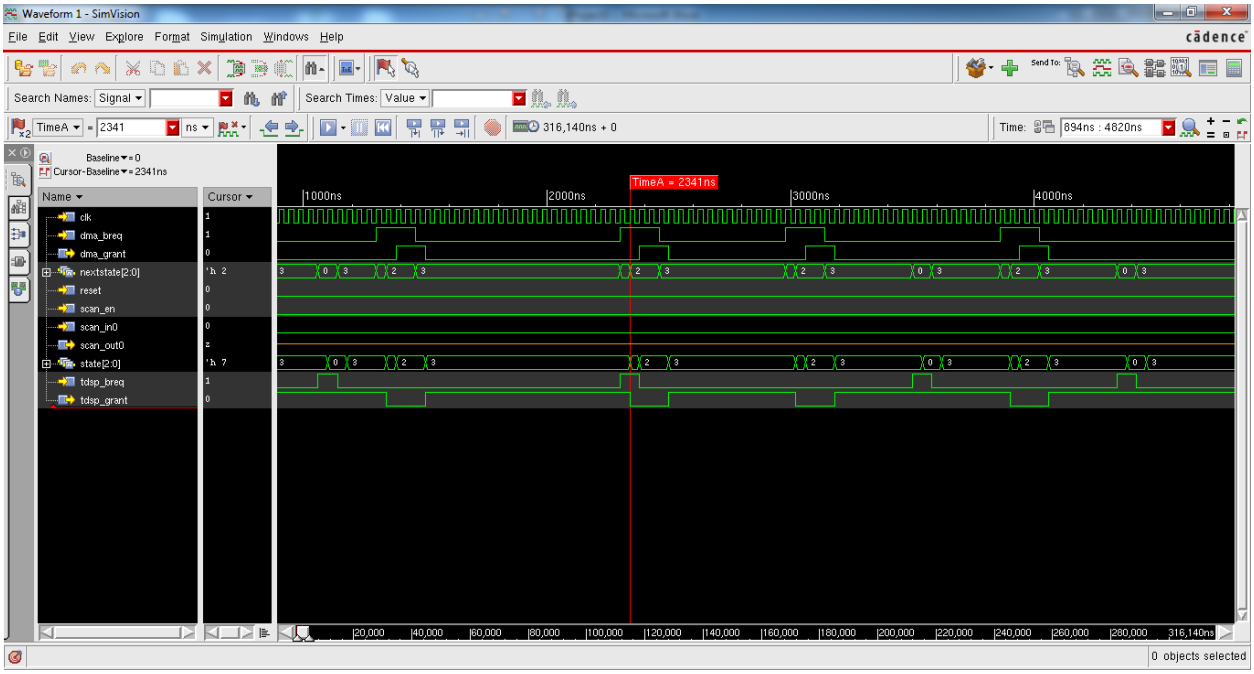
```
    i,
    j,
    k,
    l,
    dma_error1bit,
    tdsp_error1bit,
    dma_cnt2,
    tdsp_cnt2,
    dma_cnt,
    tdsp_cnt ;

wire
    grant = dma_grant | tdsp_grant ;

initial
begin
    $timeformat( -9, 2, "ns", 16);
`ifdef SDFSCAN

    $sdf_annotate("sdf/arb_tsmc18_scan.sdf", test.top);
`endif

    clk = 1'b0;
    dma_breq = 1'b0;
    reset = 1'b0;
    tdsp_breq = 1'b0;
    scan_in0 = 1'b0;
    scan_en = 1'b0;
    dma_cnt = 0 ;
    tdsp_cnt = 0 ;
    dma_cnt2 = 0;
    tdsp_cnt2=0;
    dma_wait = $random ;
    tdsp_wait = $random ;

    @(negedge clk)
     reset = 1'b1 ;
    repeat (2)
        @(negedge clk) ;
    @(negedge clk)
     reset = 1'b0 ;
    repeat (2)
        @(posedge clk) ;

    repeat (256)
    begin
        @(posedge clk)
```

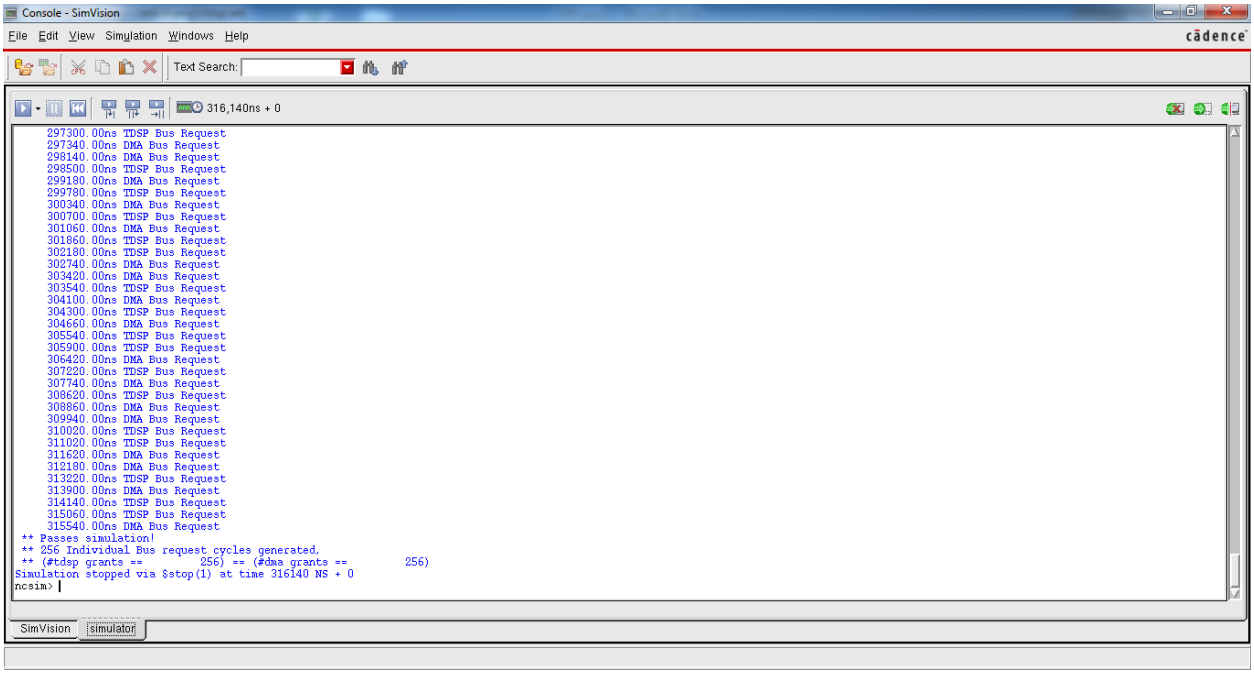
```
        dma_wait <= $random ;
       tdsp_wait <= $random ;
       fork
      // if output changes > one clock cycle #40ns of brequest high and anothor output changes to
zero,flag an error
      // use counter

            dma_request ;
          tdsp_request ;
            dma_check;
            tdsp_check;
       join
       repeat (4)
          @(posedge clk) ;
   end
   repeat (4)
      @(posedge clk) ;
   if (dma_cnt != tdsp_cnt)
   begin
      $display(" ** Fails simulation!");
      $display(" ** 256 Individual Bus request cycles generated,");
      $display(" ** (#tdsp grants == %d) != (#dma grants == %d)", tdsp_cnt, dma_cnt);
   end
   else
   begin
      $display(" ** Passes simulation!");
      $display(" ** 256 Individual Bus request cycles generated,");
      $display(" ** (#tdsp grants == %d) == (#dma grants == %d)", tdsp_cnt, dma_cnt);
   end
   $stop ;
end

always #20
   clk = ~clk ;

task dma_request ;
   begin
      repeat (dma_wait)
         @(posedge clk) ;
      dma_breq <= 1 ;
      $display("%t DMA Bus Request", $time);
      for (i = 0 ; i < (dma_wait + tdsp_wait + 10) ; i = i + 1)
         @(posedge clk)
          if (dma_grant)
          begin
             dma_cnt = dma_cnt + 1 ;
             i = (dma_wait + tdsp_wait + 10) ;
```

```
            end
        @(posedge clk)
         dma_breq <= 0 ;
       @(posedge clk);
    end
endtask

task tdsp_request ;
    begin
       repeat (tdsp_wait)
          @(posedge clk) ;
       tdsp_breq <= 1 ;
       $display("%t TDSP Bus Request", $time);
       for (j = 0 ; j < (dma_wait + tdsp_wait + 10) ; j = j + 1)
          @(posedge clk)
           if (tdsp_grant)
           begin
              tdsp_cnt = tdsp_cnt + 1 ;
              j = (dma_wait + tdsp_wait + 10) ;
           end
        @(posedge clk)
         tdsp_breq <= 0 ;
       @(posedge clk);
    end
endtask

task dma_check;
          begin
          @(negedge tdsp_grant)
          begin
          dma_cnt2 =0;
                    for (k=0 ; k<50 ; k=k+1 )
                    @(posedge clk)
                    begin
                              if(dma_grant != 1 )
                              begin
                                        dma_cnt2 = dma_cnt2 + 1;
                              //        $display("dma counter ", dma_cnt2);
                              end
                              else if(dma_grant == 1)
                              begin
                                        k  <=  50;
                              end
                              if ( dma_cnt2 > 2)
                              begin
                                        dma_error1bit <=  1;
                                        $display( "%t DMA_GRANT ERROR", $time);
```

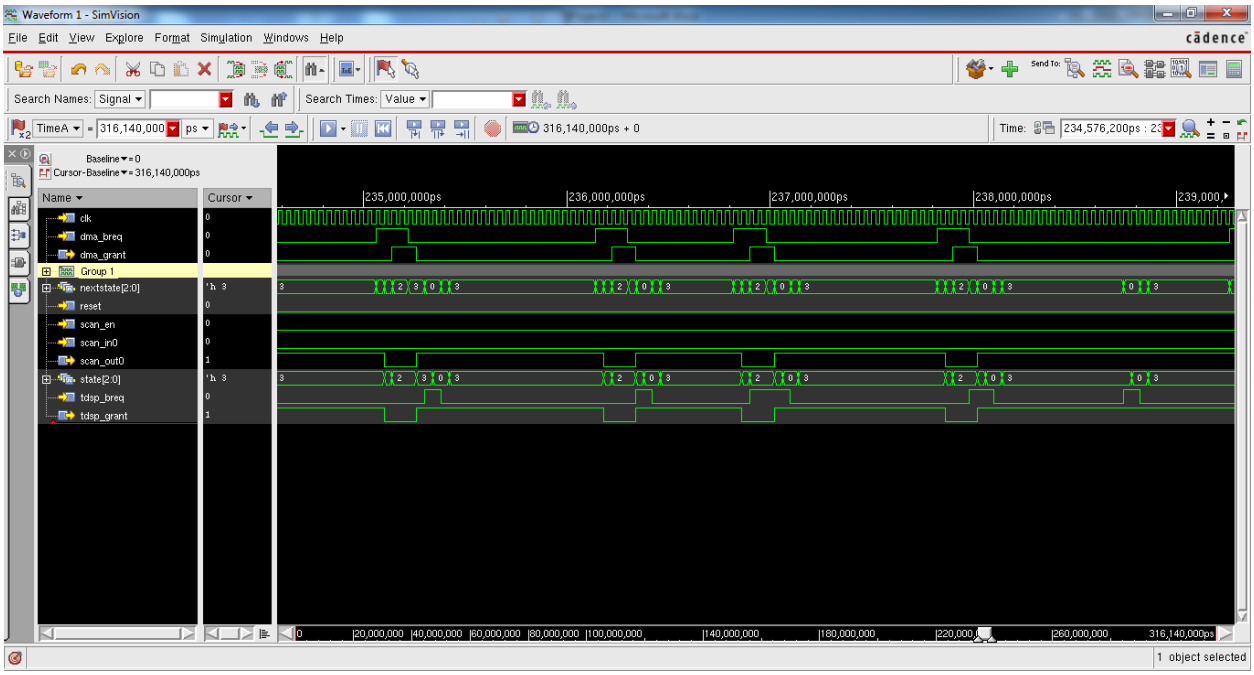
```
                        $stop;
                    end
                    @(posedge clk);
                end
        end
        end
endtask

task tdsp_check;
        begin
        @(negedge dma_grant)
        begin
                for (l=0 ; l<50 ; l=l+1 )
                @(posedge clk)
                begin
                    if(tdsp_grant != 1 )  //&& dma_breq ==1)
                    begin
                        tdsp_cnt2 = tdsp_cnt2 + 1;
                    //      $display ("tdsp counter", tdsp_cnt2);
                    end
                    else if(tdsp_grant == 1)
                    begin
                        l  <=  50;
                    end
                    if ( tdsp_cnt2 > 1)
                    begin
                        tdsp_error1bit <=  1;
                        $display( "%t TDSP_GRANT ERROR", $time);

                        $stop;
                    end
                    @(posedge clk);
                end
        end
        end
endtask

endmodule
```

(D) RTL level simulation waveform





(E) Net list Simulation



(F) Documents of logic synthesis report

```
/*
 *
 * Author: zeyar win
 *         Rochester, NY, USA
 *
 */


`timescale 1ns / 1ns


module test;
```

```
wire dma_grant, tdsp_grant;
wire scan_out0;

reg clk, dma_breq, reset, tdsp_breq;
reg scan_in0, scan_en;

arb top(
        .reset(reset),
        .clk(clk),
        .dma_breq(dma_breq),
        .dma_grant(dma_grant),
        .tdsp_breq(tdsp_breq),
        .tdsp_grant(tdsp_grant),
        .scan_in0(scan_in0),
        .scan_en(scan_en),
        .scan_out0(scan_out0)
    );


reg [4: 0]
    dma_wait,
    tdsp_wait ;

integer
    i,
```

```
    j,
    k,
    l,
    dma_error1bit,
    tdsp_error1bit,
    dma_cnt2,
    tdsp_cnt2,
    dma_cnt,
    tdsp_cnt ;

wire
    grant = dma_grant | tdsp_grant ;

initial
begin
    $timeformat( -9, 2, "ns", 16);
`ifdef SDFSCAN

    $sdf_annotate("sdf/arb_tsmc18_scan.sdf", test.top);
`endif

    clk = 1'b0;
    dma_breq = 1'b0;
    reset = 1'b0;
    tdsp_breq = 1'b0;
```

```
scan_in0 = 1'b0;
scan_en = 1'b0;
dma_cnt = 0 ;
tdsp_cnt = 0 ;
dma_cnt2 = 0;
tdsp_cnt2=0;
dma_wait = $random ;
tdsp_wait = $random ;

@(negedge clk)
 reset = 1'b1 ;
repeat (2)
    @(negedge clk) ;
@(negedge clk)
 reset = 1'b0 ;
repeat (2)
    @(posedge clk) ;

repeat (256)
begin
    @(posedge clk)
     dma_wait <= $random ;
    tdsp_wait <= $random ;
    fork
```

```
    // if output changes > one clock cycle #40ns of brequest high and anothor output changes to zero,flag an error
    // use counter

            dma_request ;
        tdsp_request ;
            dma_check;
            tdsp_check;
    join
    repeat (4)
        @(posedge clk) ;
end
repeat (4)
    @(posedge clk) ;
if (dma_cnt != tdsp_cnt)
begin
    $display(" ** Fails simulation!");
    $display(" ** 256 Individual Bus request cycles generated,");
    $display(" ** (#tdsp grants == %d) != (#dma grants == %d)", tdsp_cnt, dma_cnt);
end
else
begin
    $display(" ** Passes simulation!");
    $display(" ** 256 Individual Bus request cycles generated,");
    $display(" ** (#tdsp grants == %d) == (#dma grants == %d)", tdsp_cnt, dma_cnt);
```

```
    end
    $stop ;
end

always #20
    clk = ~clk ;

task dma_request ;
    begin
        repeat (dma_wait)
            @(posedge clk) ;
        dma_breq <= 1 ;
        $display("%t DMA Bus Request", $time);
        for (i = 0 ; i < (dma_wait + tdsp_wait + 10) ; i = i + 1)
            @(posedge clk)
             if (dma_grant)
             begin
                 dma_cnt = dma_cnt + 1 ;
                 i = (dma_wait + tdsp_wait + 10) ;
             end
         @(posedge clk)
          dma_breq <= 0 ;
        @(posedge clk);
    end
endtask
```

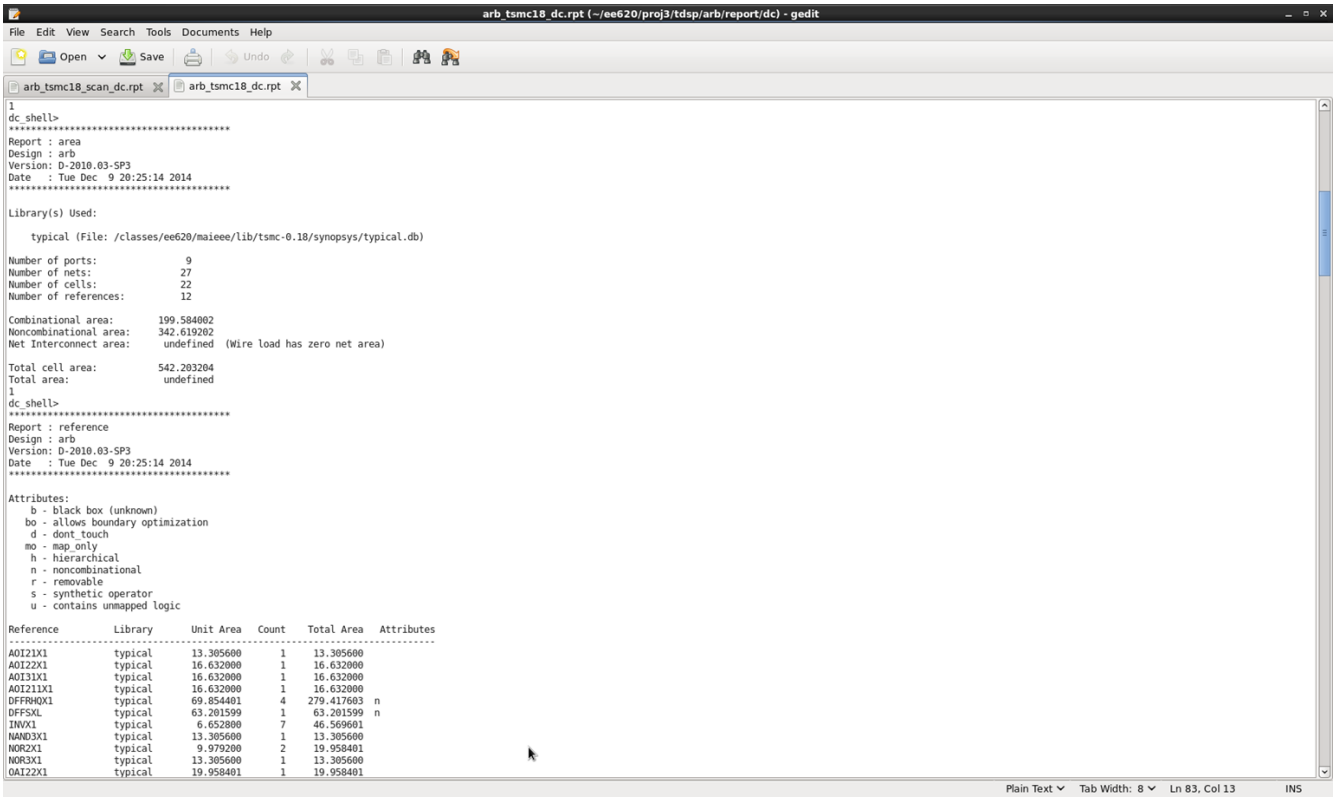
```
task tdsp_request ;
    begin
        repeat (tdsp_wait)
            @(posedge clk) ;
        tdsp_breq <= 1 ;
        $display("%t TDSP Bus Request", $time);
        for (j = 0 ; j < (dma_wait + tdsp_wait + 10) ; j = j + 1)
            @(posedge clk)
             if (tdsp_grant)
             begin
                 tdsp_cnt = tdsp_cnt + 1 ;
                 j = (dma_wait + tdsp_wait + 10) ;
             end
         @(posedge clk)
          tdsp_breq <= 0 ;
        @(posedge clk);
    end
endtask

task dma_check;
            begin
            @(negedge tdsp_grant)
            begin
            dma_cnt2 =0;
```

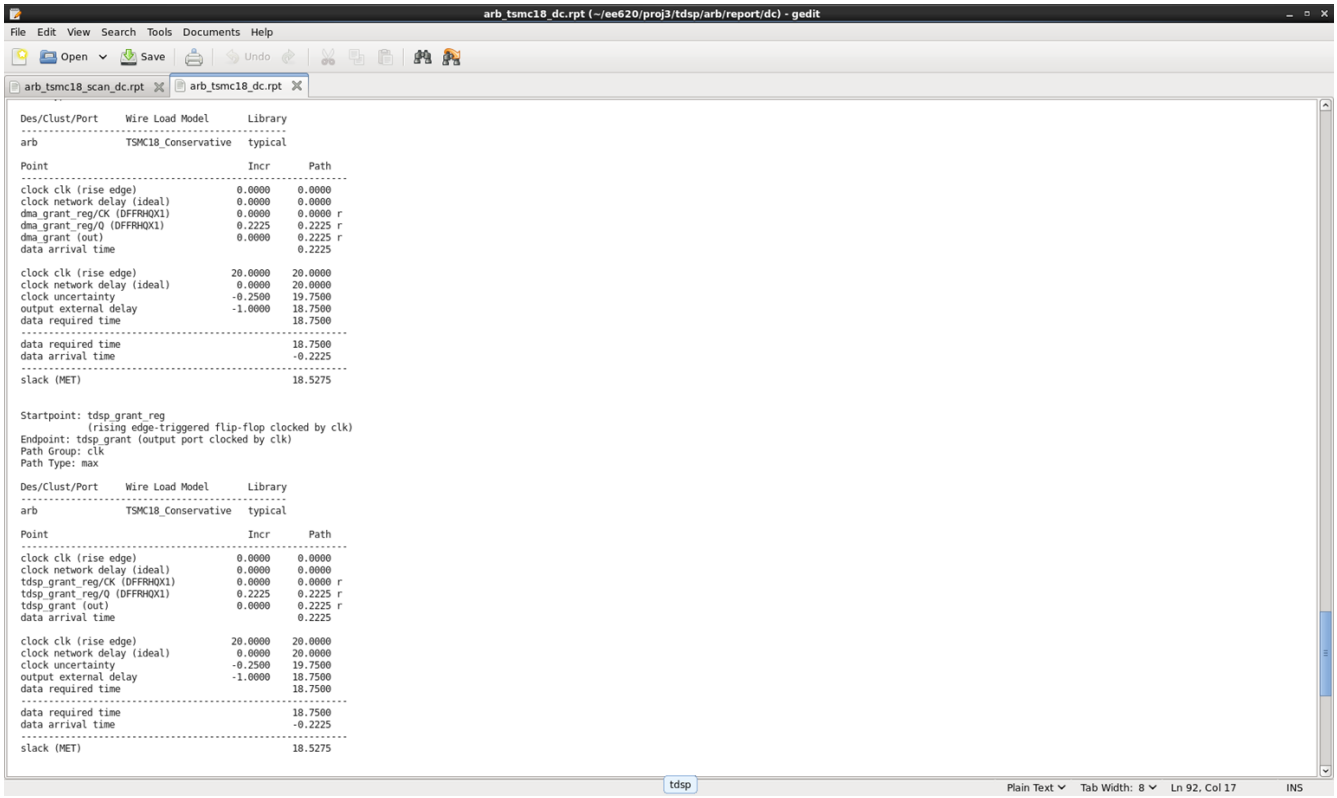
```
            for (k=0 ; k<50 ; k=k+1 )
            @(posedge clk)
            begin
                if(dma_grant != 1 )
                begin
                    dma_cnt2 = dma_cnt2 + 1;
                //  $display("dma counter ", dma_cnt2);
                end
                else if(dma_grant == 1)
                begin
                    k  <=  50;
                end
                if ( dma_cnt2 > 2)
                begin
                    dma_error1bit <=  1;
                    $display( "%t DMA_GRANT ERROR", $time);

                    $stop;
                end
                @(posedge clk);
            end
        end
        end
endtask
```

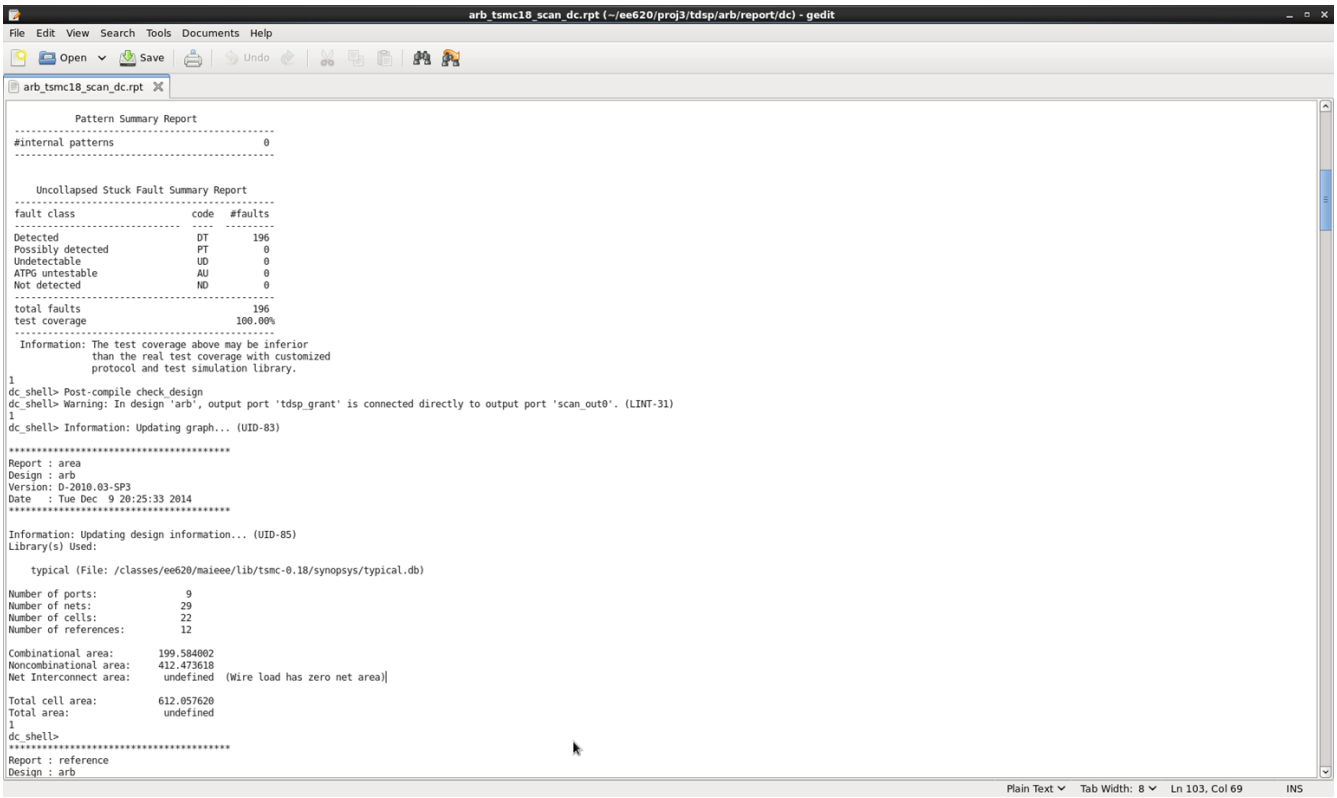
```
task tdsp_check;
        begin
        @(negedge dma_grant)
        begin
                for (l=0 ; l<50 ; l=l+1 )
                @(posedge clk)
                begin
                        if(tdsp_grant != 1 )  //&& dma_breq ==1)
                        begin
                                tdsp_cnt2 = tdsp_cnt2 + 1;
                        //      $display ("tdsp counter", tdsp_cnt2);
                        end
                        else if(tdsp_grant == 1)
                        begin
                                l  <=  50;
                        end
                        if ( tdsp_cnt2 > 1)
                        begin
                                tdsp_error1bit <=  1;
                                $display( "%t TDSP_GRANT ERROR", $time);

                                $stop;
                        end
                        @(posedge clk);
                end
```

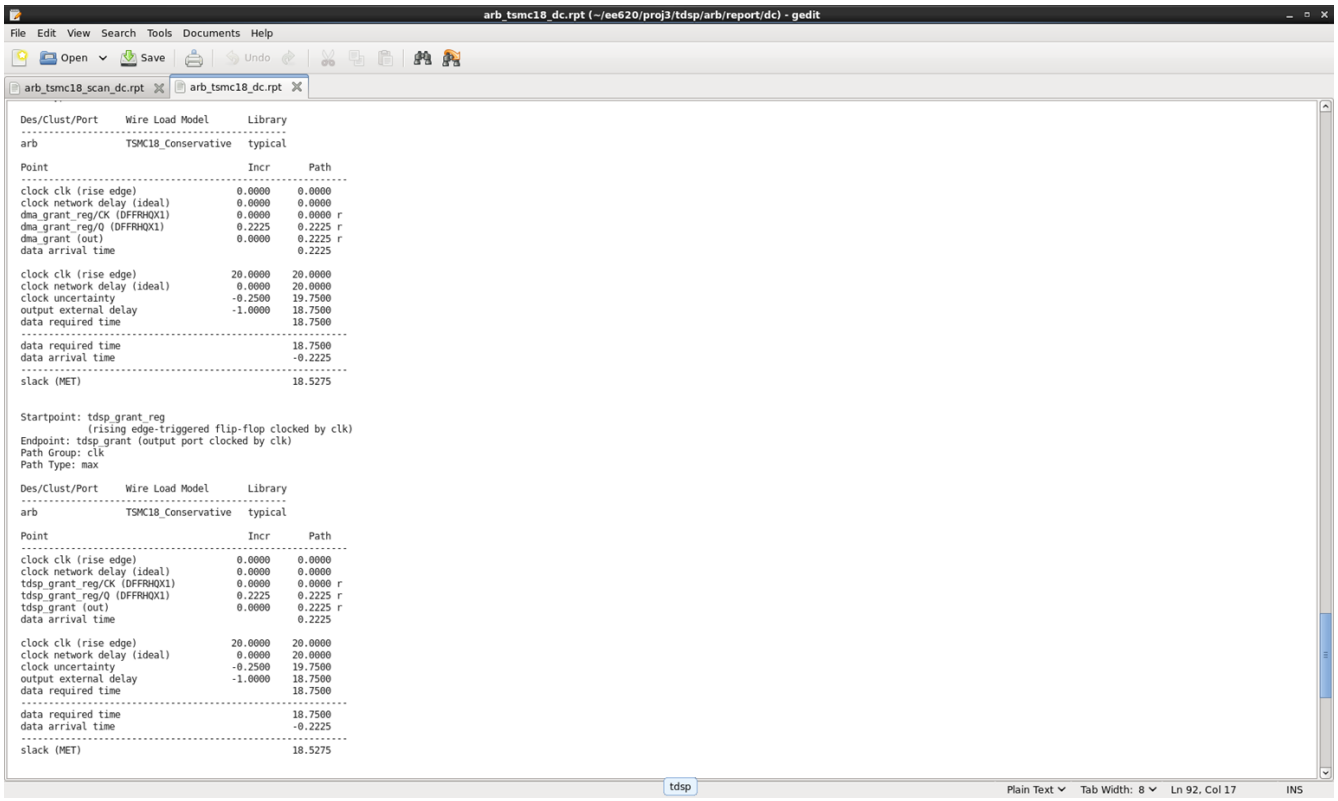
```
        end
        end
endtask
endmodule
```

F Synthesis report

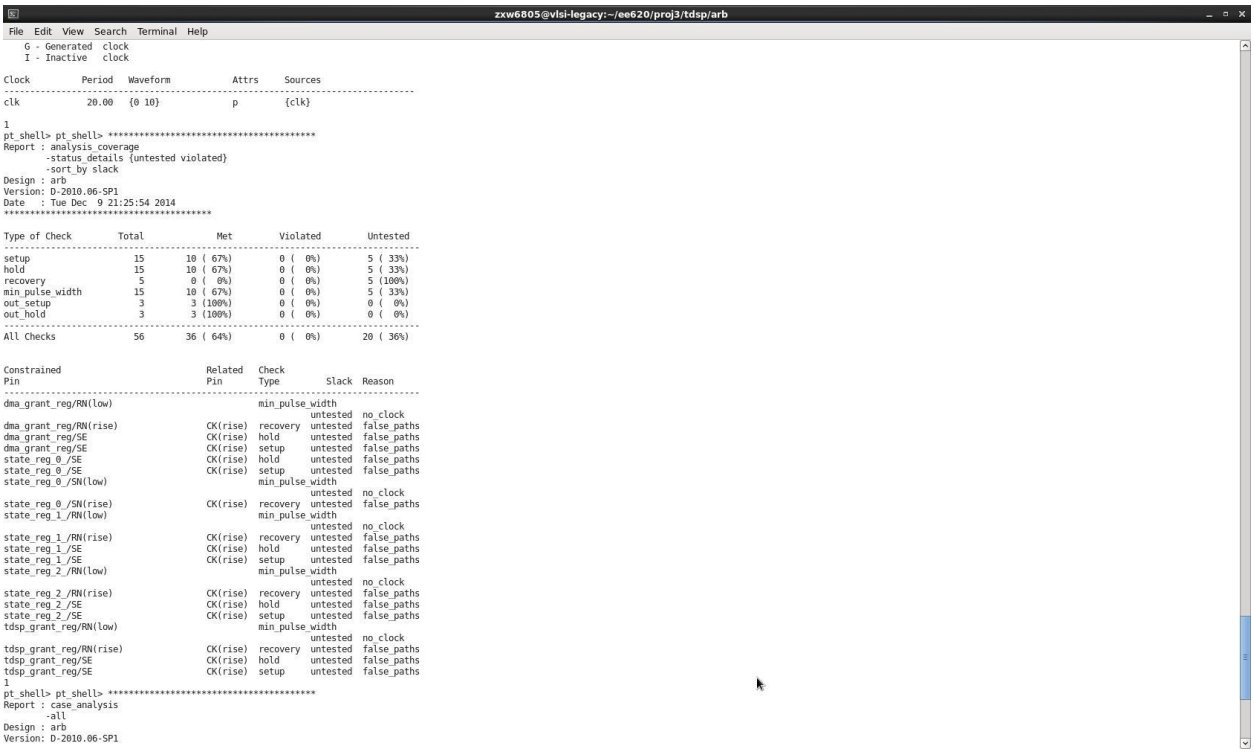


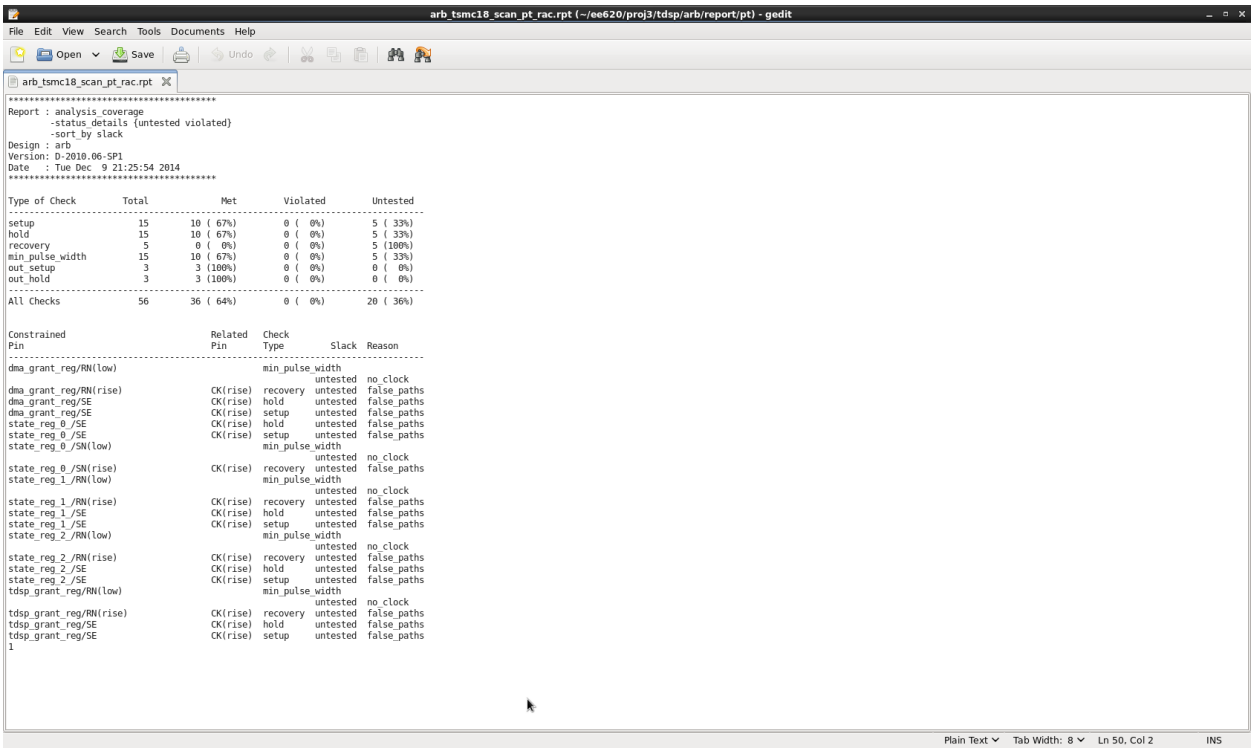

arb_tsmc18_dc.rpt (~/ee620/proj3/tdsp/arb/report/dc) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
arb_tsmc18_scan_dc.rpt
arb_tsmc18_dc.rpt
Des/Clust/Port Wire Load Model Library
arb TSMC18_Conservative typical
Point Incr Path
clock clk (rise edge) 0.0000 0.0000
clock network delay (ideal) 0.0000 0.0000
dma_grant_reg/CK (DFFRHQX1) 0.0000 0.0000 r
dma_grant_reg/Q (DFFRHQX1) 0.2225 0.2225 r
dma_grant (out) 0.0000 0.2225 r
data arrival time 0.2225
clock clk (rise edge) 20.0000 20.0000
clock network delay (ideal) 0.0000 20.0000
clock uncertainty -0.2500 19.7500
output external delay -1.0000 18.7500
data required time 18.7500
data required time 18.7500
data arrival time -0.2225
slack (MET) 18.5275
Startpoint: tdsp_grant_reg
(rising edge-triggered flip-flop clocked by clk)
Endpoint: tdsp_grant (output port clocked by clk)
Path Group: clk
Path Type: max
Des/Clust/Port Wire Load Model Library
arb TSMC18_Conservative typical
Point Incr Path
clock clk (rise edge) 0.0000 0.0000
clock network delay (ideal) 0.0000 0.0000
tdsp_grant_reg/CK (DFFRHQX1) 0.0000 0.0000 r
tdsp_grant_reg/Q (DFFRHQX1) 0.2225 0.2225 r
tdsp_grant (out) 0.0000 0.2225 r
data arrival time 0.2225
clock clk (rise edge) 20.0000 20.0000
clock network delay (ideal) 0.0000 20.0000
clock uncertainty -0.2500 19.7500
output external delay -1.0000 18.7500
data required time 18.7500
data required time 18.7500
data arrival time -0.2225
slack (MET) 18.5275
tdsp
Plain Text Tab Width: 8 Ln 92, Col 17 INS

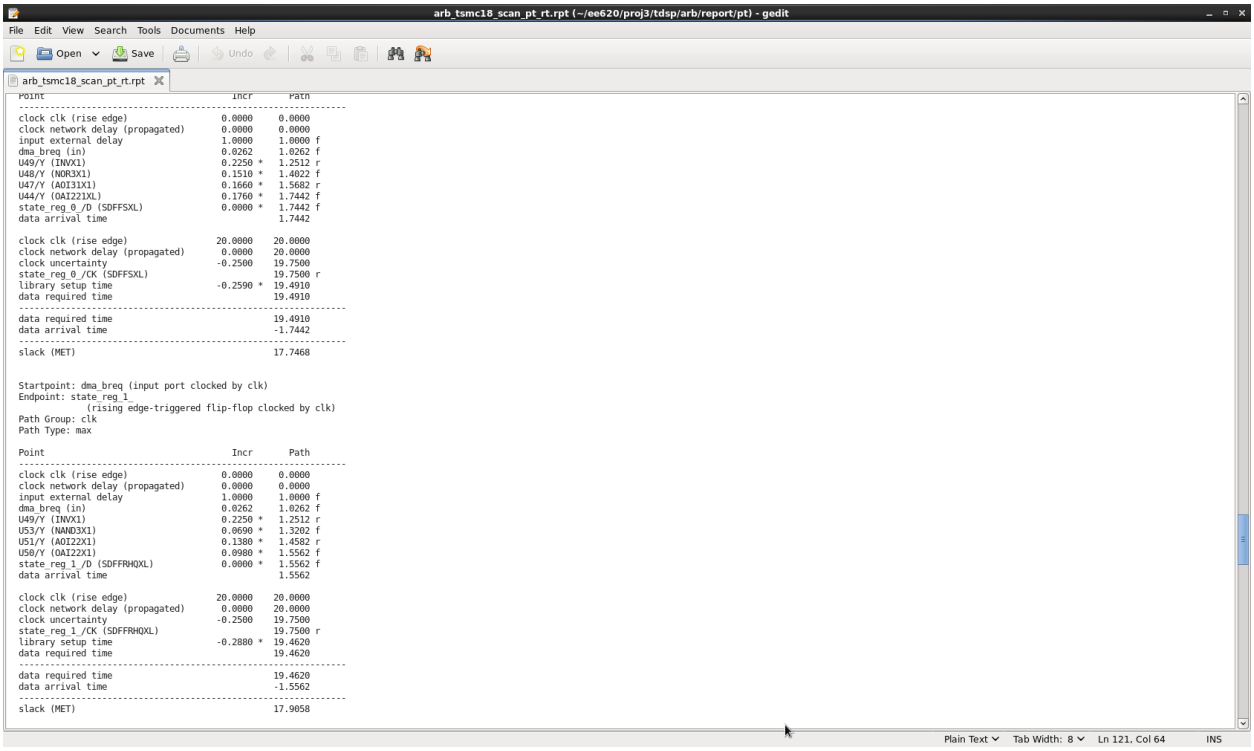

arb_tsmc18_scan_dc.rpt (~/ee620/proj3/tdsp/arb/report/dc) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
arb_tsmc18_scan_dc.rpt
Pattern Summary Report
#internal patterns 0
Uncollapsed Stuck Fault Summary Report
fault class code #faults
Detected DT 196
Possibly detected PT 0
Undetectable UD 0
ATPG untestable AU 0
Not detected ND 0
total faults 196
test coverage 100.00%
Information: The test coverage above may be inferior
than the real test coverage with customized
protocol and test simulation library.
1
dc_shell> Post-compile check_design
dc_shell> Warning: In design 'arb', output port 'tdsp_grant' is connected directly to output port 'scan_out0'. (LINT-31)
1
dc_shell> Information: Updating graph... (UID-83)
Report : area
Design : arb
Version: D-2010.03-SP3
Date : Tue Dec 9 20:25:33 2014
Information: Updating design information... (UID-85)
Library(s) Used:
typical (File: /classes/ee620/maieee/lib/tsmc-0.18/synopsys/typical.db)
Number of ports: 9
Number of nets: 29
Number of cells: 22
Number of references: 12
Combinational area: 199.584002
Noncombinational area: 412.473618
Net Interconnect area: undefined (Wire load has zero net area)
Total cell area: 612.057620
Total area: undefined
1
dc_shell>
Report : reference
Design : arb
Plain Text Tab Width: 8 Ln 103, Col 69 INS



G Various timing analyzer



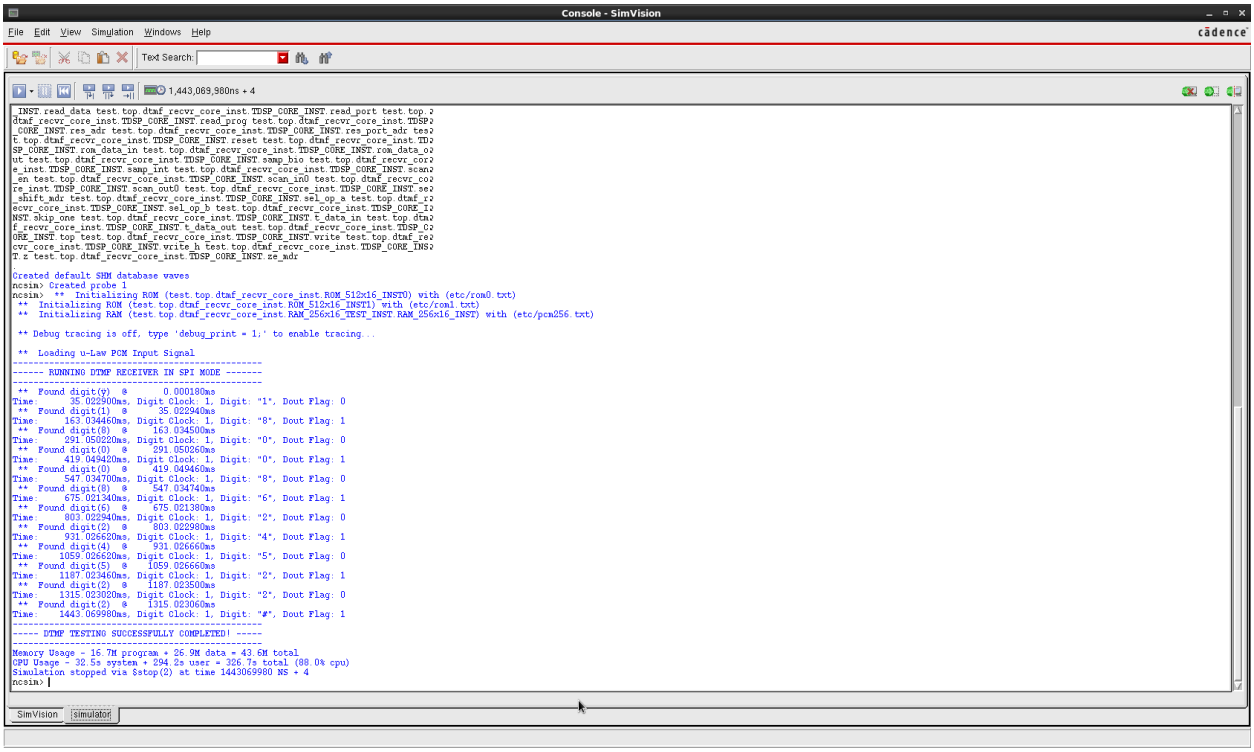


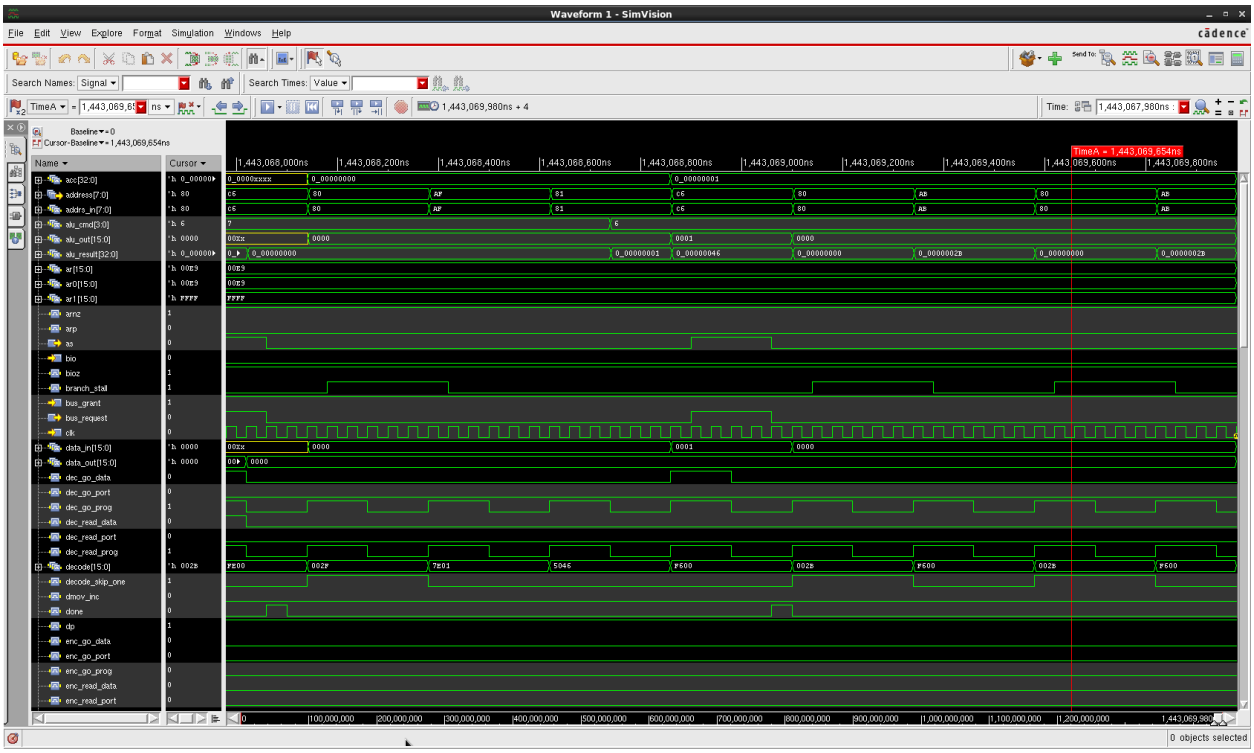


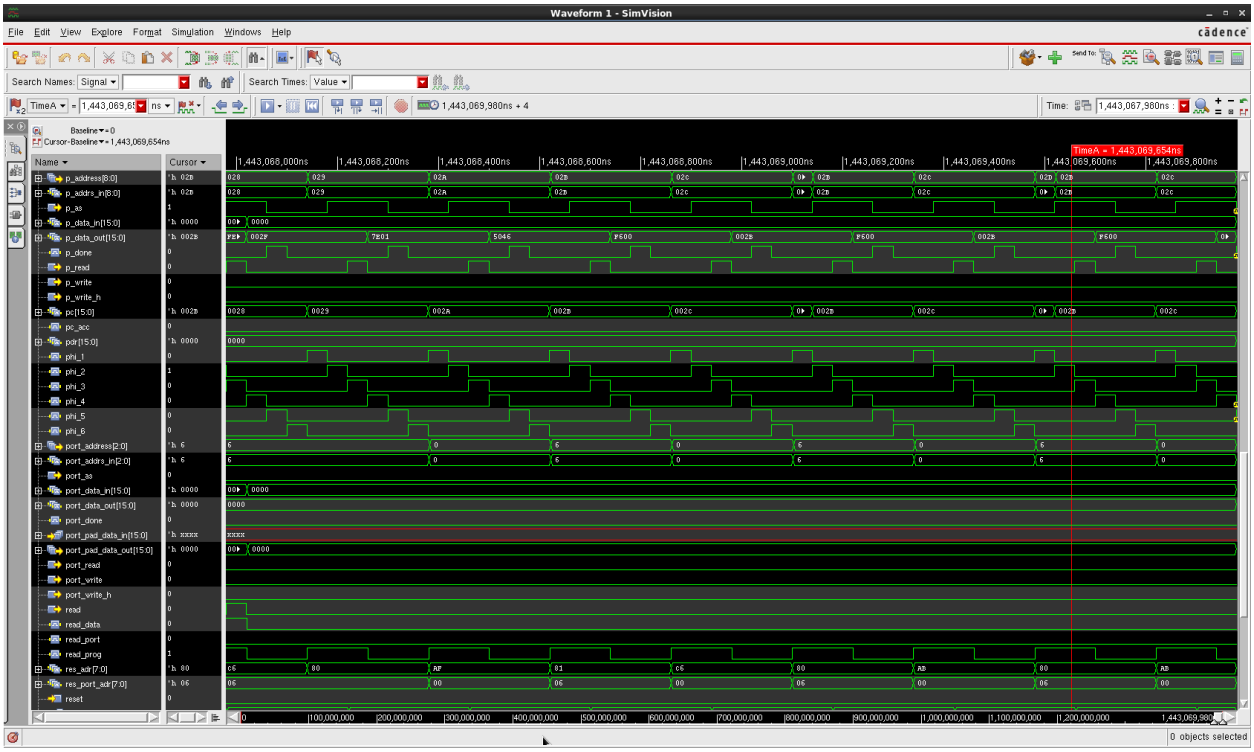
3 DTMF receiver core RTL new memory access bus arbiter (ARB) modules

DTMF simulation RTL

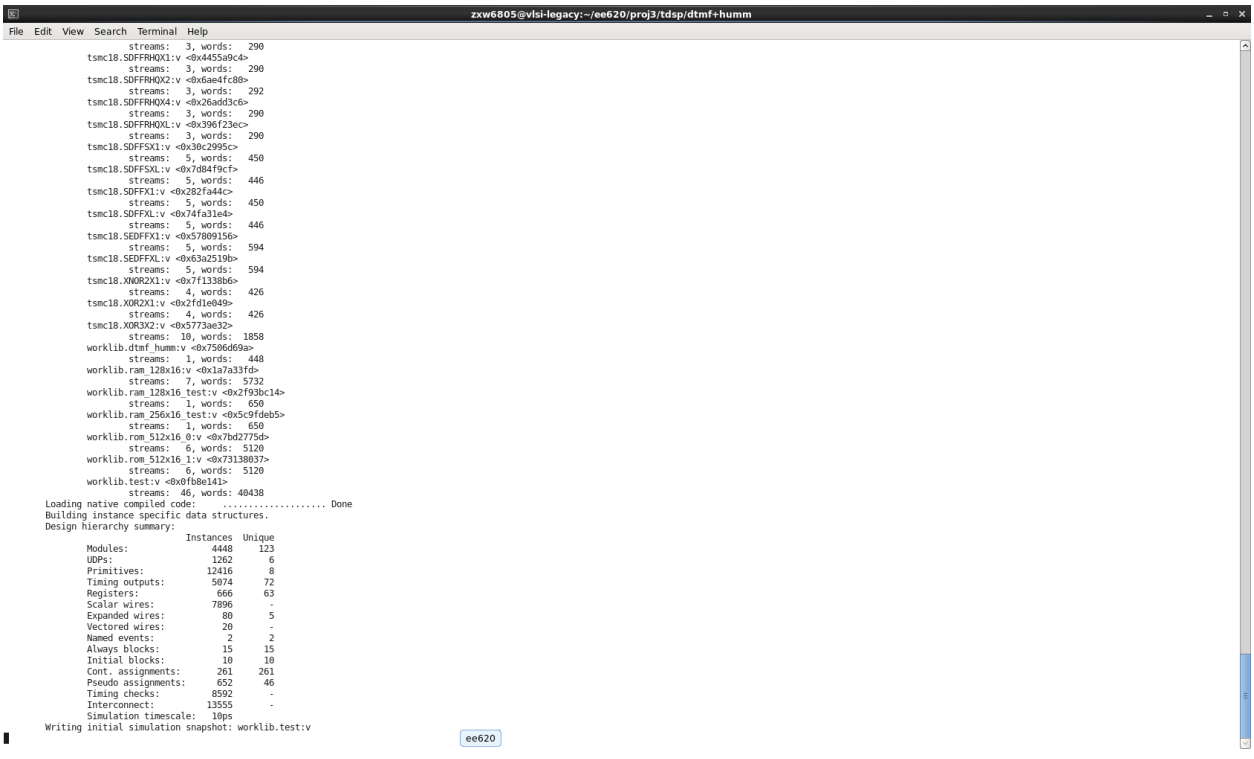
A : screenshots of RTL level simulation log and waveforms

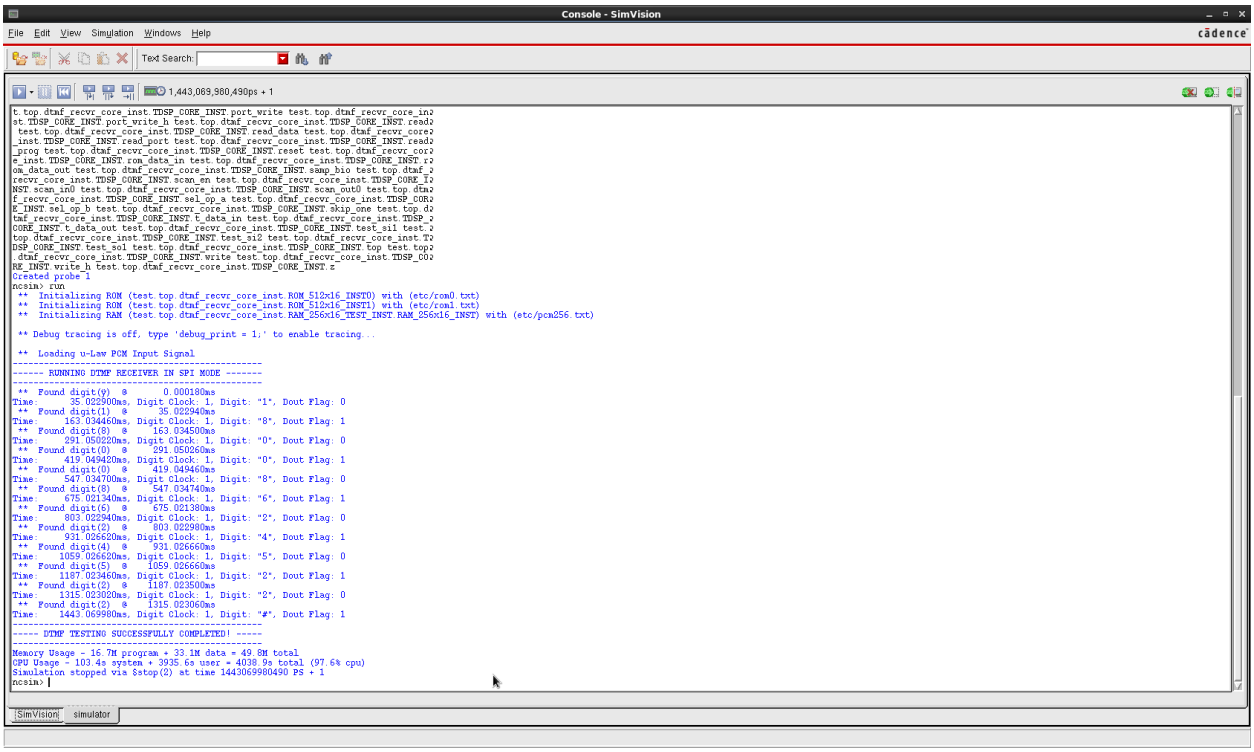


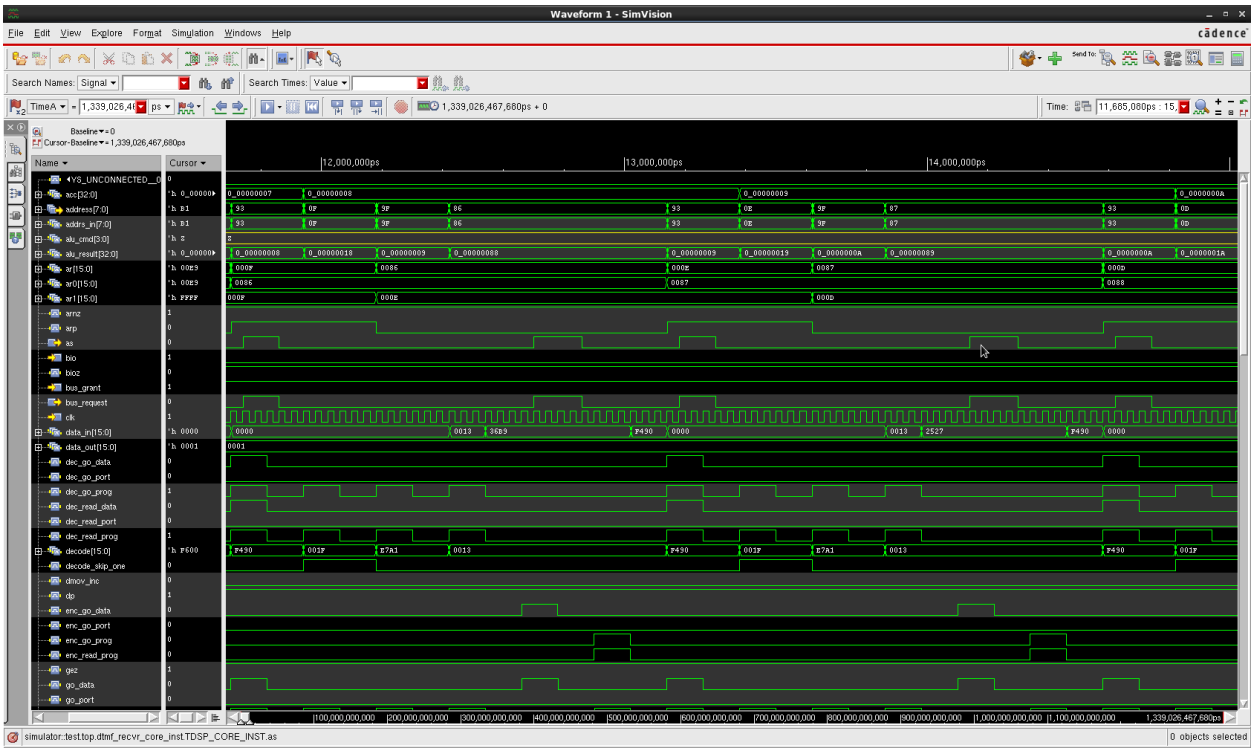


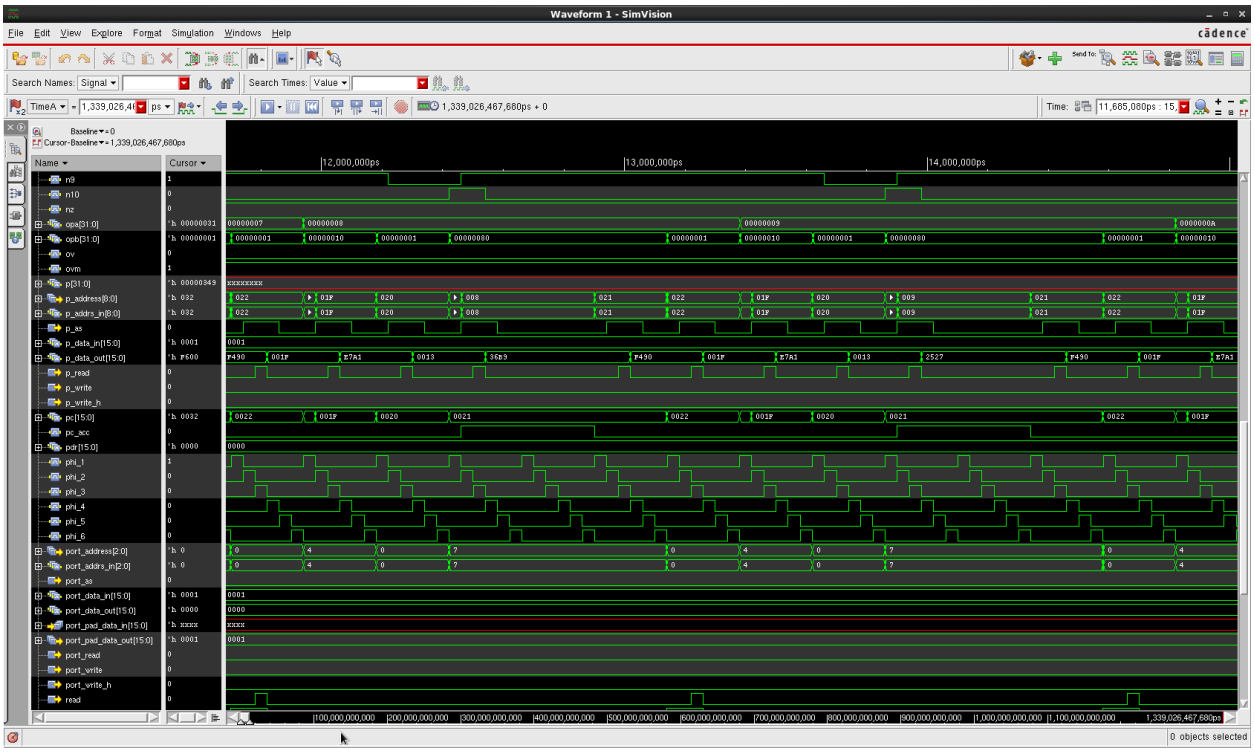


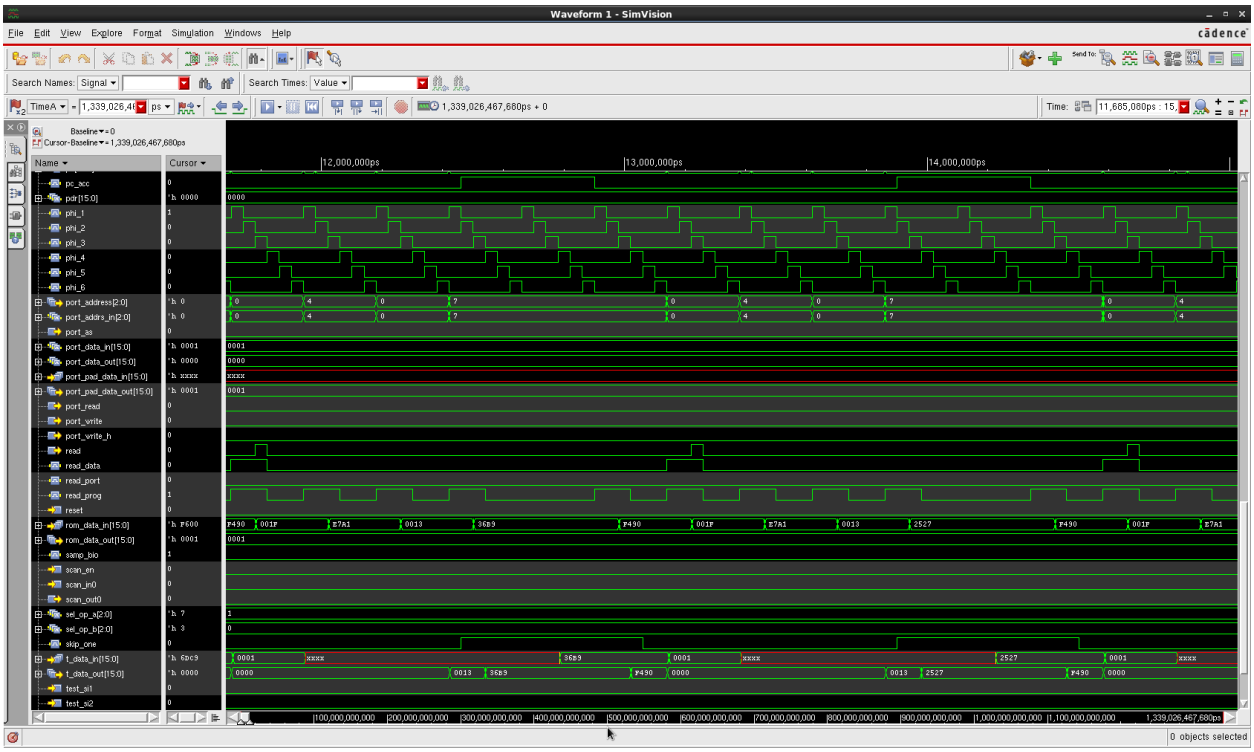
B: Screen shots of Netlist level simulation log and waveforms




Console - SimVision
File Edit View Simulation Windows Help
cadence
Text Search:
1,443,069,980,490ps + 1
t.top.dtmf_recvr_core_inst.TDSP_CORE_INST.port_write test.top.dtmf_recvr_core_in
st.TDSP_CORE_INST.port_write_h test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.read
_test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.read_data test.top.dtmf_recvr_core
_inst.TDSP_CORE_INST.read_port test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.read
_prog test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.reset test.top.dtmf_recvr_cor
e_inst.TDSP_CORE_INST.rom_data_in test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.r
om_data_out test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.samp_bio test.top.dtmf_
recvr_core_inst.TDSP_CORE_INST.scan_en test.top.dtmf_recvr_core_inst.TDSP_CORE_I
NST.scan_in0 test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.scan_out0 test.top.dtm
f_recvr_core_inst.TDSP_CORE_INST.sel_op_a test.top.dtmf_recvr_core_inst.TDSP_COR
E_INST.sel_op_b test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.skip_one test.top.d
tmf_recvr_core_inst.TDSP_CORE_INST.t_data_in test.top.dtmf_recvr_core_inst.TDSP_
CORE_INST.t_data_out test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.test_si1 test.
top.dtmf_recvr_core_inst.TDSP_CORE_INST.test_si2 test.top.dtmf_recvr_core_inst.T
DSP_CORE_INST.test_so1 test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.top test.top
.dtmf_recvr_core_inst.TDSP_CORE_INST.write test.top.dtmf_recvr_core_inst.TDSP_CO
RE_INST.write_h test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.z
Created probe 1
ncsim> run
** Initializing ROM (test.top.dtmf_recvr_core_inst.ROM_512x16_INST0) with (etc/rom0.txt)
** Initializing ROM (test.top.dtmf_recvr_core_inst.ROM_512x16_INST1) with (etc/rom1.txt)
** Initializing RAM (test.top.dtmf_recvr_core_inst.RAM_256x16_TEST_INST.RAM_256x16_INST) with (etc/pcm256.txt)
** Debug tracing is off, type 'debug_print = 1;' to enable tracing...
** Loading u-Law PCM Input Signal
------ RUNNING DTMF RECEIVER IN SPI MODE -------
** Found digit(9) @ 0.000180ms
Time: 35.022900ms, Digit Clock: 1, Digit: "1", Dout Flag: 0
** Found digit(1) @ 35.022940ms
Time: 163.034460ms, Digit Clock: 1, Digit: "8", Dout Flag: 1
** Found digit(8) @ 163.034500ms
Time: 291.050220ms, Digit Clock: 1, Digit: "0", Dout Flag: 0
** Found digit(0) @ 291.050260ms
Time: 419.049420ms, Digit Clock: 1, Digit: "0", Dout Flag: 1
** Found digit(0) @ 419.049460ms
Time: 547.034700ms, Digit Clock: 1, Digit: "8", Dout Flag: 0
** Found digit(8) @ 547.034740ms
Time: 675.021340ms, Digit Clock: 1, Digit: "6", Dout Flag: 1
** Found digit(6) @ 675.021380ms
Time: 803.022940ms, Digit Clock: 1, Digit: "2", Dout Flag: 0
** Found digit(2) @ 803.022980ms
Time: 931.026620ms, Digit Clock: 1, Digit: "4", Dout Flag: 1
** Found digit(4) @ 931.026660ms
Time: 1059.026620ms, Digit Clock: 1, Digit: "5", Dout Flag: 0
** Found digit(5) @ 1059.026660ms
Time: 1187.023460ms, Digit Clock: 1, Digit: "2", Dout Flag: 1
** Found digit(2) @ 1187.023500ms
Time: 1315.023000ms, Digit Clock: 1, Digit: "2", Dout Flag: 0
** Found digit(2) @ 1315.023060ms
Time: 1443.069980ms, Digit Clock: 1, Digit: "#", Dout Flag: 1
----- DTMF TESTING SUCCESSFULLY COMPLETED! -----
Memory Usage - 16.7M program + 33.1M data = 49.8M total
CPU Usage - 103.4s system + 3935.6s user = 4038.9s total (97.6% cpu)
Simulation stopped via $stop(2) at time 1443069980490 PS + 1
ncsim>
SimVision
simulator

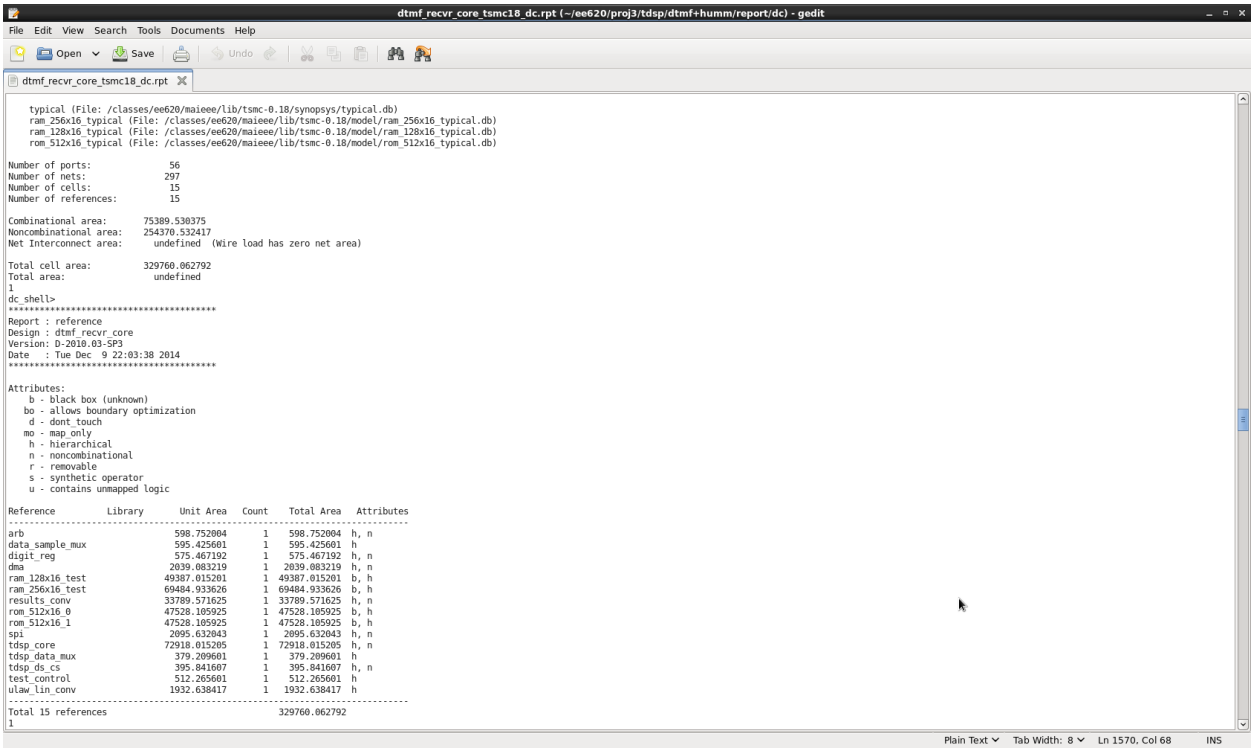

Waveform 1 - SimVision
File Edit View Explore Format Simulation Windows Help
cadence
Search Names: Signal
Search Times: Value
1,339,026,467,680ps + 0
Time: 11,665,080ps : 15,
Baseline +0
Cursor-Baseline +1,339,026,467,680ps
Name
Cursor
12,000,000ps
13,000,000ps
14,000,000ps
acc[32:0]
address[7:0]
addrs_in[7:0]
alu_cmd[3:0]
alu_result[32:0]
ar[15:0]
ar0[15:0]
ar1[15:0]
arnz
arp
as
bio
bioz
bus_grant
bus_request
clk
data_in[15:0]
data_out[15:0]
dec_go_data
dec_go_port
dec_go_prog
dec_read_data
dec_read_port
dec_read_prog
decode[15:0]
decode_skip_one
dmov_inc
dp
enc_go_data
enc_go_port
enc_go_prog
enc_read_prog
gez
go_data
go_port
100,000,000,000
200,000,000,000
300,000,000,000
400,000,000,000
500,000,000,000
600,000,000,000
700,000,000,000
800,000,000,000
900,000,000,000
1,000,000,000,000
1,100,000,000,000
1,339,026,467,680ps
simulator::test.top.dtmf_recvr_core_inst.TDSP_CORE_INST.as
0 objects selected


Waveform 1 - SimVision
File Edit View Explore Format Simulation Windows Help
cadence
Search Names: Signal
Search Times: Value
Name
Cursor
12,000,000ps
13,000,000ps
14,000,000ps
n9
n10
nz
opa[31:0]
opb[31:0]
ov
ovm
p[31:0]
p_address[8:0]
p_addrs_in[8:0]
p_as
p_data_in[15:0]
p_data_out[15:0]
p_read
p_write
p_write_h
pc[15:0]
pc_acc
pdr[15:0]
phi_1
phi_2
phi_3
phi_4
phi_5
phi_6
port_address[2:0]
port_addrs_in[2:0]
port_as
port_data_in[15:0]
port_data_out[15:0]
port_pad_data_in[15:0]
port_pad_data_out[15:0]
port_read
port_write
port_write_h
read
100,000,000,000
200,000,000,000
300,000,000,000
400,000,000,000
500,000,000,000
600,000,000,000
700,000,000,000
800,000,000,000
900,000,000,000
1,000,000,000,000
1,100,000,000,000
1,339,026,467,680ps
0 objects selected

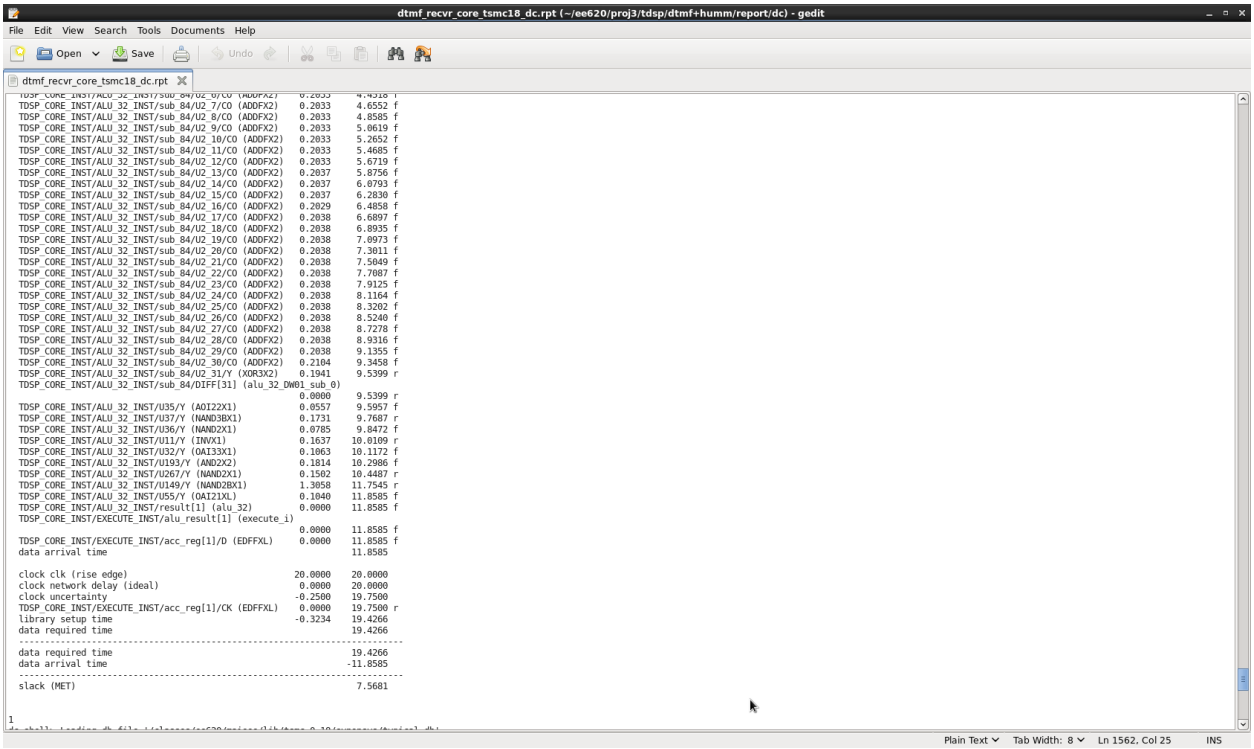

Waveform 1 - SimVision
File Edit View Explore Format Simulation Windows Help
cadence
Search Names: Signal
Search Times: Value
Name
Cursor
12,000,000ps
13,000,000ps
14,000,000ps
pc_acc
pdr[15:0]
phi_1
phi_2
phi_3
phi_4
phi_5
phi_6
port_address[2:0]
port_addrs_in[2:0]
port_as
port_data_in[15:0]
port_data_out[15:0]
port_pad_data_in[15:0]
port_pad_data_out[15:0]
port_read
port_write
port_write_h
read
read_data
read_port
read_prog
reset
rom_data_in[15:0]
rom_data_out[15:0]
samp_bio
scan_en
scan_in0
scan_out0
sel_op_a[2:0]
sel_op_b[2:0]
skip_one
t_data_in[15:0]
t_data_out[15:0]
test_s1
test_s2
100,000,000,000
200,000,000,000
300,000,000,000
400,000,000,000
500,000,000,000
600,000,000,000
700,000,000,000
800,000,000,000
900,000,000,000
1,000,000,000,000
1,100,000,000,000
1,339,026,467,680ps
0 objects selected

C logic synthesis report

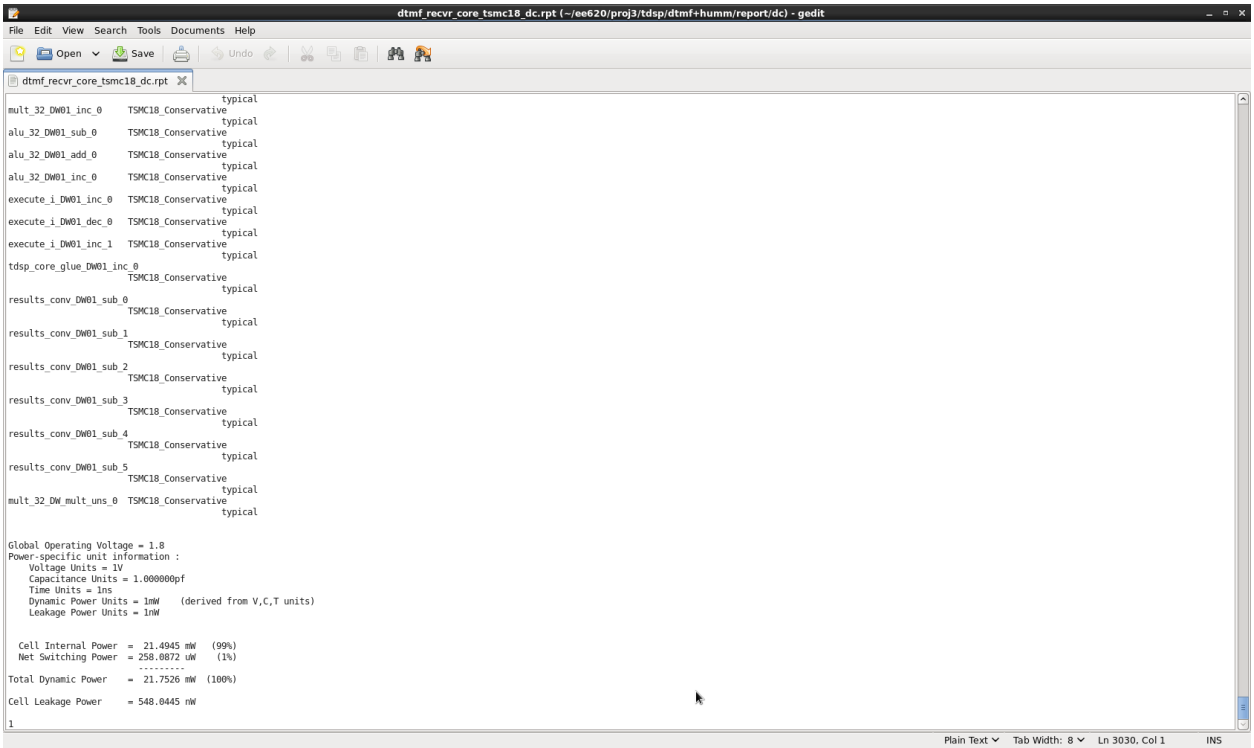
Pre scan total cell area and total number of cells



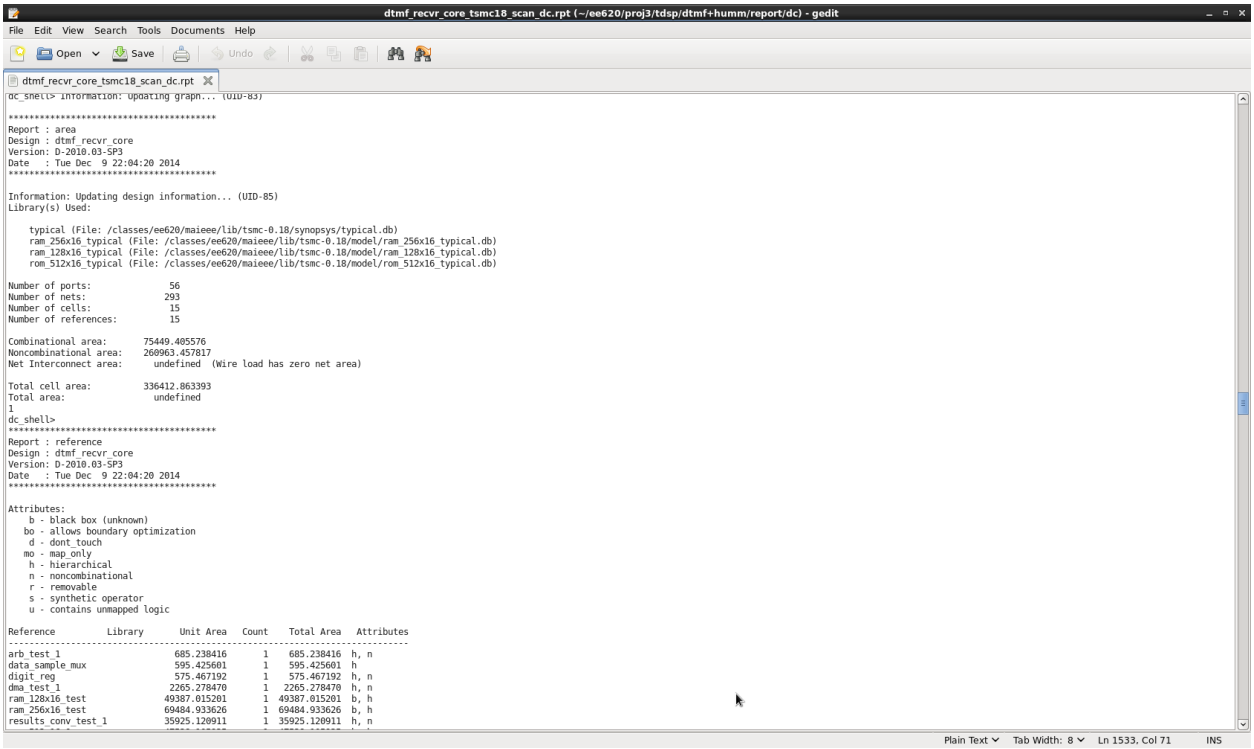
Worst time



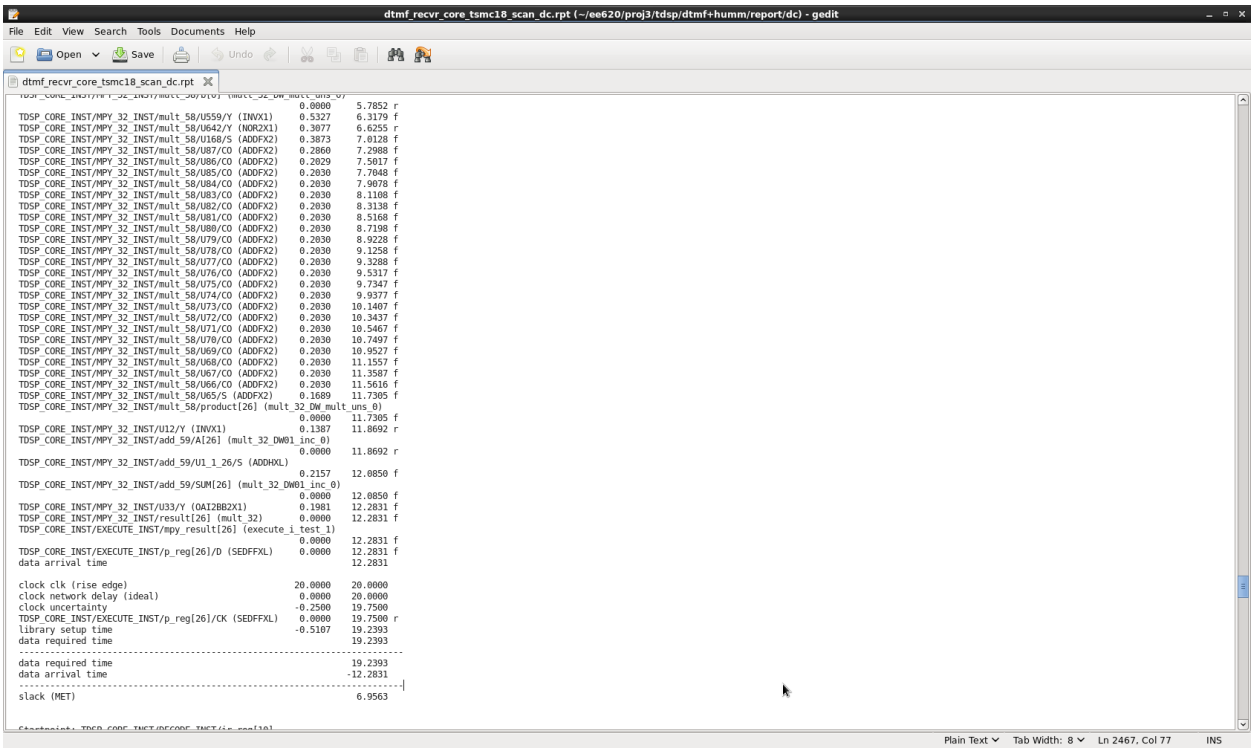
Power consumption



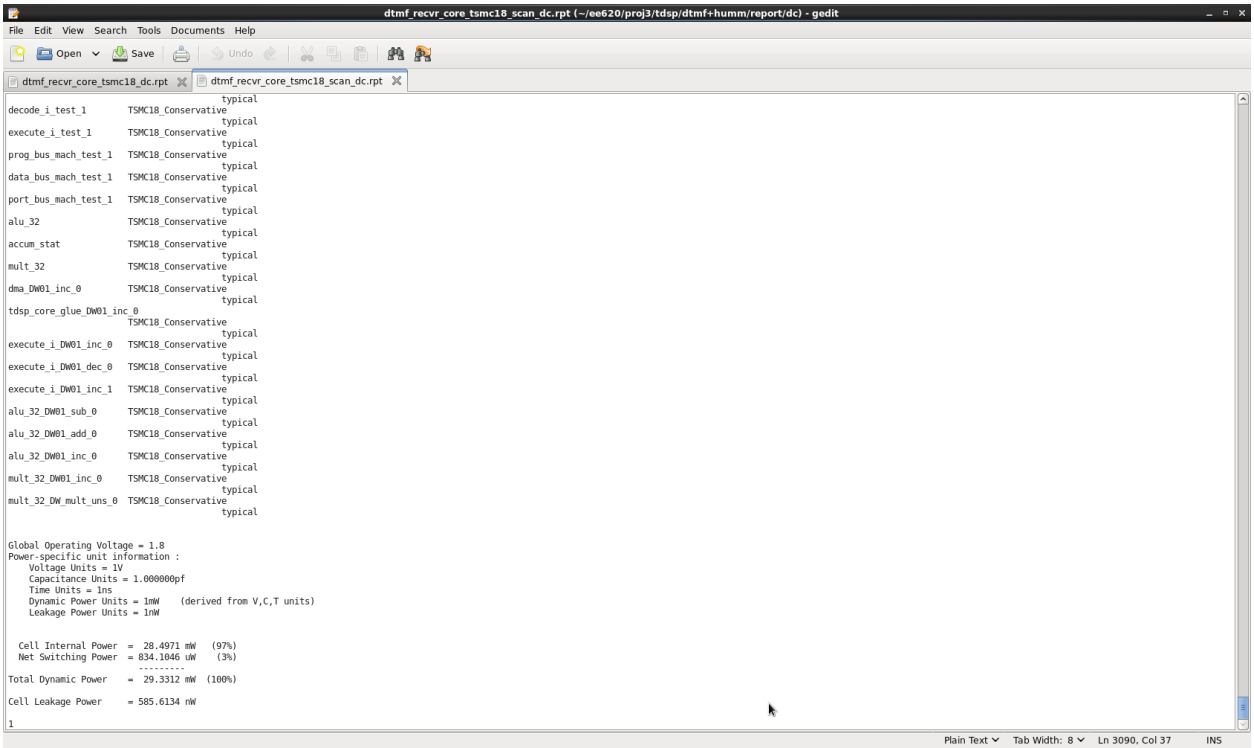
Posts scan total cell area and number of cells



Worst timing path

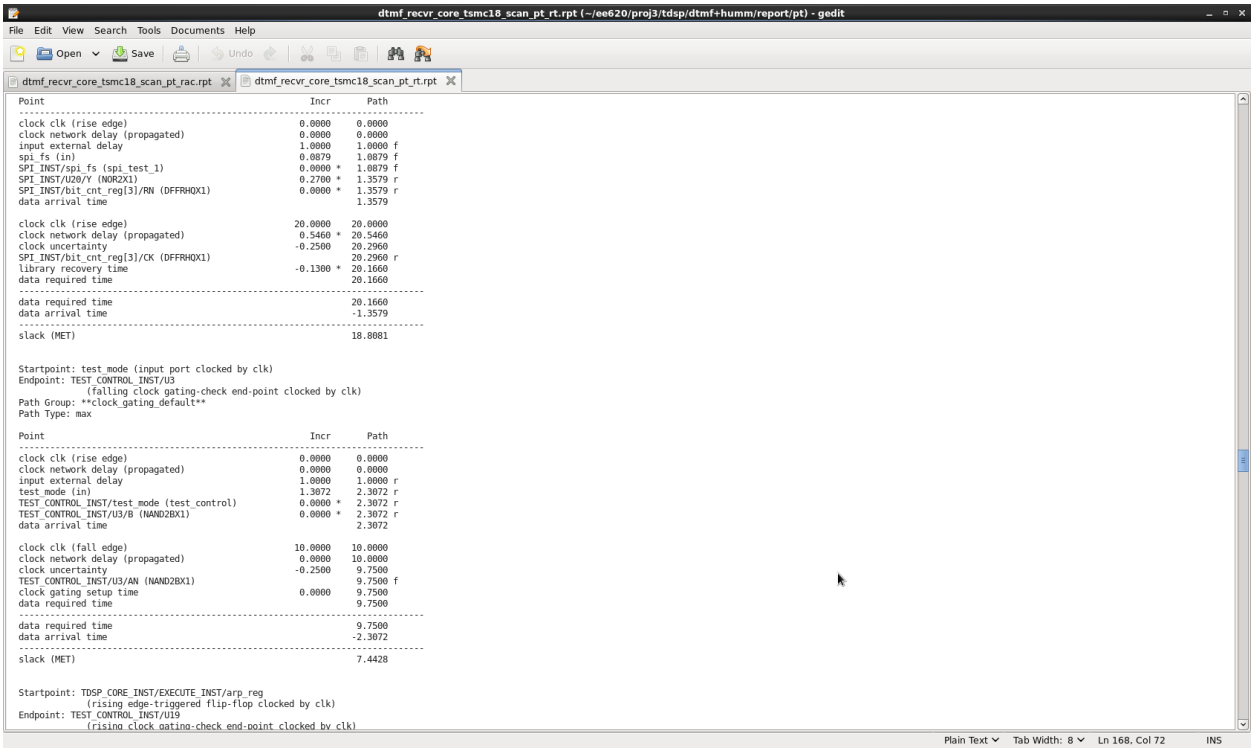


Power consumption

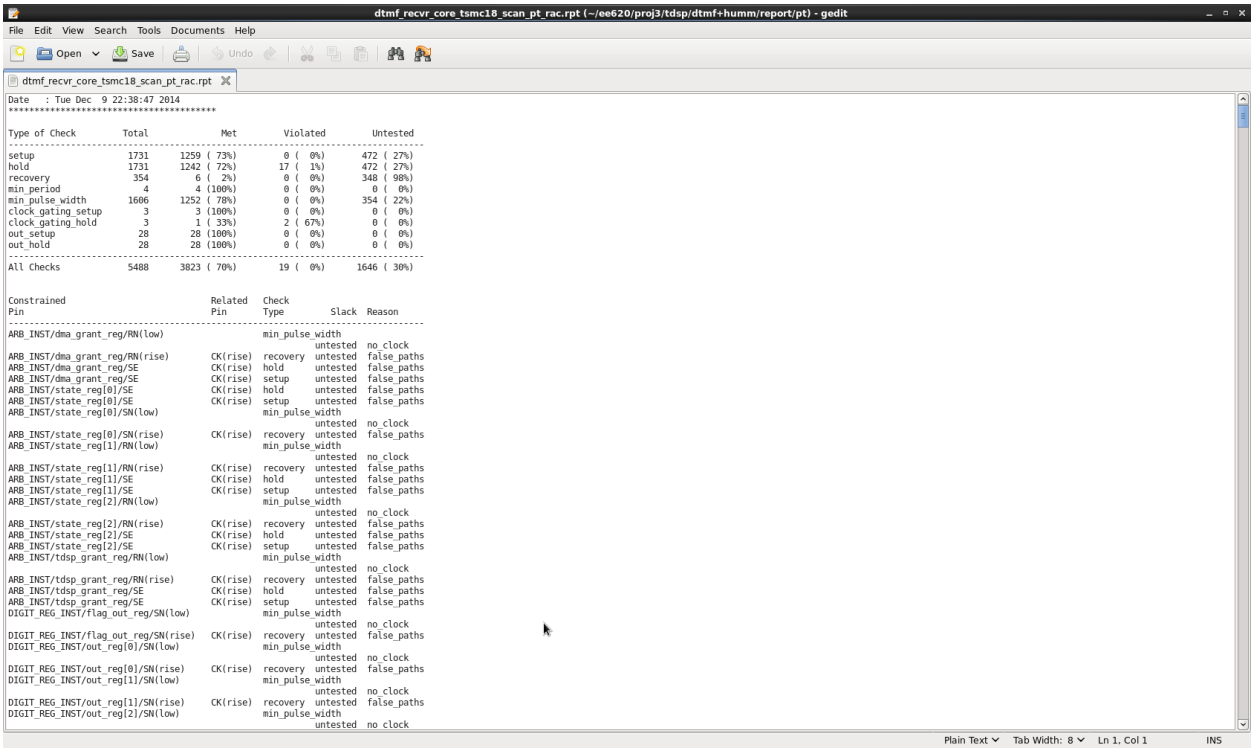


D various timing analyzer reports

Worst case for post-scan



Total timing analysis for post-scan



4 TDSP assembly language test program

A Program Description

TDSP assembly language test program is for testing with X, Y and Z pointing to specific memory locations. In this project, specific bit codes are assigned. $Y = X^2 + (2 * Z) - 1, X = X = 1\ and\ Z = Y$ operations are performed in assembly test program. If finite state machine Verilog and the test bench of assembly are implemented correctly, the # spectrum should be seen in simulation console. The DTMF_TDSP net list simulation takes about 1 hour to finish.

B Assembly language test program source code

```
X = 0x6;

LACK 1;

SACL X;


Y = 0x7;

LACK 0;

SACL Y;


Z = 0x8;

LACK 0;

SACL Z;


A= 0x9

LACK 1;

SACL A;
```

```
two =0x2;

;larp ar0, 0x40;


waitTime =0x1E;

;REGA  =0;

;LACK  0x40; 64 decmial value is stored in register A

;SACL REGA;



page0  = 0        ;

base_page0  = 0x000;

base_page1      =          0x080    ; memory page 1

ptr    =21

rcc_ptr    =22


rcc_697Hz        = (0x00e0 & 0x07f)                    ;

rcc_770Hz        = (0x00e1 & 0x07f)                    ;

rcc_852Hz        = (0x00e2 & 0x07f)                    ;

rcc_941Hz        = (0x00e3 & 0x07f)                    ;

rcc_1209Hz       = (0x00e4 & 0x07f)                    ;

rcc_1336Hz       = (0x00e5 & 0x07f)                    ;

rcc_1477Hz       = (0x00e6 & 0x07f)                    ;

rcc_1633Hz       = (0x00e7 & 0x07f)                    ;

rcc_kick= (0x00e8 & 0x07f)                        ;
```

```
rcc_len         = (rcc_kick - rcc_697Hz) ;

;

;pointer


;rc_697Hz       = 0x7fff * fp_rc_697Hz                  ;

;rc_770Hz       = 0x7fff * fp_rc_770Hz                  ;

;rc_852Hz       = 0x7fff * fp_rc_852Hz                  ;

;rc_941Hz       = 0x7fff * fp_rc_941Hz                  ;

;rc_1209Hz      = 0x7fff * fp_rc_1209Hz                 ;

;rc_1336Hz      = 0x7fff * fp_rc_1336Hz                 ;

;rc_1477Hz      = 0x7fff * fp_rc_1477Hz                 ;

;rc_1633Hz      = 0x7fff * fp_rc_1633Hz                 ;


powquiet1       =       60

powquiet2       =       61

powquiet3       =       62

powquiet4       =       63

powquiet5       =       64

powquiet6       =       65

powquiet7       =       66

powquiet8       =       67


pow1    =       68

pow2    =       69

pow3    =       70
```

pow4 = 71

pow5 = 72

pow6 = 73

pow7 = 74

pow8 = 75

LACK 0

sacl powquiet1

sacl powquiet2

sacl powquiet3

sacl powquiet4

sacl powquiet5

sacl powquiet6

sacl powquiet7

sacl powquiet8

sacl pow1

sacl pow2

sacl pow3

sacl pow5

sacl pow6

sacl pow8

bss= 0x3

csa= 0x4

lack 0x07

```
sacl bss

lack 0xff

sacl csa

ZALH bss

OR csa

sacl pow4

sacl pow7



larp ar0

lark ar0,0x40


loop:

        lt X        ;

        mpy X

        pac

        lt Z

        mpy two

        apac        ;loading X^2+2Z into accumualtor

        sub A

              ; Y = X^2+(2*Z)-1

    SACL Y;

    LAC X;

    ADD A;

    SACL X;         X=X+1
```

```
        LAC Y ;
        SACL Z;  Z= Y

        banz loop,*-,ar0;     ; use auxiliar register to loop


spectrum:
quiet:    lark      ar1,rcc_len                 ; length of rcc register block
                    lark      ar0,(rcc_697Hz+base_page1)     ; starting address
                    sar               ar0,rcc_ptr
                    lark      ar0,(powquiet1+base_page0)
                    sar               ar0,ptr
quiet_l: lar                ar0,ptr
                    zals      *+
                    sar               ar0,ptr
                    lar     ar0,rcc_ptr
                    sacl      *+,ar1
                    sar               ar0,rcc_ptr
                    banz      quiet_l,*-,ar0       ; make quiet tone

quiet2:  lark       ar1,rcc_len                 ; length of rcc register block
                    lark      ar0,(rcc_697Hz+base_page1)     ; starting address
                    sar               ar0,rcc_ptr
                    lark      ar0,(powquiet1+base_page0)
```

```
                    sar     ar0,ptr
quiet_2:    lar     ar0,ptr
                    zals    *+
                    sar     ar0,ptr
                    lar     ar0,rcc_ptr
                    sacl    *+,ar1
                    sar     ar0,rcc_ptr
                    banz    quiet_2,*-,ar0      ; make quiet tone twice
                lark ar0,waitTime

wait0:              banz wait0,*-,ar0       ;create delay


poundsign1: lark    ar1,rcc_len             ; length of rcc register block
                    lark    ar0,(rcc_697Hz+base_page1)  ; starting address
                    sar     ar0,rcc_ptr
                    lark    ar0,(pow1+base_page0)
                    sar     ar0,ptr

poundsign_1: lar    ar0,ptr
                    zals    *+
                    sar     ar0,ptr
                    lar     ar0,rcc_ptr
                    sacl    *+,ar1
                    sar     ar0,rcc_ptr
```

```
                banz    poundsign_1,*-,ar0      ; make quiet tone

poundsign2:     lark    ar1,rcc_len             ; length of rcc register block
                lark    ar0,(rcc_697Hz+base_page1)    ; starting address
                sar             ar0,rcc_ptr
                lark    ar0,(pow1+base_page0)
                sar             ar0,ptr
poundsign_2:    lar             ar0,ptr
                zals    *+
                sar             ar0,ptr
                lar   ar0,rcc_ptr
                sacl    *+,ar1
                sar             ar0,rcc_ptr
                banz    poundsign_2,*-,ar0      ; make #spectrum twice

                lark ar0,waitTime
wait1:

                banz wait1,*-,ar0       ;wait for 30 cycles

quiet3:  lark   ar1,rcc_len             ; length of rcc register block
                lark    ar0,(rcc_697Hz+base_page1)    ; starting address
                sar             ar0,rcc_ptr
                lark    ar0,(powquiet1+base_page0)
                sar             ar0,ptr
```

```
quiet_3:        lar             ar0,ptr
                zals    *+
                sar             ar0,ptr
                lar     ar0,rcc_ptr
                sacl    *+,ar1
                sar             ar0,rcc_ptr
                banz    quiet_3,*-,ar0    ; make quiet tone



quiet4:  lark   ar1,rcc_len             ; length of rcc register block
                lark    ar0,(rcc_697Hz+base_page1)    ; starting address
                sar             ar0,rcc_ptr
                lark    ar0,(powquiet1+base_page0)
                sar             ar0,ptr
quiet_4:        lar             ar0,ptr
                zals    *+
                sar             ar0,ptr
                lar     ar0,rcc_ptr
                sacl    *+,ar1
                sar             ar0,rcc_ptr
                banz    quiet_4,*-,ar0    ; make quiet tone twice
stop:           b stop
```

C Assembly Language Test Program Merged Listing

```
  tdspasm, RCS v1.1.1.1
  Listing for module: "zxw6805_test"
  Prepared: Mon Dec  8 17:43:18 EST 2014
*****************************************
```

```
0000 7e01              LACK    1
0001 5006              SACL    X
0002 7e00              LACK    0
0003 5007              SACL    Y
0004 7e00              LACK    0
0005 5008              SACL    Z
0006 7e01              LACK    1
0007 5009              SACL    A
0008 7e00              LACK    0
0009 503c              sacl    powquiet1
000a 503d              sacl    powquiet2
000b 503e              sacl    powquiet3
000c 503f              sacl    powquiet4
000d 5040              sacl    powquiet5
000e 5041              sacl    powquiet6
000f 5042              sacl    powquiet7
0010 5043              sacl    powquiet8
0011 5044              sacl    pow1
0012 5045              sacl    pow2
0013 5046              sacl    pow3
0014 5048              sacl    pow5
0015 5049              sacl    pow6
0016 504b              sacl    pow8
0017 7e07              lack    0x07
0018 5003              sacl    bss
0019 7eff              lack    0xff
001a 5004              sacl    csa
001b 6503              ZALH    bss
001c 7a04              OR      csa
001d 5047              sacl    pow4
001e 504a              sacl    pow7
001f 6880              larp    ar0
0020 7040              lark    ar0,0x40
0021            loop:
0021 4006              lt      X
0022 6d06              mpy     X
0023 7f8e              pac
0024 4008              lt      Z
0025 6d02              mpy     two
0026 7f8f              apac
0027 1009              sub     A
0028 5007              SACL    Y
0029 2006              LAC     X
002a 0009              ADD     A
002b 5006              SACL    X
002c 2007              LAC     Y
002d 5008              SACL    Z
002e f490              banz    loop,*-,ar0
002f 0021
```

```
0030             spectrum:
0030 7108          quiet: lark       ar1,rcc_len
0031 70e0                 lark       ar0,(rcc_697Hz+base_page1)
0032 3016                 sar        ar0,rcc_ptr
0033 703c                 lark       ar0,(powquiet1+base_page0)
0034 3015                 sar        ar0,ptr
0035 3815        quiet_l: lar        ar0,ptr
0036 66a8                 zals       *+
0037 3015                 sar        ar0,ptr
0038 3816                 lar        ar0,rcc_ptr
0039 50a1                 sacl       *+,ar1
003a 3016                 sar        ar0,rcc_ptr
003b f490                 banz       quiet_l,*-,ar0
003c 0035
003d 7108         quiet2: lark       ar1,rcc_len
003e 70e0                 lark       ar0,(rcc_697Hz+base_page1)
003f 3016                 sar        ar0,rcc_ptr
0040 703c                 lark       ar0,(powquiet1+base_page0)
0041 3015                 sar        ar0,ptr
0042 3815        quiet_2: lar        ar0,ptr
0043 66a8                 zals       *+
0044 3015                 sar        ar0,ptr
0045 3816                 lar        ar0,rcc_ptr
0046 50a1                 sacl       *+,ar1
0047 3016                 sar        ar0,rcc_ptr
0048 f490                 banz       quiet_2,*-,ar0
0049 0042
004a 701e                 lark       ar0,waitTime
004b f490          wait0: banz       wait0,*-,ar0
004c 004b
004d 7108      poundsign1: lark       ar1,rcc_len
004e 70e0                 lark       ar0,(rcc_697Hz+base_page1)
004f 3016                 sar        ar0,rcc_ptr
0050 7044                 lark       ar0,(pow1+base_page0)
0051 3015                 sar        ar0,ptr
0052 3815     poundsign_1: lar        ar0,ptr
0053 66a8                 zals       *+
0054 3015                 sar        ar0,ptr
0055 3816                 lar        ar0,rcc_ptr
0056 50a1                 sacl       *+,ar1
0057 3016                 sar        ar0,rcc_ptr
0058 f490                 banz       poundsign_1,*-,ar0
0059 0052
005a 7108      poundsign2: lark       ar1,rcc_len
005b 70e0                 lark       ar0,(rcc_697Hz+base_page1)
005c 3016                 sar        ar0,rcc_ptr
005d 7044                 lark       ar0,(pow1+base_page0)
005e 3015                 sar        ar0,ptr
005f 3815     poundsign_2: lar        ar0,ptr
```

```
0060 66a8              zals    *+
0061 3015              sar     ar0,ptr
0062 3816              lar     ar0,rcc_ptr
0063 50a1              sacl    *+,ar1
0064 3016              sar     ar0,rcc_ptr
0065 f490              banz    poundsign_2,*-,ar0
0066 005f
0067 701e              lark    ar0,waitTime
0068            wait1:
0068 f490              banz    wait1,*-,ar0
0069 0068
006a 7108       quiet3: lark   ar1,rcc_len
006b 70e0              lark    ar0,(rcc_697Hz+base_page1)
006c 3016              sar     ar0,rcc_ptr
006d 703c              lark    ar0,(powquiet1+base_page0)
006e 3015              sar     ar0,ptr
006f 3815       quiet_3: lar    ar0,ptr
0070 66a8              zals    *+
0071 3015              sar     ar0,ptr
0072 3816              lar     ar0,rcc_ptr
0073 50a1              sacl    *+,ar1
0074 3016              sar     ar0,rcc_ptr
0075 f490              banz    quiet_3,*-,ar0
0076 006f
0077 7108       quiet4: lark   ar1,rcc_len
0078 70e0              lark    ar0,(rcc_697Hz+base_page1)
0079 3016              sar     ar0,rcc_ptr
007a 703c              lark    ar0,(powquiet1+base_page0)
007b 3015              sar     ar0,ptr
007c 3815       quiet_4: lar    ar0,ptr
007d 66a8              zals    *+
007e 3015              sar     ar0,ptr
007f 3816              lar     ar0,rcc_ptr
0080 50a1              sacl    *+,ar1
0081 3016              sar     ar0,rcc_ptr
0082 f490              banz    quiet_4,*-,ar0
0083 007c
0084 f500        stop: b       stop
0085 0084
```

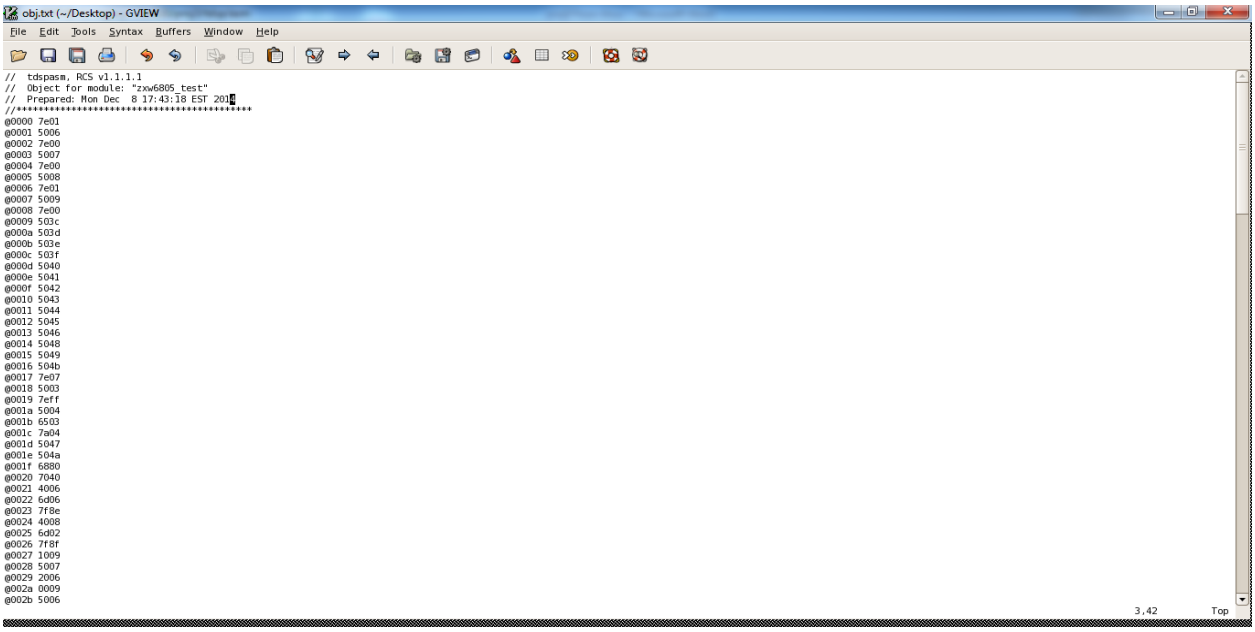
D. Assembly Language Test Program Symbol Table

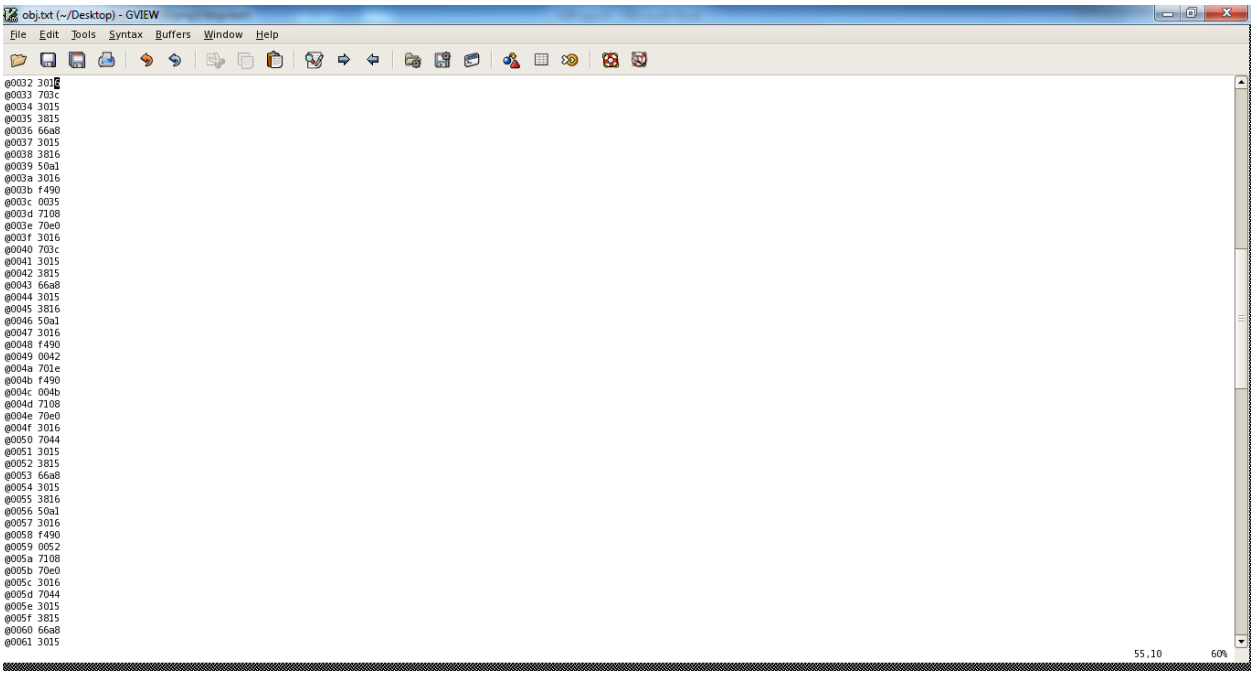
```
                tdspasm, RCS v1.1.1
 Symbol listing for module: "zxw6805_test"
 Prepared: Mon Dec  8 17:43:18 EST 2014
 * <symbol> = <hex> (<octal>) (<decimal>) <R,A>
 * where: R == relocatable, A == absolute
```

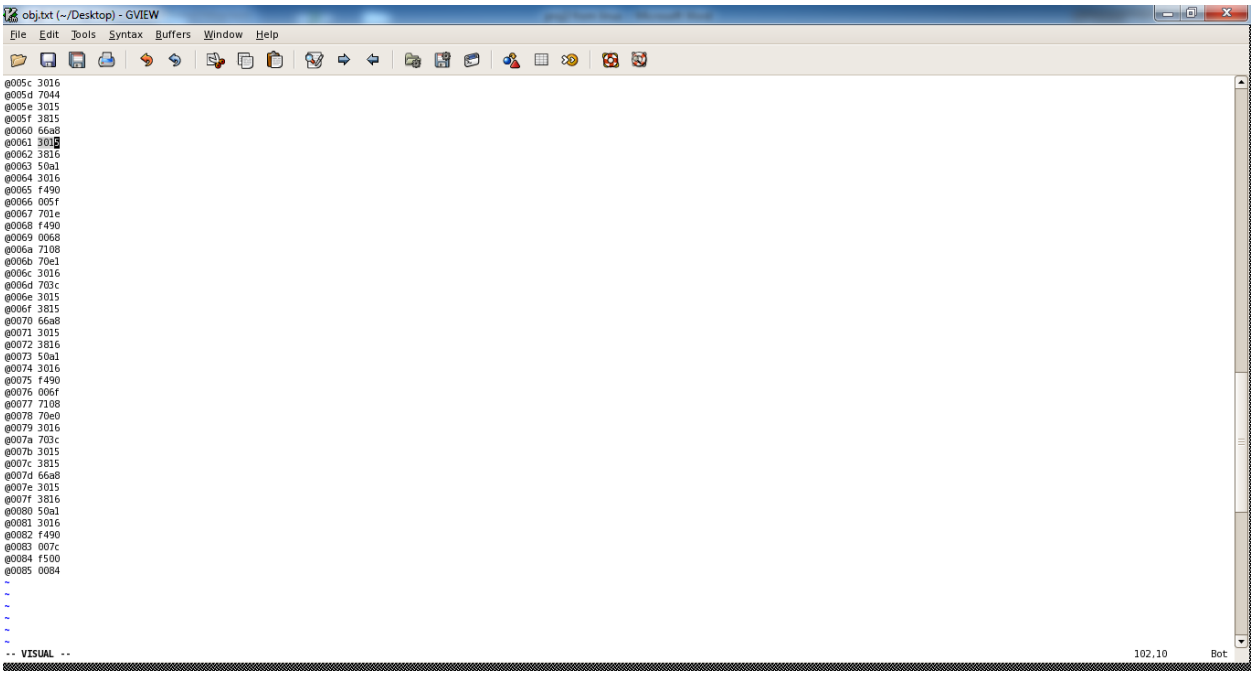
```
*****************************************
          A = 0x0009 (0000011) (    9) A
        AR0 = 0x0000 (0000000) (    0) A -- predefined symbol
        AR1 = 0x0001 (0000001) (    1) A -- predefined symbol
        PA0 = 0x0000 (0000000) (    0) A -- predefined symbol
        PA1 = 0x0001 (0000001) (    1) A -- predefined symbol
        PA2 = 0x0002 (0000002) (    2) A -- predefined symbol
        PA3 = 0x0003 (0000003) (    3) A -- predefined symbol
        PA4 = 0x0004 (0000004) (    4) A -- predefined symbol
        PA5 = 0x0005 (0000005) (    5) A -- predefined symbol
        PA6 = 0x0006 (0000006) (    6) A -- predefined symbol
        PA7 = 0x0007 (0000007) (    7) A -- predefined symbol
          X = 0x0006 (0000006) (    6) A
          Y = 0x0007 (0000007) (    7) A
          Z = 0x0008 (0000010) (    8) A
        ar0 = 0x0000 (0000000) (    0) A -- predefined symbol
        ar1 = 0x0001 (0000001) (    1) A -- predefined symbol
  base_page0 = 0x0000 (0000000) (    0) A
  base_page1 = 0x0080 (0000200) (  128) A
        bss = 0x0003 (0000003) (    3) A
        csa = 0x0004 (0000004) (    4) A
       loop = 0x0021 (0000041) (   33) R
        pa0 = 0x0000 (0000000) (    0) A -- predefined symbol
        pa1 = 0x0001 (0000001) (    1) A -- predefined symbol
        pa2 = 0x0002 (0000002) (    2) A -- predefined symbol
        pa3 = 0x0003 (0000003) (    3) A -- predefined symbol
        pa4 = 0x0004 (0000004) (    4) A -- predefined symbol
        pa5 = 0x0005 (0000005) (    5) A -- predefined symbol
        pa6 = 0x0006 (0000006) (    6) A -- predefined symbol
        pa7 = 0x0007 (0000007) (    7) A -- predefined symbol
      page0 = 0x0000 (0000000) (    0) A
 poundsign1 = 0x004d (0000115) (   77) R
 poundsign2 = 0x005a (0000132) (   90) R
poundsign_1 = 0x0052 (0000122) (   82) R
poundsign_2 = 0x005f (0000137) (   95) R
       pow1 = 0x0044 (0000104) (   68) A
       pow2 = 0x0045 (0000105) (   69) A
       pow3 = 0x0046 (0000106) (   70) A
       pow4 = 0x0047 (0000107) (   71) A
       pow5 = 0x0048 (0000110) (   72) A
       pow6 = 0x0049 (0000111) (   73) A
       pow7 = 0x004a (0000112) (   74) A
       pow8 = 0x004b (0000113) (   75) A
  powquiet1 = 0x003c (0000074) (   60) A
  powquiet2 = 0x003d (0000075) (   61) A
  powquiet3 = 0x003e (0000076) (   62) A
  powquiet4 = 0x003f (0000077) (   63) A
  powquiet5 = 0x0040 (0000100) (   64) A
  powquiet6 = 0x0041 (0000101) (   65) A
```

```
powquiet7 = 0x0042 (0000102) (   66) A
powquiet8 = 0x0043 (0000103) (   67) A
      ptr = 0x0015 (0000025) (   21) A
    quiet = 0x0030 (0000060) (   48) R
   quiet2 = 0x003d (0000075) (   61) R
   quiet3 = 0x006a (0000152) (  106) R
   quiet4 = 0x0077 (0000167) (  119) R
  quiet_2 = 0x0042 (0000102) (   66) R
  quiet_3 = 0x006f (0000157) (  111) R
  quiet_4 = 0x007c (0000174) (  124) R
  quiet_l = 0x0035 (0000065) (   53) R
rcc_1209Hz = 0x0064 (0000144) (  100) A
rcc_1336Hz = 0x0065 (0000145) (  101) A
rcc_1477Hz = 0x0066 (0000146) (  102) A
rcc_1633Hz = 0x0067 (0000147) (  103) A
 rcc_697Hz = 0x0060 (0000140) (   96) A
 rcc_770Hz = 0x0061 (0000141) (   97) A
 rcc_852Hz = 0x0062 (0000142) (   98) A
 rcc_941Hz = 0x0063 (0000143) (   99) A
  rcc_kick = 0x0068 (0000150) (  104) A
   rcc_len = 0x0008 (0000010) (    8) A
   rcc_ptr = 0x0016 (0000026) (   22) A
  spectrum = 0x0030 (0000060) (   48) R
      stop = 0x0084 (0000204) (  132) R
       two = 0x0002 (0000002) (    2) A
     wait0 = 0x004b (0000113) (   75) R
     wait1 = 0x0068 (0000150) (  104) R
  waitTime = 0x001e (0000036) (   30) A
```

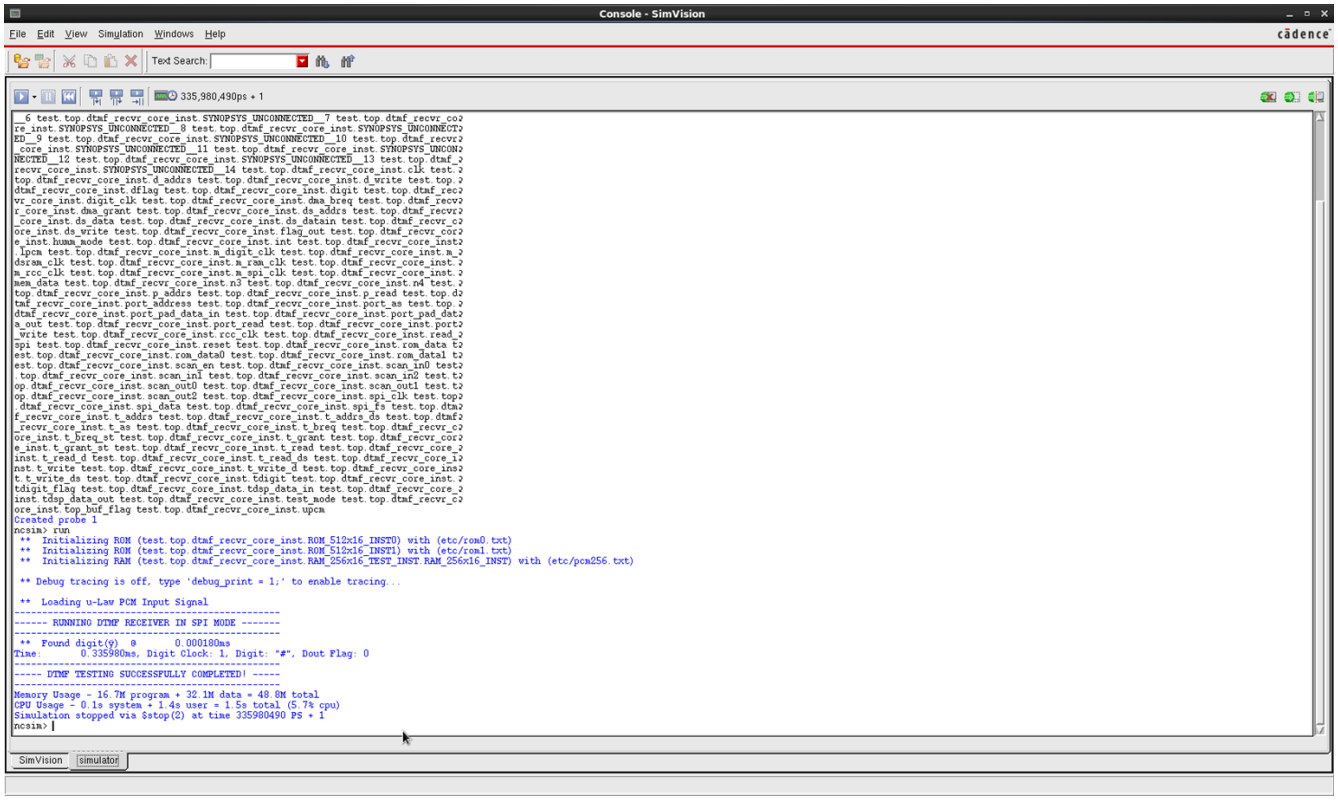
E. Assembly Language Test Program Object File




obj.txt (~/Desktop) - GVIEW
File Edit Tools Syntax Buffers Window Help
@0032 301c
@0033 703c
@0034 3015
@0035 3815
@0036 66a8
@0037 3015
@0038 3816
@0039 50a1
@003a 3016
@003b f490
@003c 0035
@003d 7108
@003e 70e0
@003f 3016
@0040 703c
@0041 3015
@0042 3815
@0043 66a8
@0044 3015
@0045 3816
@0046 50a1
@0047 3016
@0048 f490
@0049 0042
@004a 701e
@004b f490
@004c 004b
@004d 7108
@004e 70e0
@004f 3016
@0050 7044
@0051 3015
@0052 3815
@0053 66a8
@0054 3015
@0055 3816
@0056 50a1
@0057 3016
@0058 f490
@0059 0052
@005a 7108
@005b 70e0
@005c 3016
@005d 7044
@005e 3015
@005f 3815
@0060 66a8
@0061 3015
55,10 60%

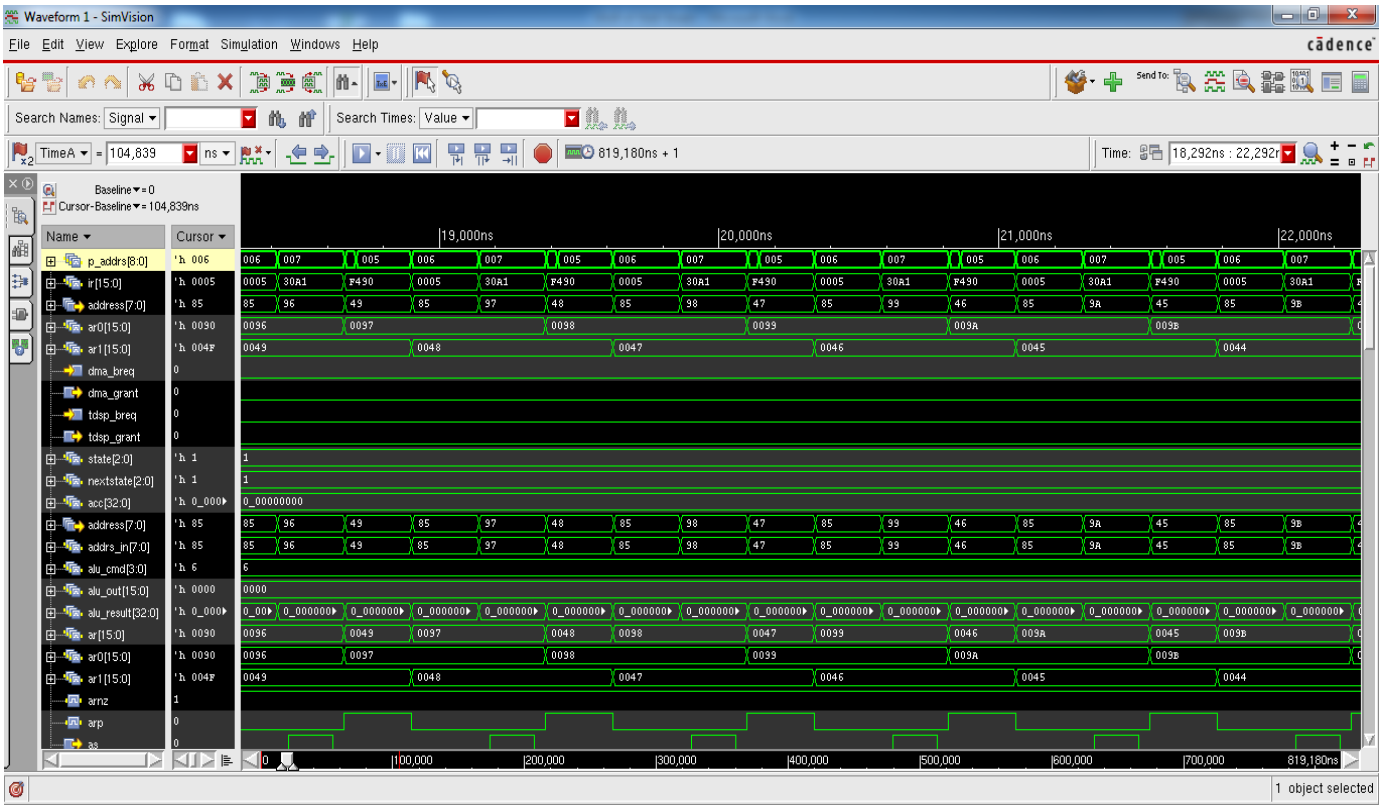

obj.txt (~/Desktop) - GVIEW
File Edit Tools Syntax Buffers Window Help
@005c 3016
@005d 7044
@005e 3015
@005f 3815
@0060 66a8
@0061 3015
@0062 3816
@0063 50a1
@0064 3016
@0065 f490
@0066 005f
@0067 701e
@0068 f490
@0069 0068
@006a 7108
@006b 70e1
@006c 3016
@006d 703c
@006e 3015
@006f 3815
@0070 66a8
@0071 3015
@0072 3816
@0073 50a1
@0074 3016
@0075 f490
@0076 006f
@0077 7108
@0078 70e0
@0079 3016
@007a 703c
@007b 3015
@007c 3815
@007d 66a8
@007e 3015
@007f 3816
@0080 50a1
@0081 3016
@0082 f490
@0083 007c
@0084 f500
@0085 0084
-- VISUAL --
102,10 Bot

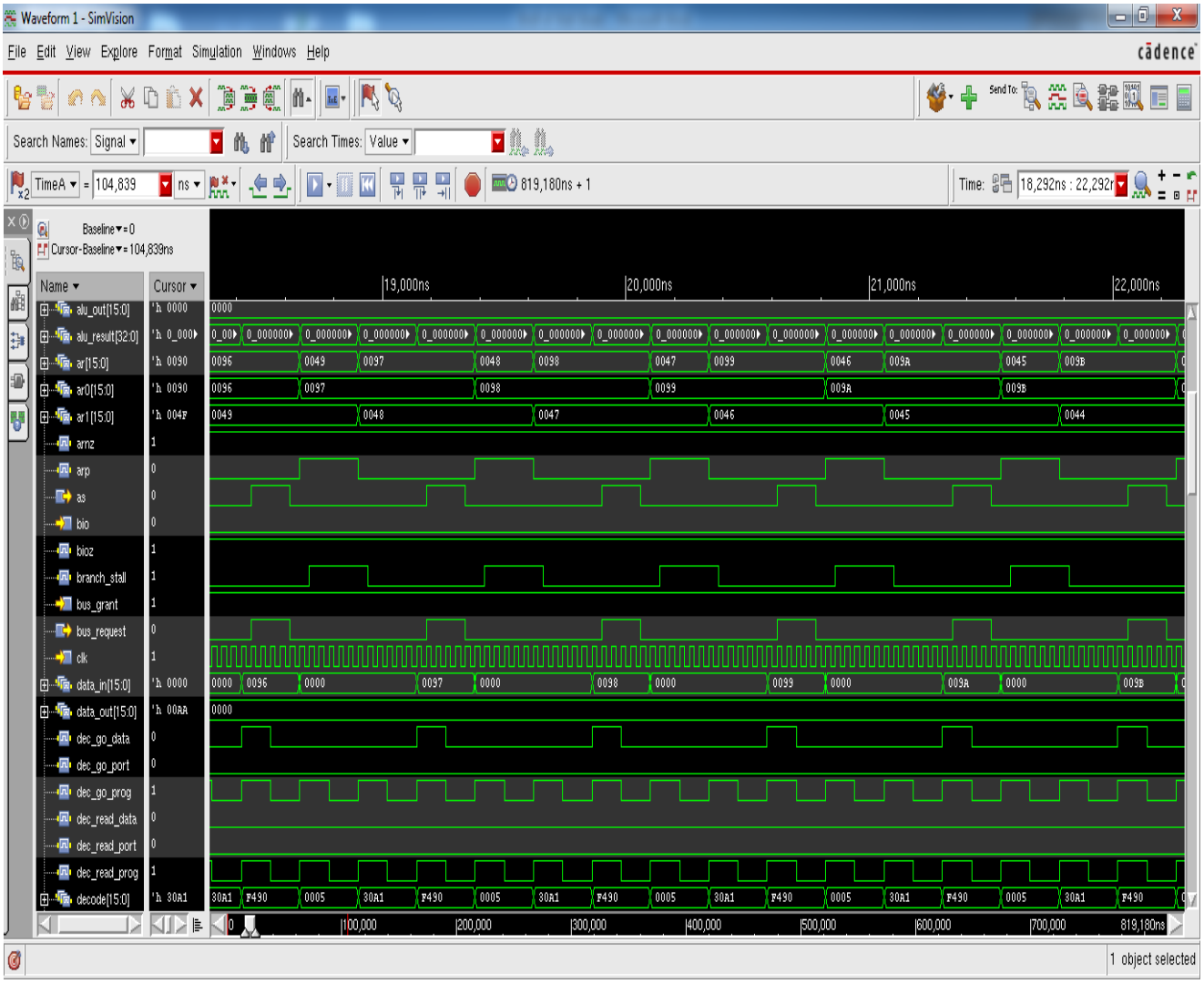
F Simulator Log files



```
zxw6805@vlsi-legacy:~/ee620/proj3/tdsp/dtmf+humm
File  Edit  View  Search  Terminal  Help
============================================================================


Information: There are 614 potential problems in your design. Please run 'check_design' for more information. (LINT-99)


  Beginning Pass 1 Mapping  (Incremental)
  -----------------------

  Updating timing information
Information: Updating design information... (UID-85)

  Beginning Mapping Optimizations  (High effort)  (Incremental)
  ------------------------------

  Beginning Incremental Implementation Selection
  ----------------------------------------------

  Beginning Delay Optimization Phase
  ----------------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:03  336429.5      0.00       0.0       1.3
    0:00:04  336429.5      0.00       0.0       1.3
    0:00:10  336409.5      0.00       0.0       1.5


  Beginning Design Rule Fixing  (max_transition)  (max_capacitance)
  ----------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:10  336409.5      0.00       0.0       1.5
    0:00:10  336412.9      0.00       0.0       0.0


  Beginning Critical Range Optimization
  -------------------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:10  336412.9      0.00       0.0       0.0


  Beginning Critical Range Optimization
  -------------------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:10  336412.9      0.00       0.0       0.0
Loading db file '/classes/ee620/maieee/lib/tsmc-0.18/synopsys/typical.db'
Loading db file '/classes/ee620/maieee/lib/tsmc-0.18/model/ram_128x16_typical.db'
Loading db file '/classes/ee620/maieee/lib/tsmc-0.18/model/ram_16x16_typical.db'
Loading db file '/classes/ee620/maieee/lib/tsmc-0.18/model/ram_256x16_typical.db'
Loading db file '/classes/ee620/maieee/lib/tsmc-0.18/model/ram_512x16_typical.db'
Loading db file '/classes/ee620/maieee/lib/tsmc-0.18/model/rom_512x16_typical.db'
```

G screen shots of gate level simulation waveforms showing program execution





5 summary and conclusions

Summary

Overall, the finite state machine Verilog implementation is not difficult since it is simple and straight forward implementation except a few minor issues with reset and always@ code. The test bench takes time to implement since I thought about the output TDSP_GRANT and DMA_GRANT should be checked for being 1 or 0 when they are supposed to be instead of just checking them being 1 within the correct clock period cycle. One combinational and two sequential registers are created in FSM Verilog program.

The Verilog file is copied into assembly file location for simulating TDSP assembly language test program. The assembly TDSP was created and used twenty years ago. Thus, it is hard to find the example code online. DTMF_TDSP.asm is used as reference for coding. After figuring out how to add and multiply number using registers and accumulators, how to make a loop using auxiliary register and how to make spectrum with minimum time delay, the assembly program is finished with # spectrum being displayed in simulation console.

Conclusion

The project overall is about understanding rather than writing codes since most of the reference materials are available for use. Finite State Machine is commonly used in digital world. The Verilog code implementation for finite state machine is easy to find online. The test bench logic is used for making sure that TDSP_GRANT and DMA_GRANT occurring within the 1 correct clock cycle instead of them being 1 wrongly 1000 clock cycles after. The assembly language test program coding is not a lot with DTMF_TDSP assembly language reference being provided. The understanding of dual tone multiple frequency register implementation and timing delay is necessary to finish the assembly language test program.