EE620 — DESIGN OF DIGITAL SYSTEMS

From: Dinesh Anand Bashkaran

UID: 57000943

Instructor**r:** Dr. Mark.A.indovina

PROJECT 3 REPORT- DTMF

SECTION 1-INTRODUCTION:

The project is all about working in the bottom level of DTMF receiver. Dual tone multi frequency (DTMF) is a type of in-band signaling. In-band signaling is one of the methods used in transmitting information in network entities. DTMF is generally used in telephones to detect the key that has been pressed. This project gives clear idea about how the received signal is being processed in TDSP to find what key has been pressed. In this project, RTL (register transfer level) module is designed for memory access bus arbiter and this module is used in the DTMF model.

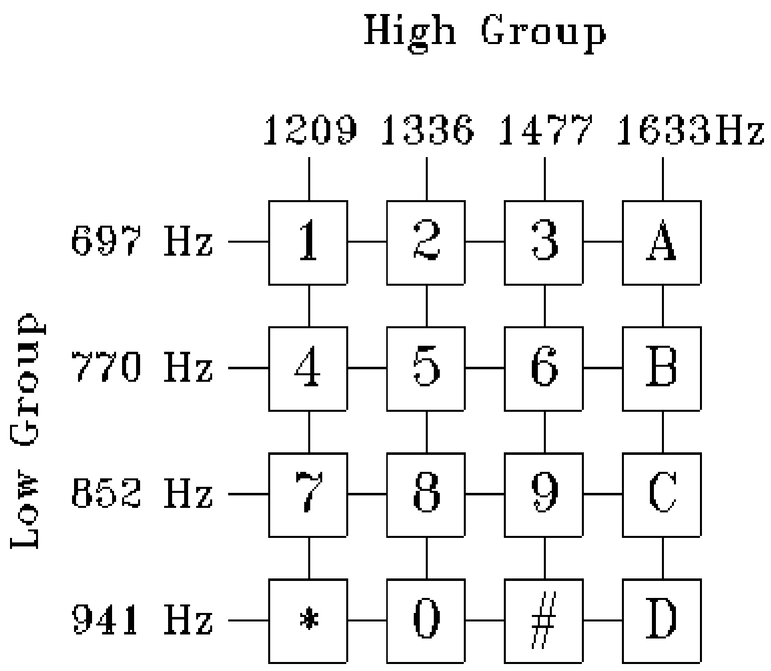


Fig: Key pad frequency view

Every key is pre-defined with a set of low-frequency and high-frequency. Every time a key is pressed it generated a spectrum which is fed as input to the DTMF receiver. The DTMF receiver figures what frequency the spectrum has and determines what key has been pressed. To determine the frequency DTMF uses a modified version of discrete fourier algorithm called as goertzel algorithm. In this project,

- A module for memory access bus arbiter is modeled.
- Test vectors are created to test the memory access bus arbiter for few definitions.
- The modeled ARB is then placed into the DTMF block.
- Assembly language is used to execute few statements.
- SPI mode test bench is modeled that stops the simulation when "#" is detected.
- Logic synthesis, timing analysis, gate level verification is done.

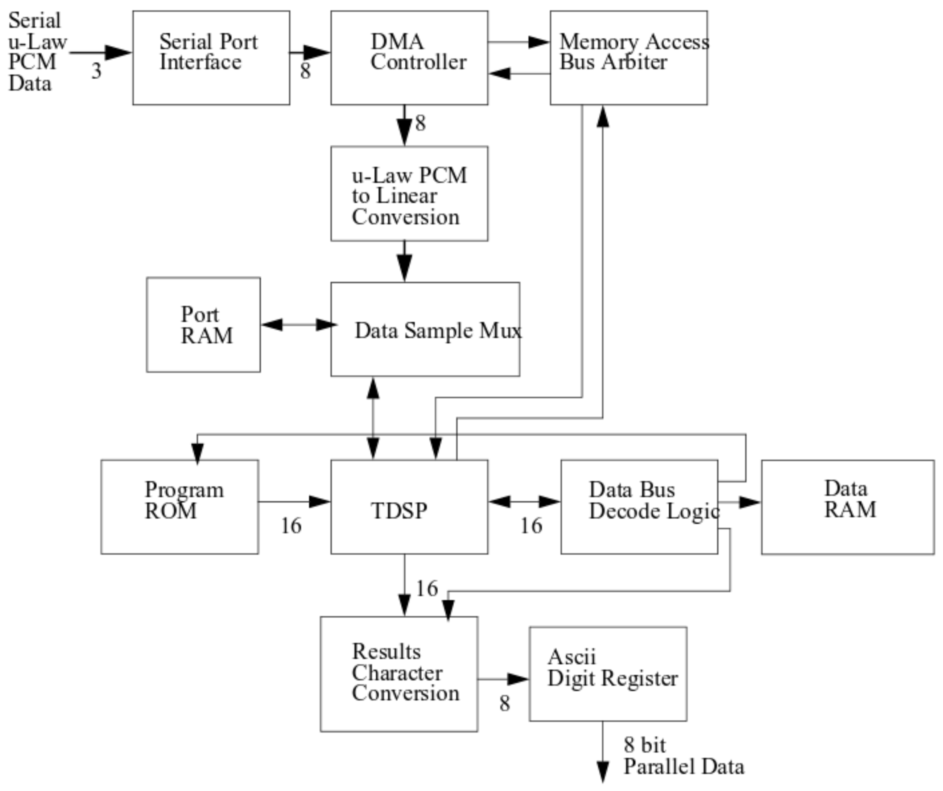


Fig: Model of DTMF block

Serial port interface: This unit accepts the PCM data, serialized first LSB first and then the data is reformed from data to byte orientation. As soon as the SPI receives the data it signals the DMA controller that new data has to be moved to the data sample memory.

Data memory access: The data memory access controller is intermediate unit that binds the data byte between the SPI and the data sample memory. DMA basically sends requests through bus arbiter to the data sample memory. Once, the access is granted the data byte is fed into the data sample RAM.

Memory Access Arbiter (ARB):

Memory access arbiter works on a simple protocol which is REQUEST, GRANT. The priority is always given to TDSP when both DMA and TDSP requests at same time. ARB acts as a coordinator that coordinates DMA and TDSP access to the data sample memory.

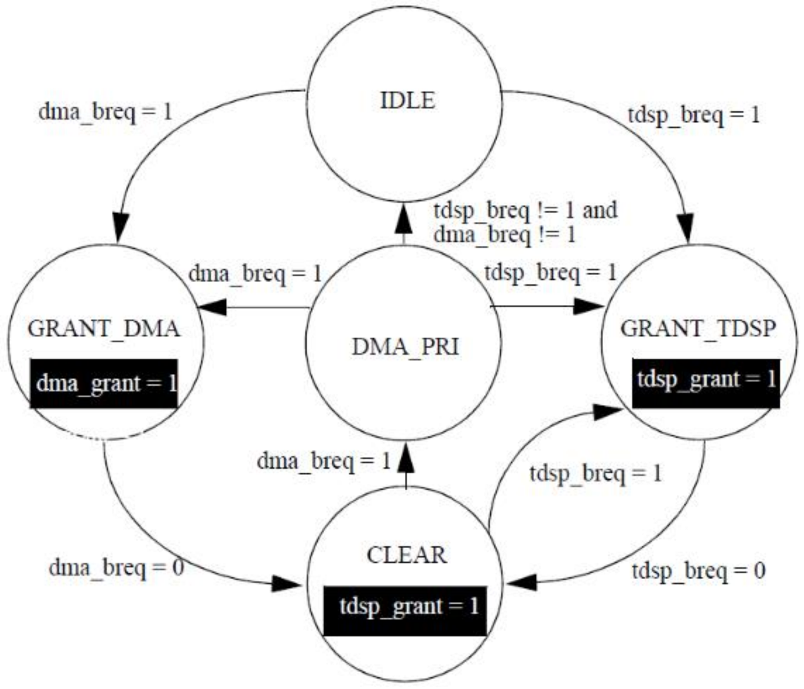


Fig: State diagram with respect to ARB

In this project, RTL level module for the ARB is coded using Verilog on RC Verilog. There are 5 states and each state has its own preference which has to be considered while the module is coded.

TDSP: Tiny digital signal processor with the instruction set of TMS320 family is used here.

Result character conversion:

As soon as the TDSP detects the signal spectrum the results are written into the RCC. The resulting spectrum in the RCC is analyzed for the DTMF digit content and once it figures content it is resolved into ASCII character representation and written into results circular buffer. So, once a sequence is processed, the ASCII character is moved to the ASCII digit register.

Working: To detect the tones, the DTMF uses modified form of discrete fourier transform which is called as Goertzel algorithm. This way is very efficient in calculating partial frequency spectrum and of course is what we are looking for to find the DTMF center frequencies. As the nature of the Goertzel algorithm is recursive it will run entirely in firmware on tiny digital signal processor.

Algorithm:

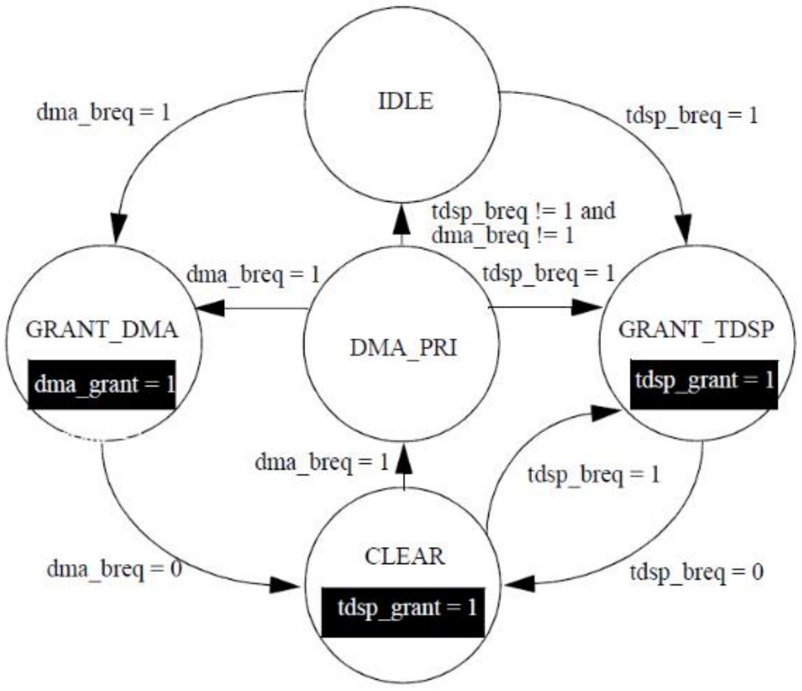
Goertzel algorithm works on second order impulse response. The last iteration of the algorithm is (N-1) and this is where both parts of the graph will be calculated. A complex multiplication is only required once per iteration. As soon as the frequency is calculated it is checked if any DTMF digit is found.

A few set of conditions has to be maintained:

- A spectrum has to pass through the processor for a minimum period of 45ms.
- A minimum time has to be given to the processor to execute every single instruction.

SECTION 2:

a.**MEMORY ACCESS BUS ARBITER (ARB) MODULE:**



As discussed, ARB acts as a unit that works on REQUEST and GRANT scheme. With respect to project the ARB design is as follows,

- IDLE State (1): This state has priority towards GRANT_TDSP when dma/tdsp requests same time. But when dma request and tdsp doesn't then state moves to DMA_GRANT. When tdsp request and dma doesn't then state moves to TDSP_GRANT. When both the request are low the state stays in IDLE.
- GRANT_DMA (2): This state has value of DMA_GRANT =1. When dma request is low the state move to CLEAR.
- GRANT_TDSP (0): TDSP_GRANT =1 is the value of this state. When the tdsp request is zero the state moves to CLEAR.
- CLEAR (3): The value of this state is TDSP_GRANT =1. This state has the priority to the TDSP, which means when both dma/tdsp requests high the state moves to GRANT_TDSP. When dma requests then state moves to DMA_PRI. When tdsp request the state moves to GRANT_TDSP.

- DMA_PRI (7): This state has the priority towards the DMA request. When both dma/tdsp sends request same time, the state moves to GRANT_DMA. When tdsp request is high then state moves to GRANT_TDSP. When both dma/tdsp request is low then the state moves to IDLE.

Considering these conditions, the ARB is coded in Verilog.

b. **arb.v RTL source code**

```
/*
 *
 * Author:  Mark A. Indovina
 *         Rochester, NY, USA
 *
 */

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

//
// explicit state machine states
//
`include "./include/arb.h"

reg[2:0] pstate;
reg dma_grant,tdsp_grant;

always@(posedge reset or posedge clk)

begin

            if(reset)
                        begin
                        dma_grant <= 0;
                        tdsp_grant <= 0;
                        pstate <= `ARB_IDLE;
                        end
            else
                        begin


case (pstate)

`ARB_IDLE: begin
            dma_grant<=0;
                        tdsp_grant<=0;
             if(tdsp_breq ==1 && dma_breq ==1)
                                    begin
                                    pstate <= `ARB_GRANT_TDSP;
                                    tdsp_grant <=1 ;
                                    dma_grant <= 0;
                                    end
                  else if(dma_breq ==1 && tdsp_breq == 0)
                                    begin
                                    pstate <= `ARB_GRANT_DMA;
                                    dma_grant <= 1 ;
                                    tdsp_grant <= 0;
                                    end
                        else if (tdsp_breq == 1 && dma_breq ==0)
                                    begin
                                    pstate <= `ARB_GRANT_TDSP;
                                    tdsp_grant <=1;
                                    dma_grant <= 0;
                                    end
             else
                        pstate <= `ARB_IDLE;
            end

`ARB_GRANT_DMA: begin
                    dma_grant<= 1;
                    tdsp_grant <= 0;
            if(dma_breq == 0)
                    begin
                            pstate <= `ARB_CLEAR;
                            tdsp_grant <= 1;
                            dma_grant <= 0 ;
                    end
            end

`ARB_GRANT_TDSP: begin
            dma_grant<=0;
            tdsp_grant<=1;
            if(tdsp_breq == 0)
                            begin
                            pstate <= `ARB_CLEAR;
                        tdsp_grant <= 1;
                            dma_grant <= 0 ;
                end
                    end

`ARB_CLEAR: begin

                    dma_grant<=0;
                    tdsp_grant<=1;
            if(tdsp_breq ==1)
                            begin
                            pstate <= `ARB_GRANT_TDSP;
                            tdsp_grant <= 1;
                            dma_grant <= 0 ;
                            end
            /*      else if(tdsp_breq ==1 && dma_breq ==0)
                    begin
                    pstate <= `ARB_GRANT_TDSP;
                    tdsp_grant <=1;
                    dma_grant <= 0 ;
            */ //end
            else if(dma_breq ==1)
                            begin
                            pstate <= `ARB_DMA_PRI;
                            dma_grant <=0;
                            tdsp_grant <= 0 ;
                            end
                    else
            if(tdsp_breq ==0 && dma_breq ==0)
                    pstate<=`ARB_CLEAR;
                end

`ARB_DMA_PRI :begin
                        dma_grant <=0;
                                    tdsp_grant <= 0 ;
            if(dma_breq==1 && tdsp_breq==1)
                                    begin
                                    pstate <= `ARB_GRANT_DMA;
                                    dma_grant <=1;
                                    tdsp_grant <= 0;
                                    end
                        else if ( dma_breq == 1 && tdsp_breq == 0)
                                    begin
                                    pstate <= `ARB_GRANT_DMA ;
                                    dma_grant <= 1;
                                    tdsp_grant <= 0 ;
                                    end
            else if(tdsp_breq ==1 && dma_breq ==0)
                                    begin
                                    pstate <= `ARB_GRANT_TDSP;
                                    tdsp_grant <= 1;
                                    dma_grant <= 0;
                                    end
                  else if (dma_breq==0 && tdsp_breq ==0)
                                    begin
                                    pstate <= `ARB_IDLE;
                                    dma_grant <= 0 ;
                                    tdsp_grant <= 0;
                                    end
          end



            default :
                        pstate<= `ARB_IDLE;

 endcase

end
end
 endmodule
```

C. arb_test.v test bench source code:

```
            /*
 *
 * Author:  Mark A. Indovina
 *         Rochester, NY, USA
 *
 */


`timescale 1ns / 1ns

module test;

`define stop \
$finish;

wire dma_grant, tdsp_grant;
wire scan_out0;

reg clk, dma_breq, reset, tdsp_breq, dma_error, tdsp_error;
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,
   dma_cnt,
   tdsp_cnt ,
   a,b;
```

```
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_wait = $random ;
   tdsp_wait = $random ;
   a=0;
   b=0;
   dma_error=1'b0;
   tdsp_error=1'b0;

   @(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
         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

                 @(posedge dma_breq)
                begin
                if(tdsp_grant ==1)
                                begin
                                @(negedge tdsp_grant);
                          // if(dma_breq ==1)
                                        begin
                                        repeat(3)
                                        @(posedge clk);
                                        if(dma_grant ==0)
                                                begin
                                                dma_error<=1'b1;
                                                $display ("simulation time %t", $time);
                                                $display ("GRANT DMA ERROR");
                                                `stop
                                                //`endif
                                                end

                                        end
                                end


          /*  else if(tdsp_grant==0)
                                begin

                                repeat (6)
                                @(posedge clk)
                                if(dma_grant ==0)
                                    begin
                          dma_error<=1'b1;
                                    $display ("simulation time %t", $time);
                                    $display ("GRANT DMA ERROR");
                                    `stop
                                    //`endif
                                    end
                      end  */
                  end
```

```
        end

endtask


task tdsp_check;
        begin
         @(posedge tdsp_breq)
        begin

                if(dma_grant == 1)
                        begin
                        @(negedge dma_grant);
                        if(tdsp_breq)
                                begin

                                @(posedge clk);
                                if(tdsp_grant ==0)
                                                begin
                                                tdsp_error<=1'b1;
                                                        $display("TDSP ERROR" );
                                                        $display ("simulation time %t", $time);
                                                        ` stop
                                                        //`endif
                                                //      $finish; */
                                                end

                                end
                        end




        else if(dma_grant == 0)
                        begin
                        repeat(1)
                        @(posedge clk);
                        if(tdsp_grant == 0 )
                                        begin
                            tdsp_error<=1'b1;
                                        $display("TDSP ERROR" );
                                        $display ("simulation time %t", $time);
                                        ` stop
                                        //`endif
                                        end
                        end
```

```
                    end

          end

endtask



endmodule
```

d. RTL level simulation waveform:

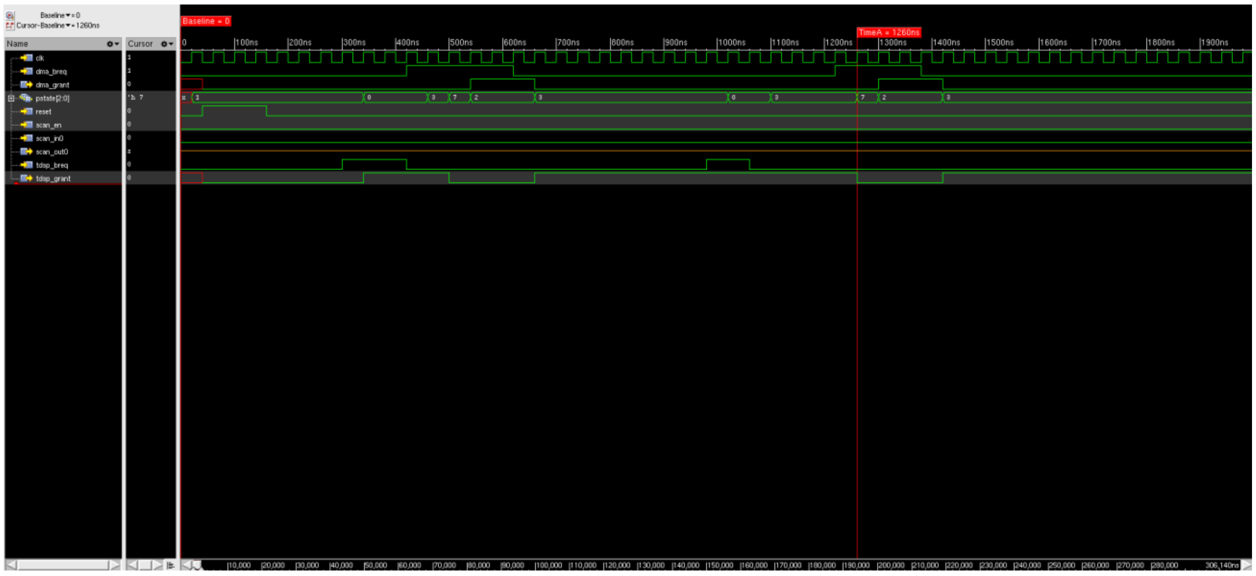
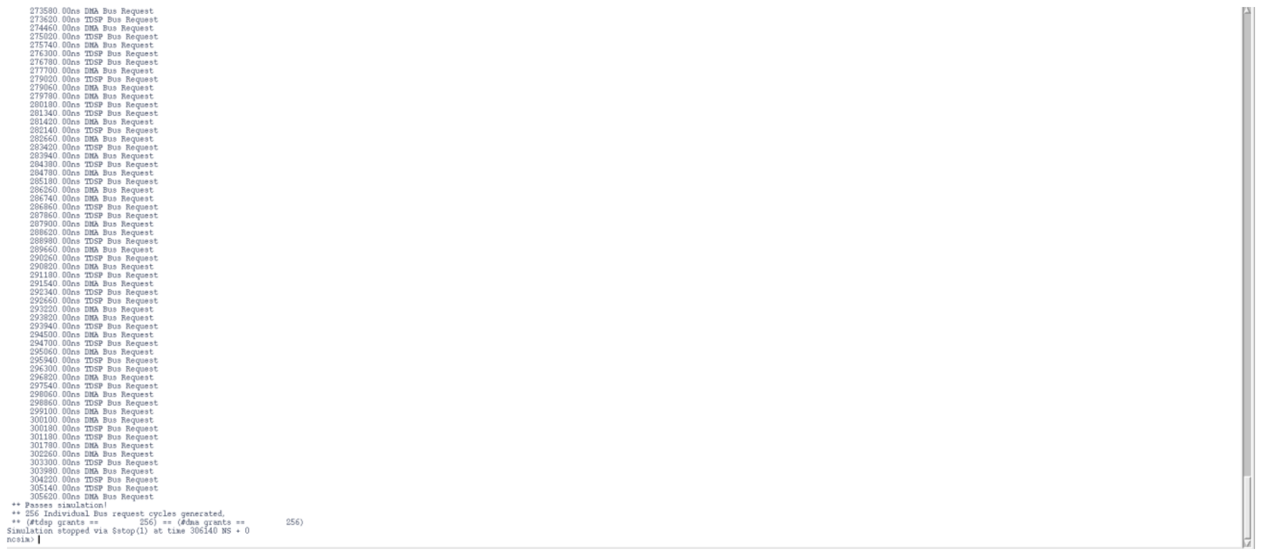


FIG: RTL waveform



```
    273580.00ns DMA Bus Request
    273620.00ns TDSP Bus Request
    274460.00ns DMA Bus Request
    275020.00ns TDSP Bus Request
    275740.00ns DMA Bus Request
    276300.00ns TDSP Bus Request
    276780.00ns TDSP Bus Request
    277700.00ns DMA Bus Request
    279020.00ns TDSP Bus Request
    279060.00ns DMA Bus Request
    279780.00ns DMA Bus Request
    280180.00ns TDSP Bus Request
    281340.00ns TDSP Bus Request
    281420.00ns DMA Bus Request
    282140.00ns TDSP Bus Request
    282660.00ns DMA Bus Request
    283420.00ns TDSP Bus Request
    283940.00ns DMA Bus Request
    284380.00ns TDSP Bus Request
    284780.00ns DMA Bus Request
    285180.00ns TDSP Bus Request
    286260.00ns DMA Bus Request
    286740.00ns DMA Bus Request
    286860.00ns TDSP Bus Request
    287860.00ns TDSP Bus Request
    287900.00ns DMA Bus Request
    288620.00ns DMA Bus Request
    288980.00ns TDSP Bus Request
    289660.00ns DMA Bus Request
    290260.00ns TDSP Bus Request
    290820.00ns DMA Bus Request
    291180.00ns TDSP Bus Request
    291540.00ns DMA Bus Request
    292340.00ns TDSP Bus Request
    292660.00ns TDSP Bus Request
    293220.00ns DMA Bus Request
    293820.00ns DMA Bus Request
    293940.00ns TDSP Bus Request
    294500.00ns DMA Bus Request
    294700.00ns TDSP Bus Request
    295060.00ns DMA Bus Request
    295940.00ns TDSP Bus Request
    296300.00ns TDSP Bus Request
    296820.00ns DMA Bus Request
    297540.00ns TDSP Bus Request
    298060.00ns DMA Bus Request
    298860.00ns TDSP Bus Request
    299100.00ns DMA Bus Request
    300100.00ns DMA Bus Request
    300180.00ns TDSP Bus Request
    301180.00ns TDSP Bus Request
    301780.00ns DMA Bus Request
    302260.00ns DMA Bus Request
    303300.00ns TDSP Bus Request
    303980.00ns DMA Bus Request
    304220.00ns TDSP Bus Request
    305140.00ns TDSP Bus Request
    305620.00ns DMA Bus Request
 ** Passes simulation!
 ** 256 Individual Bus request cycles generated.
 ** (#tdsp grants ==          256) == (#dma grants ==          256)
Simulation stopped via $stop(1) at time 306140 NS + 0
ncsim>
```

FIG: Console Window

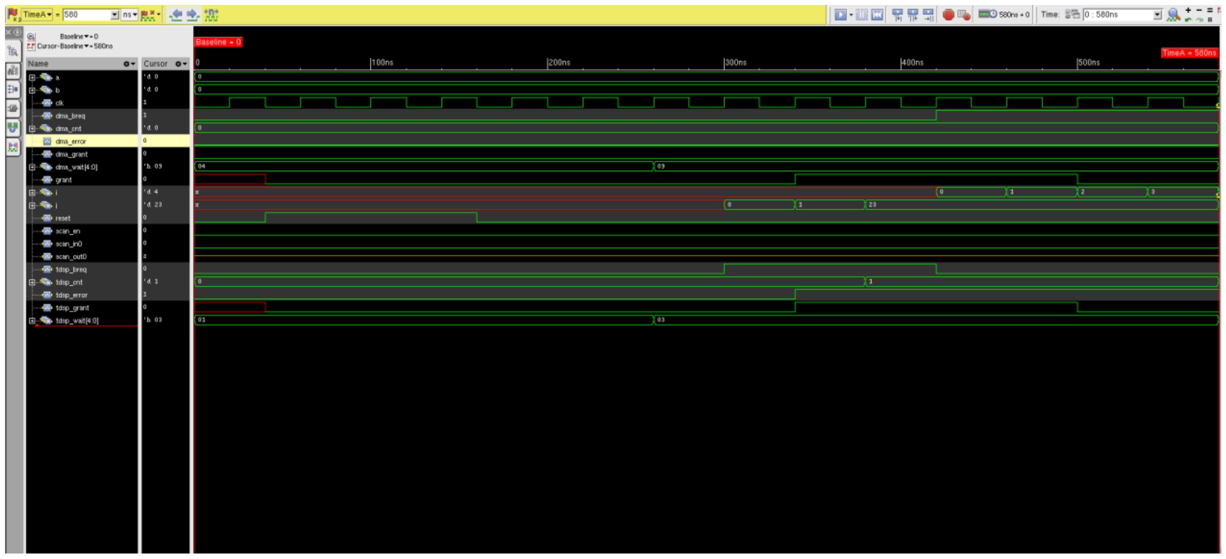
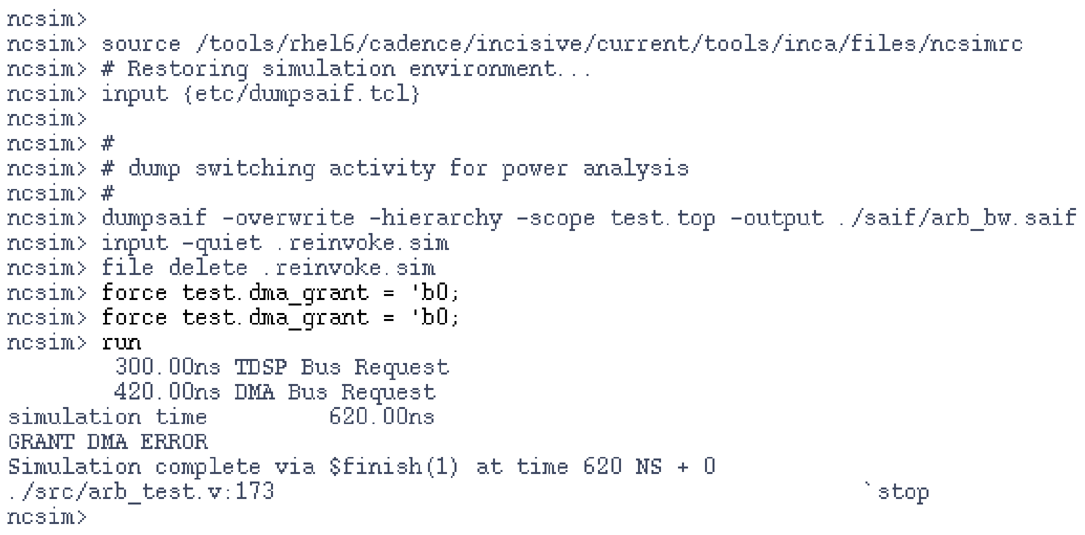


Fig: simulation stopped after forcing dma_grant to 0



```
ncsim>
ncsim> source /tools/rhel6/cadence/incisive/current/tools/inca/files/ncsimrc
ncsim> # Restoring simulation environment...
ncsim> input {etc/dumpsaif.tcl}
ncsim>
ncsim> #
ncsim> # dump switching activity for power analysis
ncsim> #
ncsim> dumpsaif -overwrite -hierarchy -scope test.top -output ./saif/arb_bw.saif
ncsim> input -quiet .reinvoke.sim
ncsim> file delete .reinvoke.sim
ncsim> force test.dma_grant = 'b0;
ncsim> force test.dma_grant = 'b0;
ncsim> run
        300.00ns TDSP Bus Request
        420.00ns DMA Bus Request
simulation time         620.00ns
GRANT DMA ERROR
Simulation complete via $finish(1) at time 620 NS + 0
./src/arb_test.v:173                                         `stop
ncsim>
```

Fig: DMA Error report

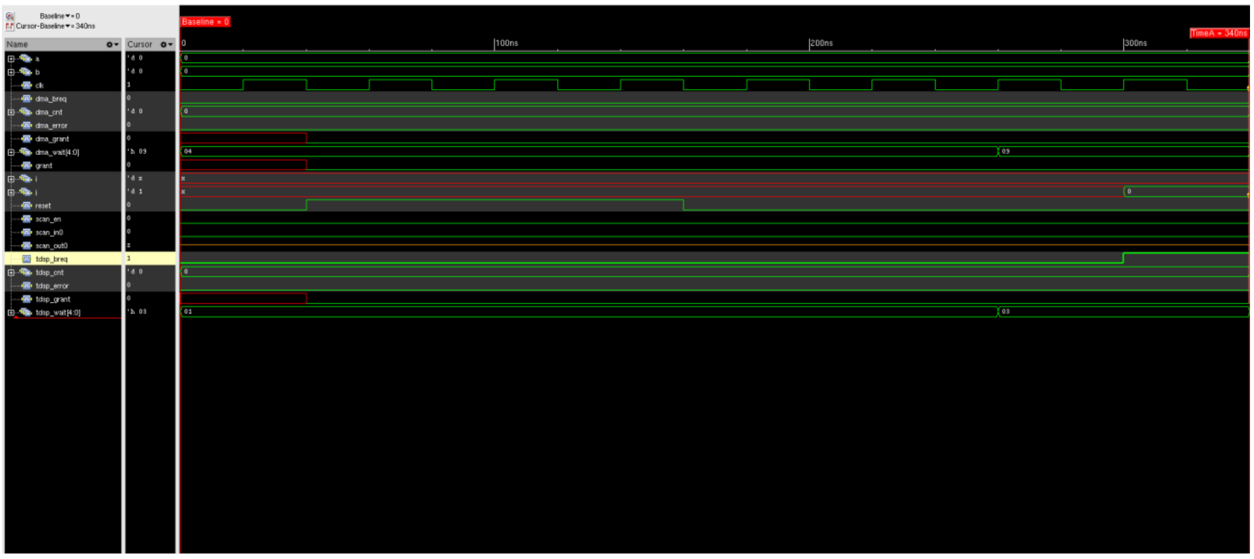
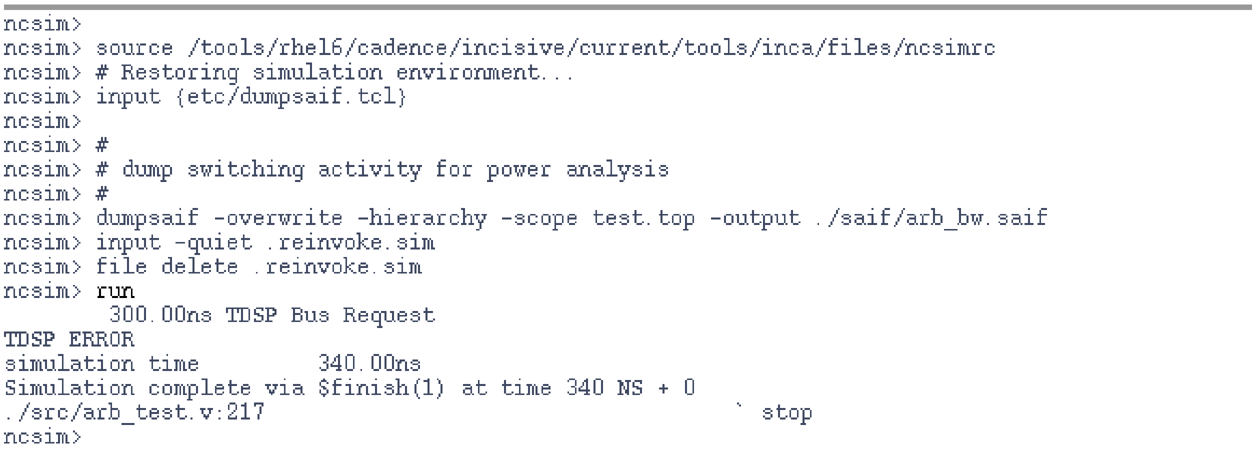


Fig: Simulation stopped after the Tdsp_grant is made 0



```
ncsim>
ncsim> source /tools/rhel6/cadence/incisive/current/tools/inca/files/ncsimrc
ncsim> # Restoring simulation environment...
ncsim> input {etc/dumpsaif.tcl}
ncsim>
ncsim> #
ncsim> # dump switching activity for power analysis
ncsim> #
ncsim> dumpsaif -overwrite -hierarchy -scope test.top -output ./saif/arb_bw.saif
ncsim> input -quiet .reinvoke.sim
ncsim> file delete .reinvoke.sim
ncsim> run
        300.00ns TDSP Bus Request
TDSP ERROR
simulation time         340.00ns
Simulation complete via $finish(1) at time 340 NS + 0
./src/arb_test.v:217                                    ` stop
ncsim>
```

Fig: TDSP Error Report l

e. Netlist level simulation waveform:

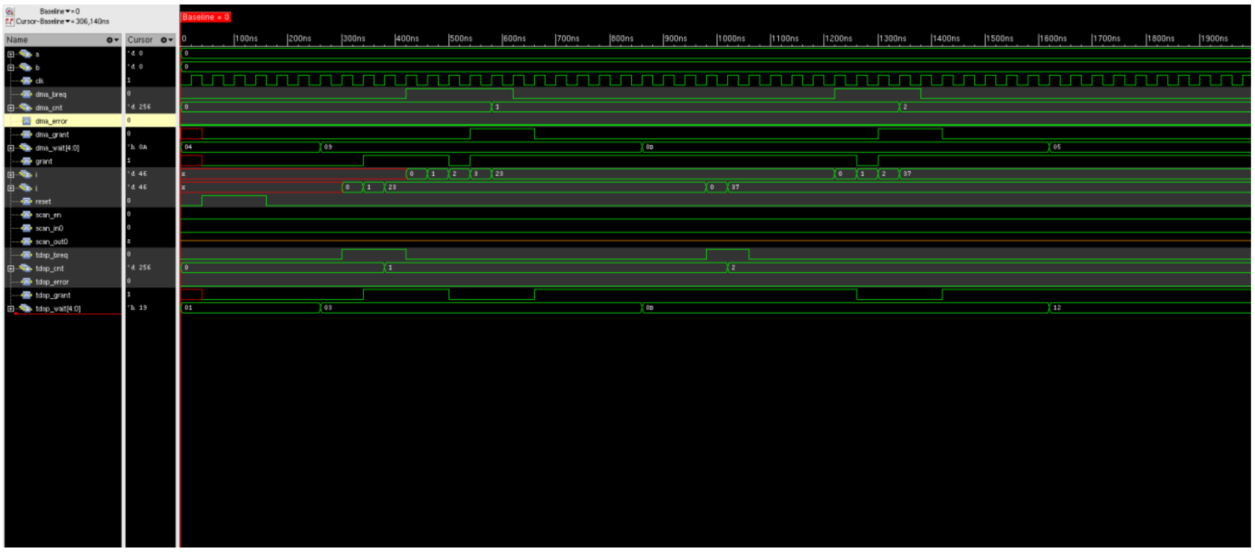
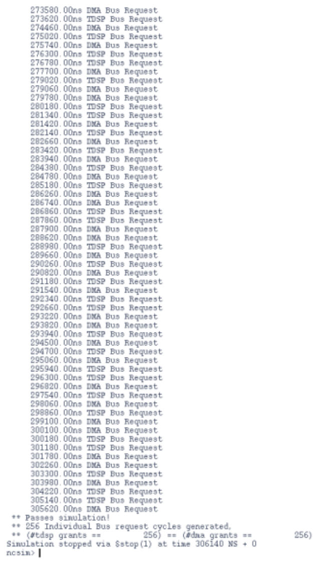


Fig: Netlist level simulation waveform

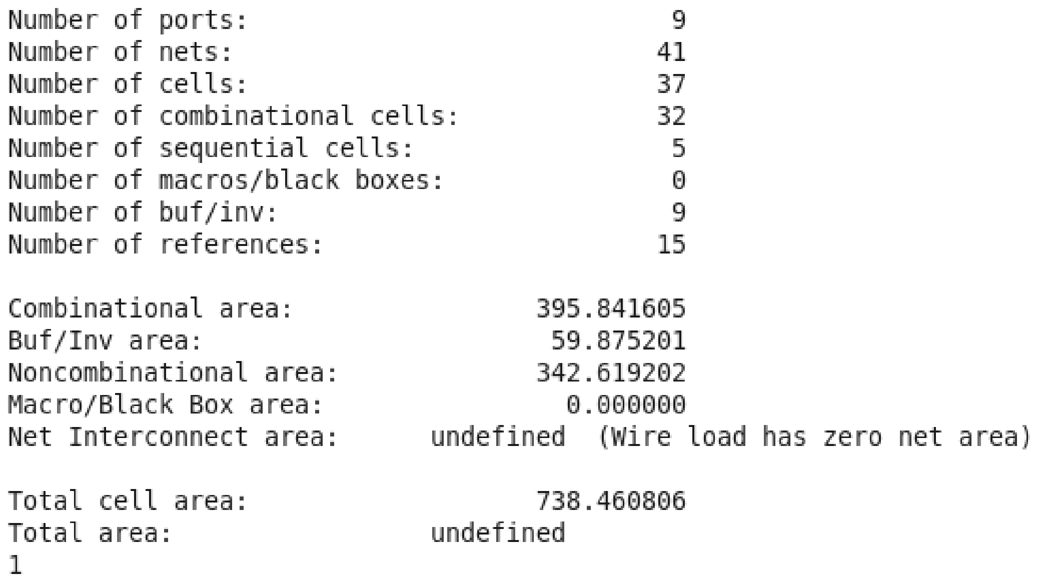


```
    273580.00ns DMA Bus Request
    273620.00ns TDSP Bus Request
    274460.00ns DMA Bus Request
    275020.00ns TDSP Bus Request
    275740.00ns DMA Bus Request
    276300.00ns TDSP Bus Request
    276780.00ns TDSP Bus Request
    277700.00ns DMA Bus Request
    279020.00ns TDSP Bus Request
    279060.00ns DMA Bus Request
    279780.00ns DMA Bus Request
    280180.00ns TDSP Bus Request
    281340.00ns TDSP Bus Request
    281420.00ns DMA Bus Request
    282140.00ns TDSP Bus Request
    282660.00ns DMA Bus Request
    283420.00ns TDSP Bus Request
    283940.00ns DMA Bus Request
    284380.00ns TDSP Bus Request
    284780.00ns DMA Bus Request
    285180.00ns TDSP Bus Request
    286260.00ns DMA Bus Request
    286740.00ns DMA Bus Request
    286860.00ns TDSP Bus Request
    287860.00ns TDSP Bus Request
    287900.00ns DMA Bus Request
    288620.00ns DMA Bus Request
    288980.00ns TDSP Bus Request
    289660.00ns DMA Bus Request
    290260.00ns TDSP Bus Request
    290820.00ns DMA Bus Request
    291180.00ns TDSP Bus Request
    291540.00ns DMA Bus Request
    292340.00ns TDSP Bus Request
    292660.00ns TDSP Bus Request
    293220.00ns DMA Bus Request
    293820.00ns DMA Bus Request
    293940.00ns TDSP Bus Request
    294500.00ns DMA Bus Request
    294700.00ns TDSP Bus Request
    295060.00ns DMA Bus Request
    295940.00ns TDSP Bus Request
    296300.00ns TDSP Bus Request
    296820.00ns DMA Bus Request
    297540.00ns TDSP Bus Request
    298060.00ns DMA Bus Request
    298860.00ns TDSP Bus Request
    299100.00ns DMA Bus Request
    300100.00ns DMA Bus Request
    300180.00ns TDSP Bus Request
    301180.00ns TDSP Bus Request
    301780.00ns DMA Bus Request
    302260.00ns DMA Bus Request
    303300.00ns TDSP Bus Request
    303980.00ns DMA Bus Request
    304220.00ns TDSP Bus Request
    305140.00ns TDSP Bus Request
    305620.00ns DMA Bus Request
 ** Passes simulation!
 ** 256 Individual Bus request cycles generated.
 ** (#tdsp grants ==         256) == (#dma grants ==         256)
Simulation stopped via $stop(1) at time 306140 NS + 0
ncsim> 
```

Fig: Console window with respect to Netlist level simulation.

f. Logic Synthesis report:

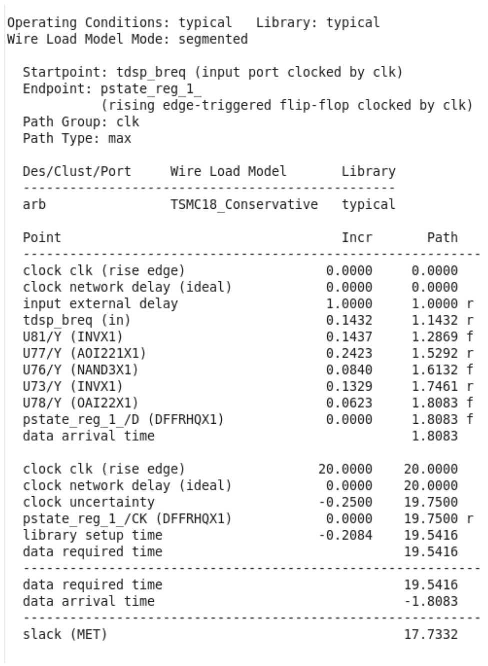
1,2: Total number of cell and total cell area:



```
Number of ports:                                9
Number of nets:                                41
Number of cells:                               37
Number of combinational cells:                 32
Number of sequential cells:                     5
Number of macros/black boxes:                   0
Number of buf/inv:                              9
Number of references:                          15

Combinational area:               395.841605
Buf/Inv area:                      59.875201
Noncombinational area:            342.619202
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  738.460806
Total area:                 undefined
1
```

Fig: Pre-scan for total cell area and total cell number

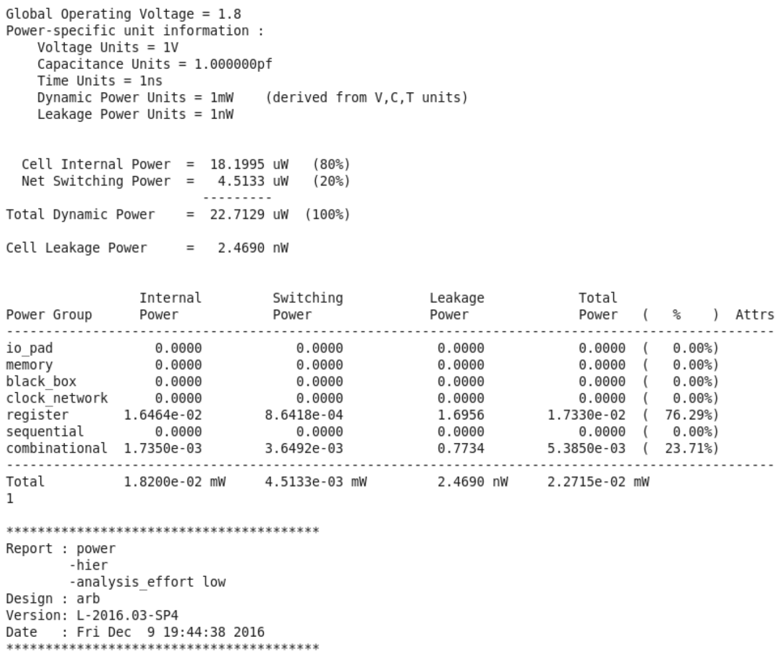
3: worst case timing path for pre scan netlist:



```
Operating Conditions: typical   Library: typical
Wire Load Model Mode: segmented

  Startpoint: tdsp_breq (input port clocked by clk)
  Endpoint: pstate_reg_1_
            (rising edge-triggered flip-flop 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
  input external delay                    1.0000     1.0000 r
  tdsp_breq (in)                          0.1432     1.1432 r
  U81/Y (INVX1)                           0.1437     1.2869 f
  U77/Y (AOI221X1)                        0.2423     1.5292 r
  U76/Y (NAND3X1)                         0.0840     1.6132 f
  U73/Y (INVX1)                           0.1329     1.7461 r
  U78/Y (OAI22X1)                         0.0623     1.8083 f
  pstate_reg_1_/D (DFFRHQX1)              0.0000     1.8083 f
  data arrival time                                  1.8083

  clock clk (rise edge)                  20.0000    20.0000
  clock network delay (ideal)             0.0000    20.0000
  clock uncertainty                      -0.2500    19.7500
  pstate_reg_1_/CK (DFFRHQX1)             0.0000    19.7500 r
  library setup time                     -0.2084    19.5416
  data required time                                19.5416
  -----------------------------------------------------------
  data required time                                19.5416
  data arrival time                                 -1.8083
  -----------------------------------------------------------
  slack (MET)                                       17.7332
```

Fig: pre scan netlist- worst case timing path

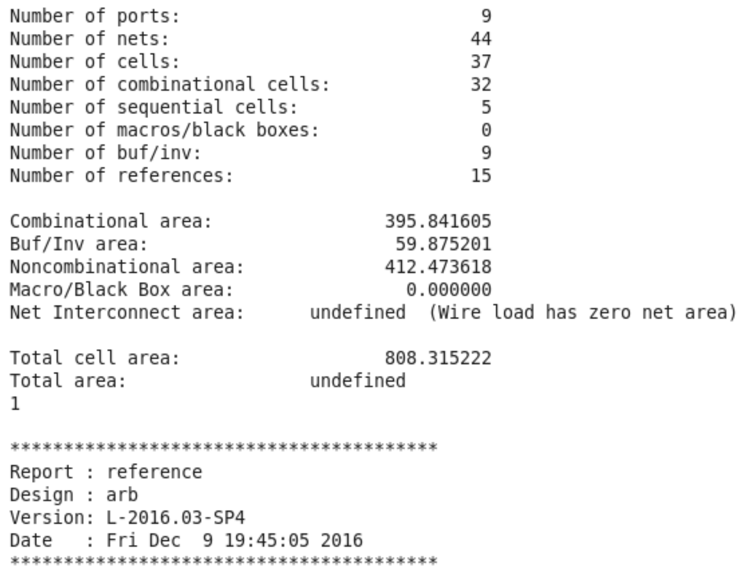
4: Power consumption (dynamic and leakage) for pre scan Netlist:



```
Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  18.1995 uW   (80%)
  Net Switching Power  =   4.5133 uW   (20%)
                         ---------
Total Dynamic Power    =  22.7129 uW  (100%)

Cell Leakage Power     =   2.4690 nW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)
register       1.6464e-02       8.6418e-04            1.6956        1.7330e-02  (  76.29%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational  1.7350e-03       3.6492e-03            0.7734        5.3850e-03  (  23.71%)
--------------------------------------------------------------------------------------------------
Total          1.8200e-02 mW    4.5133e-03 mW         2.4690 nW     2.2715e-02 mW
1

****************************************
Report : power
        -hier
        -analysis_effort low
Design : arb
Version: L-2016.03-SP4
Date   : Fri Dec  9 19:44:38 2016
****************************************
```

Fig : pre scan – power consumption

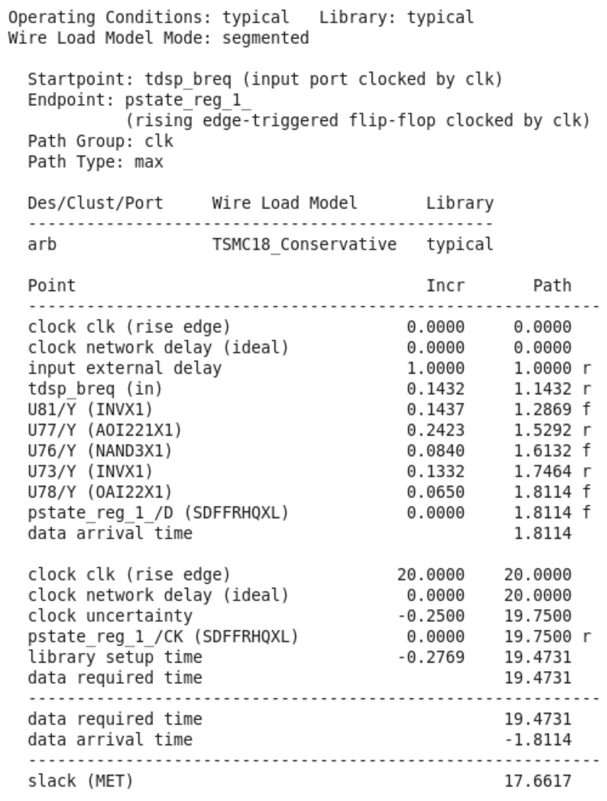
5,6: Total number of cell and Total cell area for post scan Netlist:



```
Number of ports:                            9
Number of nets:                            44
Number of cells:                           37
Number of combinational cells:             32
Number of sequential cells:                 5
Number of macros/black boxes:               0
Number of buf/inv:                          9
Number of references:                      15

Combinational area:                395.841605
Buf/Inv area:                       59.875201
Noncombinational area:             412.473618
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                   808.315222
Total area:                 undefined
1

****************************************
Report : reference
Design : arb
Version: L-2016.03-SP4
Date   : Fri Dec  9 19:45:05 2016
****************************************
```

Fig: post scan cell area and number of cells

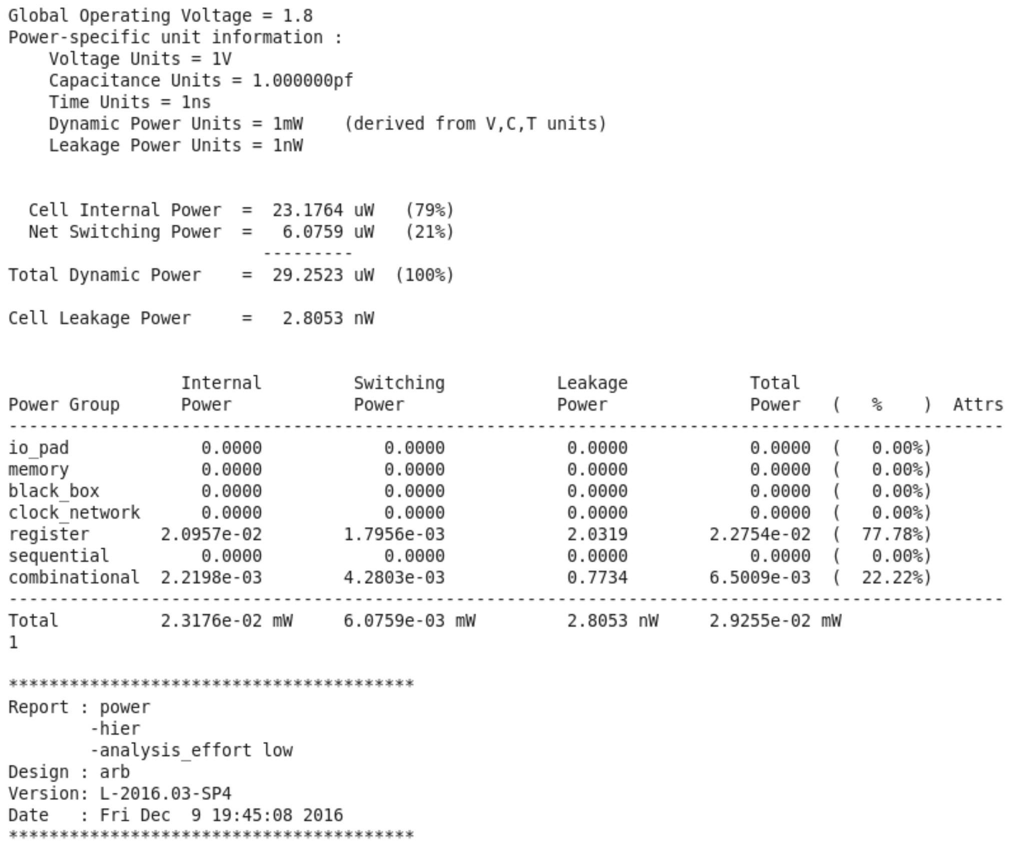
7: Worst case timing path for post scan Netlist:



```
Operating Conditions: typical   Library: typical
Wire Load Model Mode: segmented

  Startpoint: tdsp_breq (input port clocked by clk)
  Endpoint: pstate_reg_1_
            (rising edge-triggered flip-flop 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
  input external delay                   1.0000     1.0000 r
  tdsp_breq (in)                         0.1432     1.1432 r
  U81/Y (INVX1)                          0.1437     1.2869 f
  U77/Y (AOI221X1)                       0.2423     1.5292 r
  U76/Y (NAND3X1)                        0.0840     1.6132 f
  U73/Y (INVX1)                          0.1332     1.7464 r
  U78/Y (OAI22X1)                        0.0650     1.8114 f
  pstate_reg_1_/D (SDFFRHQXL)            0.0000     1.8114 f
  data arrival time                                 1.8114

  clock clk (rise edge)                 20.0000    20.0000
  clock network delay (ideal)            0.0000    20.0000
  clock uncertainty                     -0.2500    19.7500
  pstate_reg_1_/CK (SDFFRHQXL)           0.0000    19.7500 r
  library setup time                    -0.2769    19.4731
  data required time                               19.4731
  ----------------------------------------------------------
  data required time                               19.4731
  data arrival time                                -1.8114
  ----------------------------------------------------------
  slack (MET)                                      17.6617
```

Fig: post scan – worst case timing path

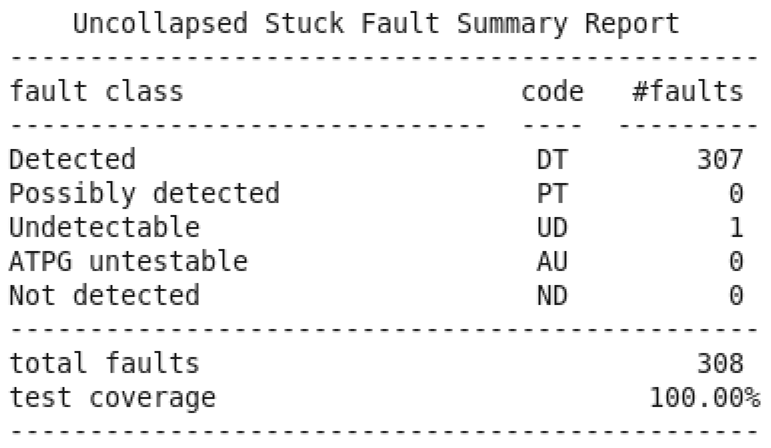
8: Power consumption (leakage and dynamic) for post scan Netlist:



```
Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  23.1764 uW   (79%)
  Net Switching Power  =   6.0759 uW   (21%)
                         ---------
Total Dynamic Power    =  29.2523 uW  (100%)

Cell Leakage Power     =   2.8053 nW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register       2.0957e-02       1.7956e-03            2.0319         2.2754e-02  (  77.78%)
sequential         0.0000           0.0000            0.0000             0.0000  (   0.00%)
combinational  2.2198e-03       4.2803e-03            0.7734         6.5009e-03  (  22.22%)
--------------------------------------------------------------------------------------------------
Total          2.3176e-02 mW    6.0759e-03 mW         2.8053 nW      2.9255e-02 mW
1

****************************************
Report : power
        -hier
        -analysis_effort low
Design : arb
Version: L-2016.03-SP4
Date   : Fri Dec  9 19:45:08 2016
****************************************
```

Fig: Post scan power analysis

9: DFT test coverage for the post scan Netlist

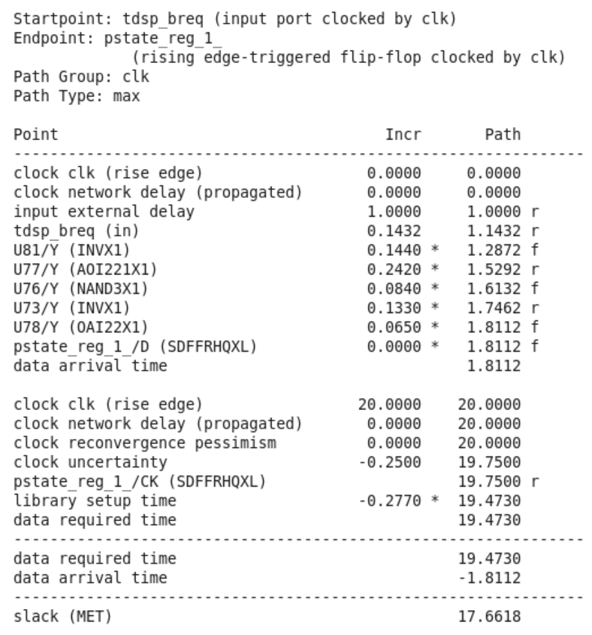


```
    Uncollapsed Stuck Fault Summary Report
 -----------------------------------------------
 fault class                     code   #faults
 ------------------------------  ----  ---------
 Detected                         DT        307
 Possibly detected                PT          0
 Undetectable                     UD          1
 ATPG untestable                  AU          0
 Not detected                     ND          0
 -----------------------------------------------
 total faults                               308
 test coverage                          100.00%
 -----------------------------------------------
```

Fig: post scan test coverage

g. Timing analyzer report:

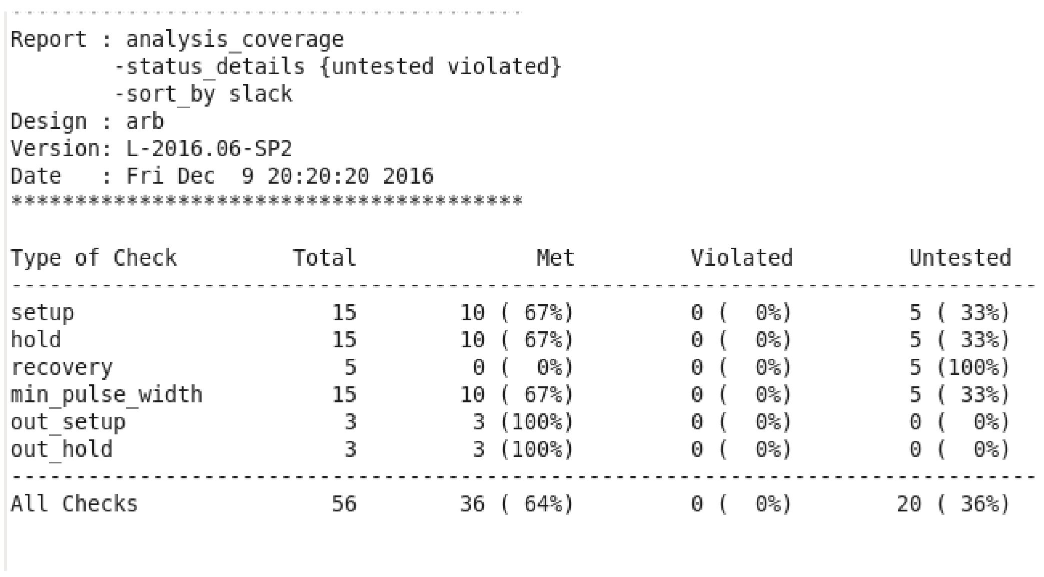
1: worst case timing path for the post scan Netlist:



```
Startpoint: tdsp_breq (input port clocked by clk)
Endpoint: pstate_reg_1_
            (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Point                                    Incr       Path
---------------------------------------------------------------
clock clk (rise edge)                  0.0000     0.0000
clock network delay (propagated)       0.0000     0.0000
input external delay                   1.0000     1.0000 r
tdsp_breq (in)                         0.1432     1.1432 r
U81/Y (INVX1)                          0.1440 *   1.2872 f
U77/Y (AOI221X1)                       0.2420 *   1.5292 r
U76/Y (NAND3X1)                        0.0840 *   1.6132 f
U73/Y (INVX1)                          0.1330 *   1.7462 r
U78/Y (OAI22X1)                        0.0650 *   1.8112 f
pstate_reg_1_/D (SDFFRHQXL)            0.0000 *   1.8112 f
data arrival time                                 1.8112

clock clk (rise edge)                 20.0000    20.0000
clock network delay (propagated)       0.0000    20.0000
clock reconvergence pessimism          0.0000    20.0000
clock uncertainty                     -0.2500    19.7500
pstate_reg_1_/CK (SDFFRHQXL)                     19.7500 r
library setup time                    -0.2770 *  19.4730
data required time                               19.4730
---------------------------------------------------------------
data required time                               19.4730
data arrival time                                -1.8112
---------------------------------------------------------------
slack (MET)                                      17.6618
```

Fig: worst case timing path for post scan netlist

2: Total timing analysis coverage for the post scan Netlist:



```
Report : analysis_coverage
        -status_details {untested violated}
        -sort_by slack
Design : arb
Version: L-2016.06-SP2
Date   : Fri Dec  9 20:20:20 2016
****************************************

Type of Check         Total             Met         Violated         Untested
--------------------------------------------------------------------------------
setup                    15        10 ( 67%)         0 (  0%)         5 ( 33%)
hold                     15        10 ( 67%)         0 (  0%)         5 ( 33%)
recovery                  5         0 (  0%)         0 (  0%)         5 (100%)
min_pulse_width          15        10 ( 67%)         0 (  0%)         5 ( 33%)
out_setup                 3         3 (100%)         0 (  0%)         0 (  0%)
out_hold                  3         3 (100%)         0 (  0%)         0 (  0%)
--------------------------------------------------------------------------------
All Checks               56        36 ( 64%)         0 (  0%)        20 ( 36%)
```

Fig: Total timing analysis for post scan netlist

Section 3- DTMF receiver core RTL database including new memory access bus arbiter(ARB) module:

A: Screen shot of RTL level simulation log and waveform for HUMM and DTMF verification

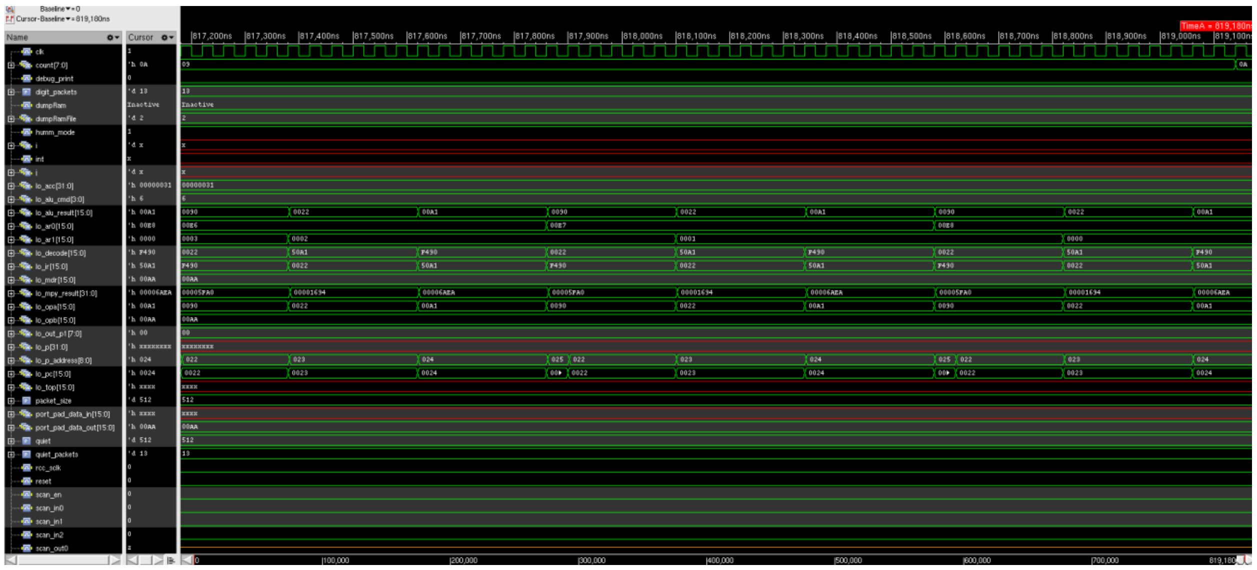
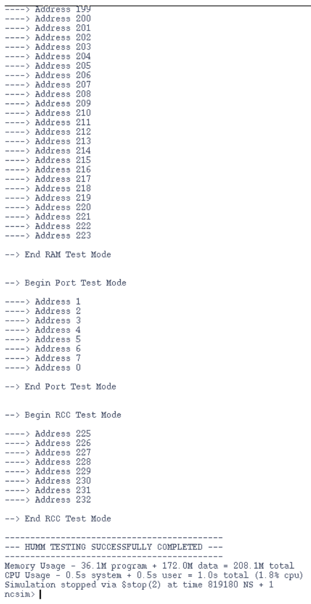


Fig: RTL simulation waveform



```
----> Address 199
----> Address 200
----> Address 201
----> Address 202
----> Address 203
----> Address 204
----> Address 205
----> Address 206
----> Address 207
----> Address 208
----> Address 209
----> Address 210
----> Address 211
----> Address 212
----> Address 213
----> Address 214
----> Address 215
----> Address 216
----> Address 217
----> Address 218
----> Address 219
----> Address 220
----> Address 221
----> Address 222
----> Address 223

--> End RAM Test Mode


--> Begin Port Test Mode

----> Address 1
----> Address 2
----> Address 3
----> Address 4
----> Address 5
----> Address 6
----> Address 7
----> Address 0

--> End Port Test Mode


--> Begin RCC Test Mode

----> Address 225
----> Address 226
----> Address 227
----> Address 228
----> Address 229
----> Address 230
----> Address 231
----> Address 232

--> End RCC Test Mode

-------------------------------------------
--- HUMM TESTING SUCCESSFULLY COMPLETED ---
-------------------------------------------
Memory Usage - 36.1M program + 172.0M data = 208.1M total
CPU Usage - 0.5s system + 0.5s user = 1.0s total (1.8% cpu)
Simulation stopped via $stop(2) at time 819180 NS + 1
ncsim> |
```

Fig: Console window with report

B. screen shot of netlist level simulation log and waveforms for HUMM and DTMF receiver verification suites:

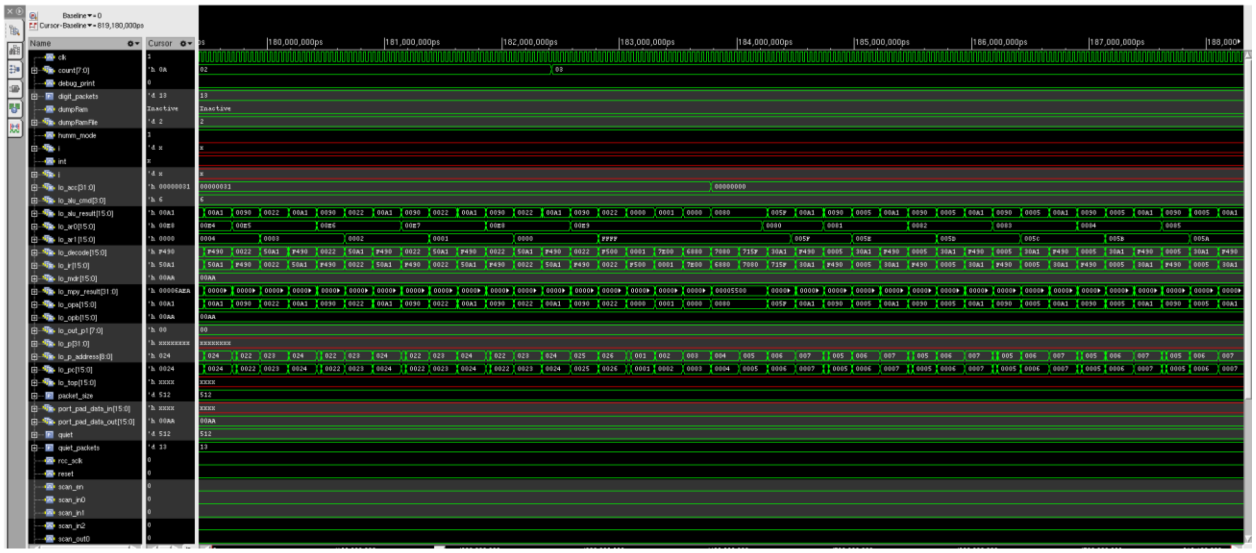
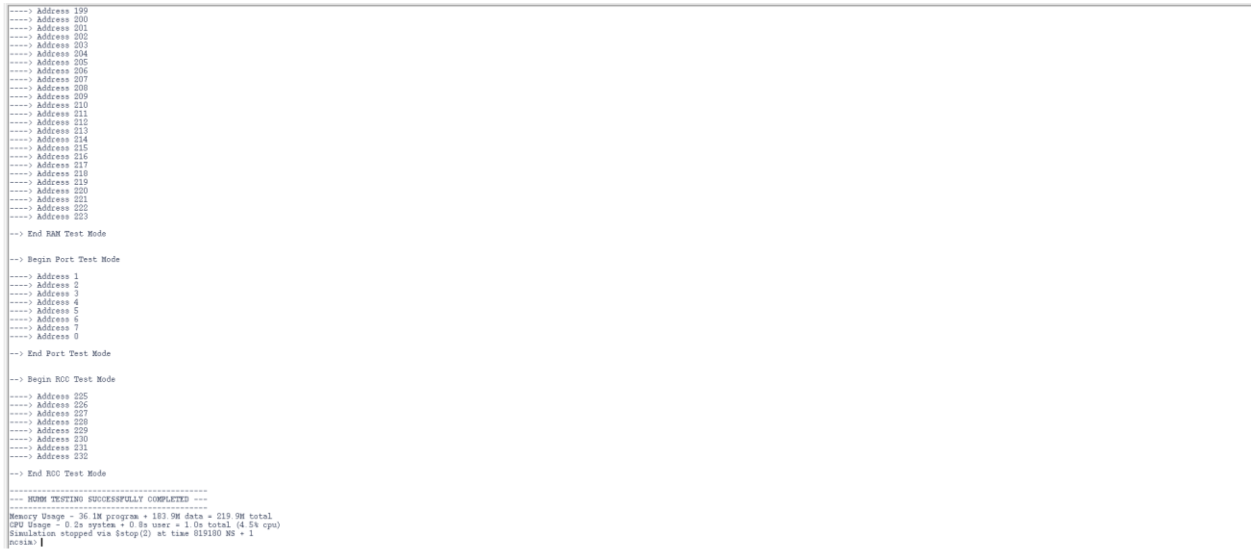


Fig: Netlist simulation waveform



```
----> Address 199
----> Address 200
----> Address 201
----> Address 202
----> Address 203
----> Address 204
----> Address 205
----> Address 206
----> Address 207
----> Address 208
----> Address 209
----> Address 210
----> Address 211
----> Address 212
----> Address 213
----> Address 214
----> Address 215
----> Address 216
----> Address 217
----> Address 218
----> Address 219
----> Address 220
----> Address 221
----> Address 222
----> Address 223

--> End RAM Test Mode


--> Begin Port Test Mode

----> Address 1
----> Address 2
----> Address 3
----> Address 4
----> Address 5
----> Address 6
----> Address 7
----> Address 0

--> End Port Test Mode


--> Begin RCC Test Mode

----> Address 225
----> Address 226
----> Address 227
----> Address 228
----> Address 229
----> Address 230
----> Address 231
----> Address 232

--> End RCC Test Mode

------------------------------------------
--- HUMM TESTING SUCCESSFULLY COMPLETED ---
------------------------------------------
Memory Usage - 36.1M program + 183.9M data = 219.9M total
CPU Usage - 0.2s system + 0.8s user = 1.0s total (4.5% cpu)
Simulation stopped via $stop(2) at time 819180 NS + 1
ncsim> |
```

Fig: Console window with report analysis

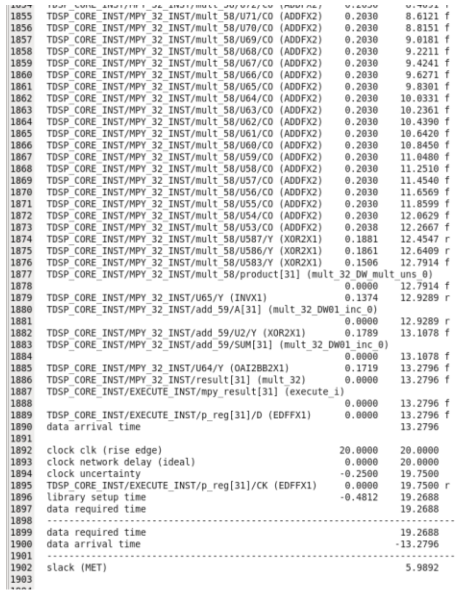
C: **Logic synthesis report:**

1, 2: Total cell area and total number of cells for the Pre scan netlist:

```
****************************************
Report : area
Design : dtmf_recvr_core
Version: L-2016.03-SP4
Date   : Fri Dec  9 23:16:24 2016
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    typical (File: /classes/ee620/maieee/lib/tsmc-0.18/synopsys/typical.db)
    ram_256x16_typical (File: /classes/ee620/maieee/lib/tsmc-0.18/model/ram_256x16_typical.db)
    ram_128x16_typical (File: /classes/ee620/maieee/lib/tsmc-0.18/model/ram_128x16_typical.db)
    rom_512x16_typical (File: /classes/ee620/maieee/lib/tsmc-0.18/model/rom_512x16_typical.db)

Number of ports:                         2921
Number of nets:                          7825
Number of cells:                         4778
Number of combinational cells:           4074
Number of sequential cells:               659
Number of macros/black boxes:               4
Number of buf/inv:                        882
Number of references:                      15

Combinational area:              76600.339948
Buf/Inv area:                     5974.214475
Noncombinational area:           41224.075741
Macro/Black Box area:           213123.171875
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                330947.587564
Total area:                 undefined
1
```

Fig: Cell area calculation

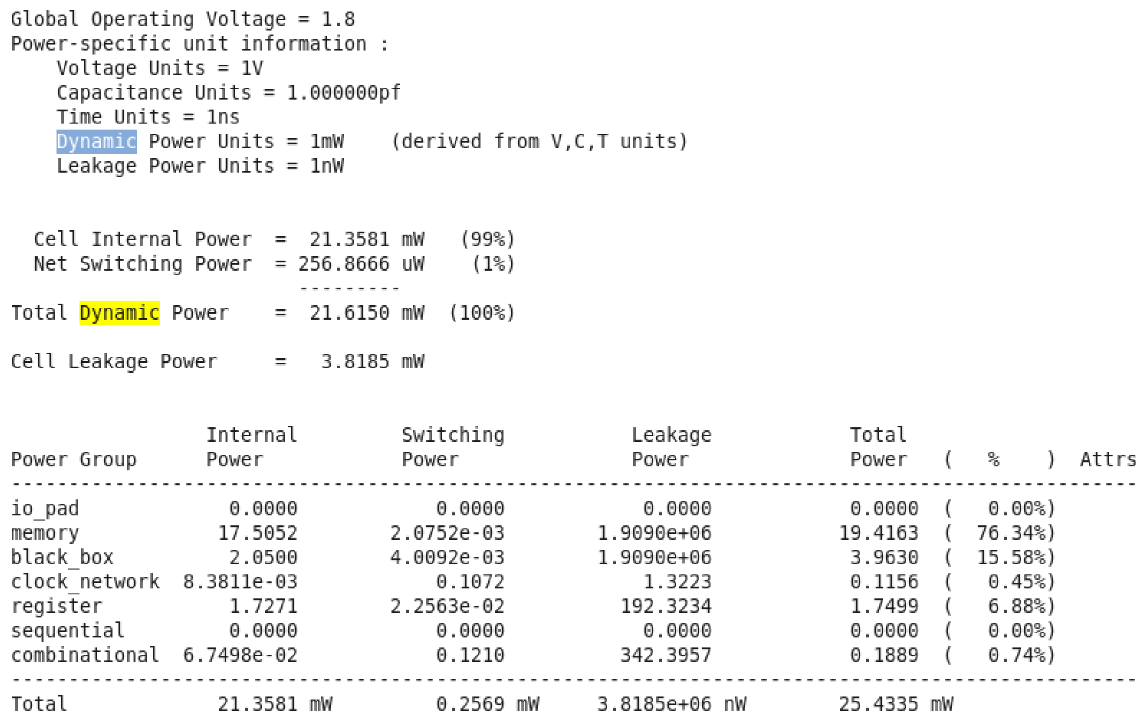
3: Worst case timing path for the pre scan netlist:



```
TDSP_CORE_INST/MPY_32_INST/mult_58/U71/CO (ADDFX2)    0.2030     8.6121 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U70/CO (ADDFX2)    0.2030     8.8151 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U69/CO (ADDFX2)    0.2030     9.0181 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U68/CO (ADDFX2)    0.2030     9.2211 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U67/CO (ADDFX2)    0.2030     9.4241 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U66/CO (ADDFX2)    0.2030     9.6271 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U65/CO (ADDFX2)    0.2030     9.8301 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U64/CO (ADDFX2)    0.2030    10.0331 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U63/CO (ADDFX2)    0.2030    10.2361 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U62/CO (ADDFX2)    0.2030    10.4390 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U61/CO (ADDFX2)    0.2030    10.6420 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U60/CO (ADDFX2)    0.2030    10.8450 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U59/CO (ADDFX2)    0.2030    11.0480 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U58/CO (ADDFX2)    0.2030    11.2510 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U57/CO (ADDFX2)    0.2030    11.4540 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U56/CO (ADDFX2)    0.2030    11.6569 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U55/CO (ADDFX2)    0.2030    11.8599 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U54/CO (ADDFX2)    0.2030    12.0629 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U53/CO (ADDFX2)    0.2038    12.2667 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U587/Y (XOR2X1)    0.1881    12.4547 r
TDSP_CORE_INST/MPY_32_INST/mult_58/U586/Y (XOR2X1)    0.1861    12.6409 r
TDSP_CORE_INST/MPY_32_INST/mult_58/U583/Y (XOR2X1)    0.1506    12.7914 f
TDSP_CORE_INST/MPY_32_INST/mult_58/product[31] (mult_32_DW_mult_uns_0)
                                                      0.0000    12.7914 f
TDSP_CORE_INST/MPY_32_INST/U65/Y (INVX1)              0.1374    12.9289 r
TDSP_CORE_INST/MPY_32_INST/add_59/A[31] (mult_32_DW01_inc_0)
                                                      0.0000    12.9289 r
TDSP_CORE_INST/MPY_32_INST/add_59/U2/Y (XOR2X1)       0.1789    13.1078 f
TDSP_CORE_INST/MPY_32_INST/add_59/SUM[31] (mult_32_DW01_inc_0)
                                                      0.0000    13.1078 f
TDSP_CORE_INST/MPY_32_INST/U64/Y (OAI2BB2X1)          0.1719    13.2796 f
TDSP_CORE_INST/MPY_32_INST/result[31] (mult_32)       0.0000    13.2796 f
TDSP_CORE_INST/EXECUTE_INST/mpy_result[31] (execute_i)
                                                      0.0000    13.2796 f
TDSP_CORE_INST/EXECUTE_INST/p_reg[31]/D (EDFFX1)      0.0000    13.2796 f
data arrival time                                               13.2796

clock clk (rise edge)                                20.0000    20.0000
clock network delay (ideal)                           0.0000    20.0000
clock uncertainty                                    -0.2500    19.7500
TDSP_CORE_INST/EXECUTE_INST/p_reg[31]/CK (EDFFX1)     0.0000    19.7500 r
library setup time                                   -0.4812    19.2688
data required time                                              19.2688
---------------------------------------------------------------------------
data required time                                              19.2688
data arrival time                                              -13.2796
---------------------------------------------------------------------------
slack (MET)                                                      5.9892
```

Fig: worst case timing path

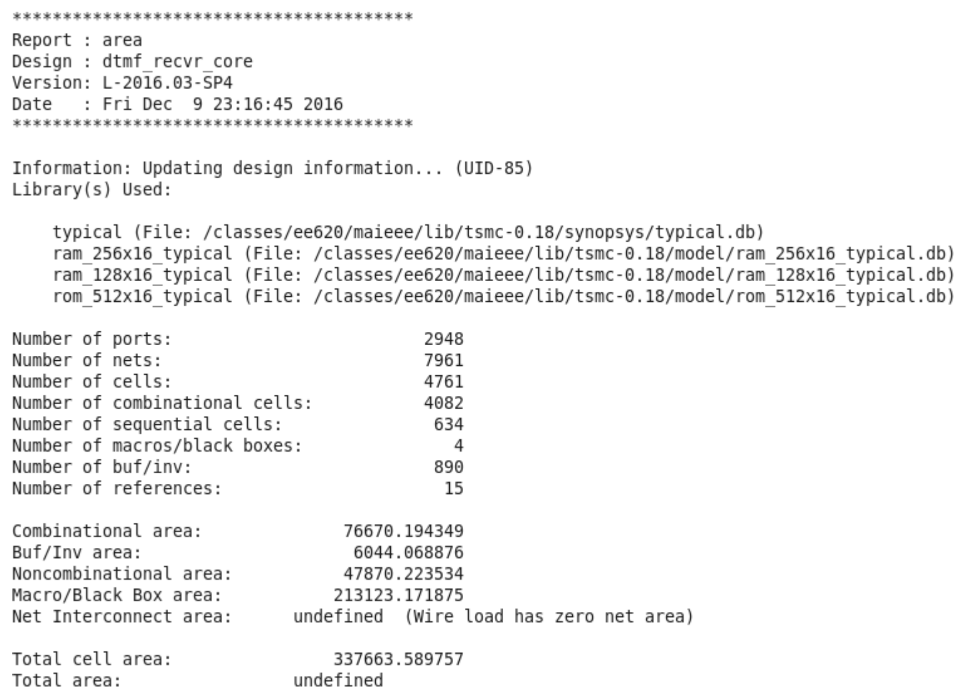
4: **power consumption (dynamic and leakage) for pre scan netlist:**



```
Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  21.3581 mW   (99%)
  Net Switching Power  = 256.8666 uW    (1%)
                         ---------
Total Dynamic Power    =  21.6150 mW  (100%)

Cell Leakage Power     =   3.8185 mW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000             0.0000            0.0000  (   0.00%)
memory            17.5052        2.0752e-03        1.9090e+06           19.4163  (  76.34%)
black_box          2.0500        4.0092e-03        1.9090e+06            3.9630  (  15.58%)
clock_network  8.3811e-03           0.1072             1.3223            0.1156  (   0.45%)
register           1.7271        2.2563e-02          192.3234            1.7499  (   6.88%)
sequential         0.0000           0.0000             0.0000            0.0000  (   0.00%)
combinational  6.7498e-02           0.1210           342.3957            0.1889  (   0.74%)
--------------------------------------------------------------------------------------------------
Total             21.3581 mW        0.2569 mW     3.8185e+06 nW        25.4335 mW
```

Fig: Power consumption for pre scan netlist

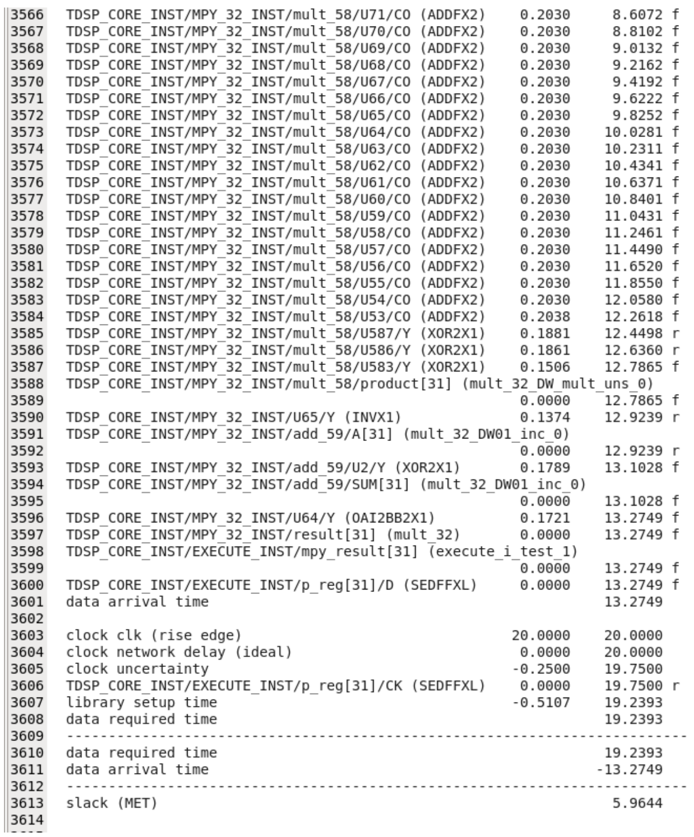
5,6: **Total cell area and total number of cell area for post scan:**



```
****************************************
Report : area
Design : dtmf_recvr_core
Version: L-2016.03-SP4
Date   : Fri Dec  9 23:16:45 2016
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    typical (File: /classes/ee620/maieee/lib/tsmc-0.18/synopsys/typical.db)
    ram_256x16_typical (File: /classes/ee620/maieee/lib/tsmc-0.18/model/ram_256x16_typical.db)
    ram_128x16_typical (File: /classes/ee620/maieee/lib/tsmc-0.18/model/ram_128x16_typical.db)
    rom_512x16_typical (File: /classes/ee620/maieee/lib/tsmc-0.18/model/rom_512x16_typical.db)

Number of ports:                         2948
Number of nets:                          7961
Number of cells:                         4761
Number of combinational cells:           4082
Number of sequential cells:               634
Number of macros/black boxes:               4
Number of buf/inv:                        890
Number of references:                      15

Combinational area:              76670.194349
Buf/Inv area:                     6044.068876
Noncombinational area:           47870.223534
Macro/Black Box area:           213123.171875
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                337663.589757
Total area:                 undefined
```

Fig: cell area and number of cells for Post scan Netlist

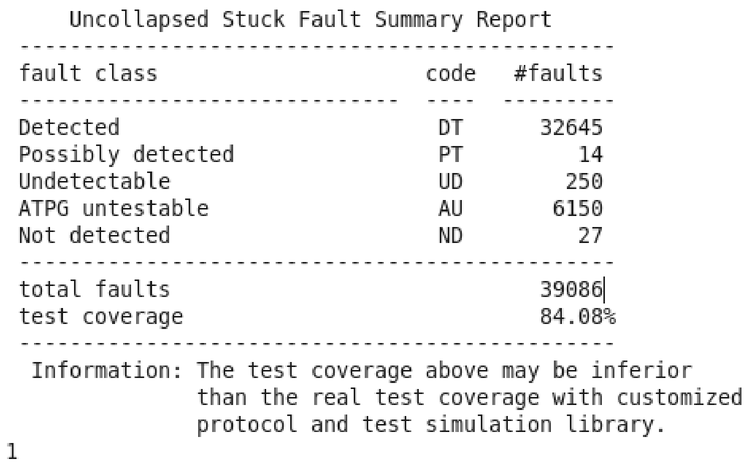
7: **worst case timing path for post scan netlist:**



```
TDSP_CORE_INST/MPY_32_INST/mult_58/U71/CO (ADDFX2)    0.2030     8.6072 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U70/CO (ADDFX2)    0.2030     8.8102 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U69/CO (ADDFX2)    0.2030     9.0132 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U68/CO (ADDFX2)    0.2030     9.2162 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U67/CO (ADDFX2)    0.2030     9.4192 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U66/CO (ADDFX2)    0.2030     9.6222 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U65/CO (ADDFX2)    0.2030     9.8252 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U64/CO (ADDFX2)    0.2030    10.0281 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U63/CO (ADDFX2)    0.2030    10.2311 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U62/CO (ADDFX2)    0.2030    10.4341 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U61/CO (ADDFX2)    0.2030    10.6371 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U60/CO (ADDFX2)    0.2030    10.8401 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U59/CO (ADDFX2)    0.2030    11.0431 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U58/CO (ADDFX2)    0.2030    11.2461 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U57/CO (ADDFX2)    0.2030    11.4490 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U56/CO (ADDFX2)    0.2030    11.6520 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U55/CO (ADDFX2)    0.2030    11.8550 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U54/CO (ADDFX2)    0.2030    12.0580 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U53/CO (ADDFX2)    0.2038    12.2618 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U587/Y (XOR2X1)    0.1881    12.4498 r
TDSP_CORE_INST/MPY_32_INST/mult_58/U586/Y (XOR2X1)    0.1861    12.6360 r
TDSP_CORE_INST/MPY_32_INST/mult_58/U583/Y (XOR2X1)    0.1506    12.7865 f
TDSP_CORE_INST/MPY_32_INST/mult_58/product[31] (mult_32_DW_mult_uns_0)
                                                      0.0000    12.7865 f
TDSP_CORE_INST/MPY_32_INST/U65/Y (INVX1)              0.1374    12.9239 r
TDSP_CORE_INST/MPY_32_INST/add_59/A[31] (mult_32_DW01_inc_0)
                                                      0.0000    12.9239 r
TDSP_CORE_INST/MPY_32_INST/add_59/U2/Y (XOR2X1)       0.1789    13.1028 f
TDSP_CORE_INST/MPY_32_INST/add_59/SUM[31] (mult_32_DW01_inc_0)
                                                      0.0000    13.1028 f
TDSP_CORE_INST/MPY_32_INST/U64/Y (OAI2BB2X1)          0.1721    13.2749 f
TDSP_CORE_INST/MPY_32_INST/result[31] (mult_32)       0.0000    13.2749 f
TDSP_CORE_INST/EXECUTE_INST/mpy_result[31] (execute_i_test_1)
                                                      0.0000    13.2749 f
TDSP_CORE_INST/EXECUTE_INST/p_reg[31]/D (SEDFFXL)     0.0000    13.2749 f
data arrival time                                               13.2749

clock clk (rise edge)                                20.0000    20.0000
clock network delay (ideal)                           0.0000    20.0000
clock uncertainty                                    -0.2500    19.7500
TDSP_CORE_INST/EXECUTE_INST/p_reg[31]/CK (SEDFFXL)    0.0000    19.7500 r
library setup time                                   -0.5107    19.2393
data required time                                              19.2393
-----------------------------------------------------------------------
data required time                                              19.2393
data arrival time                                              -13.2749
-----------------------------------------------------------------------
slack (MET)                                                      5.9644
```

Fig: worst case timing path for post scan Netlist

9: **Test coverage for post scan netlist:**

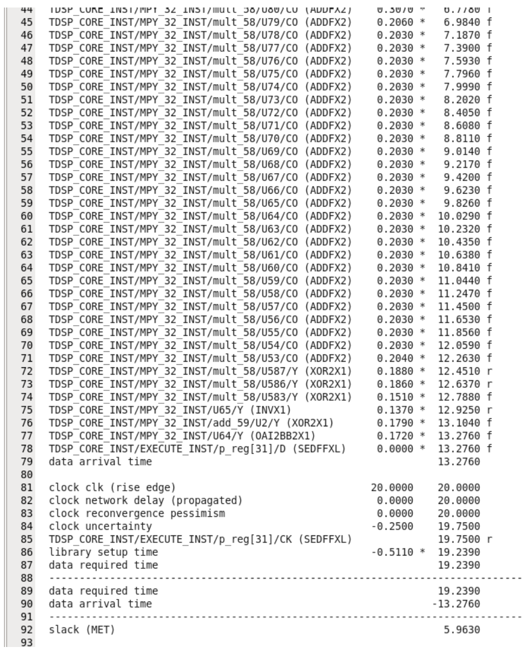


```
    Uncollapsed Stuck Fault Summary Report
 -----------------------------------------------
 fault class                     code   #faults
 ------------------------------  ----  ---------
 Detected                         DT      32645
 Possibly detected                PT         14
 Undetectable                     UD        250
 ATPG untestable                  AU       6150
 Not detected                     ND         27
 -----------------------------------------------
 total faults                             39086
 test coverage                            84.08%
 -----------------------------------------------
  Information: The test coverage above may be inferior
              than the real test coverage with customized
              protocol and test simulation library.
1
```

Fig: Post scan test vector coverage

D: **Timing analyzer report:**

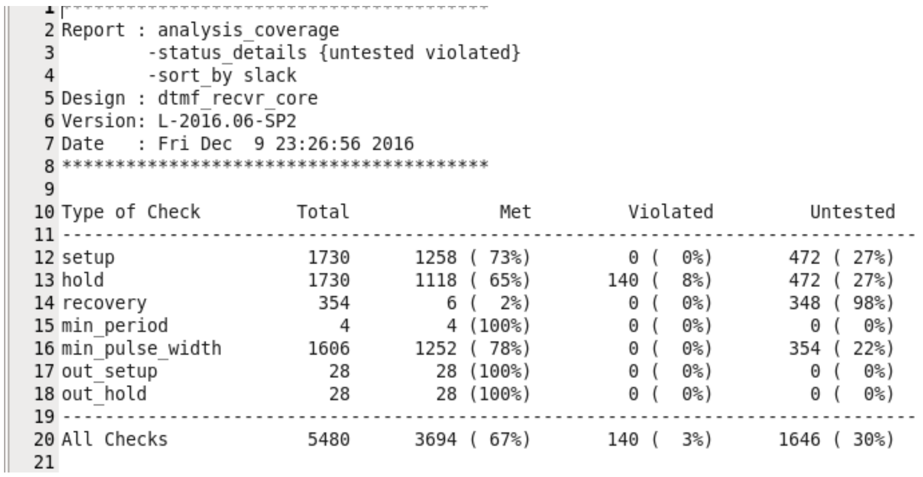
1: worst case timing path for the post scan netlist



```
TDSP_CORE_INST/MPY_32_INST/mult_58/U80/CO (ADDFX2)    0.3070 *   6.7780 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U79/CO (ADDFX2)    0.2060 *   6.9840 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U78/CO (ADDFX2)    0.2030 *   7.1870 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U77/CO (ADDFX2)    0.2030 *   7.3900 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U76/CO (ADDFX2)    0.2030 *   7.5930 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U75/CO (ADDFX2)    0.2030 *   7.7960 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U74/CO (ADDFX2)    0.2030 *   7.9990 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U73/CO (ADDFX2)    0.2030 *   8.2020 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U72/CO (ADDFX2)    0.2030 *   8.4050 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U71/CO (ADDFX2)    0.2030 *   8.6080 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U70/CO (ADDFX2)    0.2030 *   8.8110 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U69/CO (ADDFX2)    0.2030 *   9.0140 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U68/CO (ADDFX2)    0.2030 *   9.2170 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U67/CO (ADDFX2)    0.2030 *   9.4200 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U66/CO (ADDFX2)    0.2030 *   9.6230 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U65/CO (ADDFX2)    0.2030 *   9.8260 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U64/CO (ADDFX2)    0.2030 *  10.0290 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U63/CO (ADDFX2)    0.2030 *  10.2320 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U62/CO (ADDFX2)    0.2030 *  10.4350 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U61/CO (ADDFX2)    0.2030 *  10.6380 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U60/CO (ADDFX2)    0.2030 *  10.8410 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U59/CO (ADDFX2)    0.2030 *  11.0440 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U58/CO (ADDFX2)    0.2030 *  11.2470 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U57/CO (ADDFX2)    0.2030 *  11.4500 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U56/CO (ADDFX2)    0.2030 *  11.6530 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U55/CO (ADDFX2)    0.2030 *  11.8560 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U54/CO (ADDFX2)    0.2030 *  12.0590 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U53/CO (ADDFX2)    0.2040 *  12.2630 f
TDSP_CORE_INST/MPY_32_INST/mult_58/U587/Y (XOR2X1)    0.1880 *  12.4510 r
TDSP_CORE_INST/MPY_32_INST/mult_58/U586/Y (XOR2X1)    0.1860 *  12.6370 r
TDSP_CORE_INST/MPY_32_INST/mult_58/U583/Y (XOR2X1)    0.1510 *  12.7880 f
TDSP_CORE_INST/MPY_32_INST/U65/Y (INVX1)              0.1370 *  12.9250 r
TDSP_CORE_INST/MPY_32_INST/add_59/U2/Y (XOR2X1)       0.1790 *  13.1040 f
TDSP_CORE_INST/MPY_32_INST/U64/Y (OAI2BB2X1)          0.1720 *  13.2760 f
TDSP_CORE_INST/EXECUTE_INST/p_reg[31]/D (SEDFFXL)     0.0000 *  13.2760 f
data arrival time                                               13.2760

clock clk (rise edge)                                20.0000    20.0000
clock network delay (propagated)                      0.0000    20.0000
clock reconvergence pessimism                         0.0000    20.0000
clock uncertainty                                    -0.2500    19.7500
TDSP_CORE_INST/EXECUTE_INST/p_reg[31]/CK (SEDFFXL)              19.7500 r
library setup time                                   -0.5110 *  19.2390
data required time                                              19.2390
---------------------------------------------------------------------------
data required time                                              19.2390
data arrival time                                              -13.2760
---------------------------------------------------------------------------
slack (MET)                                                      5.9630
```

Fig: Worst case timing path for the post scan netlist

2: **Total timing analysis coverage for the post scan netlist:**



```
****************************************
Report : analysis_coverage
        -status_details {untested violated}
        -sort_by slack
Design : dtmf_recvr_core
Version: L-2016.06-SP2
Date   : Fri Dec  9 23:26:56 2016
****************************************

Type of Check         Total             Met         Violated         Untested
--------------------------------------------------------------------------------
setup                  1730      1258 ( 73%)        0 (   0%)      472 ( 27%)
hold                   1730      1118 ( 65%)      140 (   8%)      472 ( 27%)
recovery                354         6 (  2%)        0 (   0%)      348 ( 98%)
min_period                4         4 (100%)        0 (   0%)        0 (   0%)
min_pulse_width        1606      1252 ( 78%)        0 (   0%)      354 ( 22%)
out_setup                28        28 (100%)        0 (   0%)        0 (   0%)
out_hold                 28        28 (100%)        0 (   0%)        0 (   0%)
--------------------------------------------------------------------------------
All Checks             5480      3694 ( 67%)      140 (   3%)     1646 ( 30%)
```

Fig: total timing coverage for post scan Netlist

Section 4: TDSP ASSEMBLY LANGUAGE TEST PROGRAM

a. **program description:**

TDSP assembly language is created to meet the following requirements:

1- Variable x is initiated to the location 13 and value 3 is stored in x.
2- Variable Y is initiated to the location 14 and value 4 is stored in y.
3- Variable z is initiated to the location 15 and value 2 is stored in z.
4- In a loop of 64 cycles, the following commands are executed. The value of “ I “ changes from 0 to 64 inside this loop.
   - Y = 2*x +z-2
   - X = y+5
   - Y = -y
   - Z = y-i

The bit assigned in the project 1 is 101101 and the decimal equivalency is 13.

- Y = 2*x + z – 2
```
LT x        ; x is loaded to T register
MPYK 2      ; T register . 2 is loaded into P register
LAC z       ; load accumulator with Z
APAC        ; accumulator + P register into accumulator
SUBS a      ; accumulator - a = accumulator
SACL y      ; laod accumulator into y
```
- X = y+5
```
LACK 5      ; load accumulator with 5
ADDS y      ; add y + acumulator and load into accumulator
SACL x      ; load accumulator into x
```
- Y = -y
```
LT y        ; load y to T register
MPYK 1      ; T register.1 is loaded to P register
LACK 0      ; load accumulator with 0
SPAC        ; accumulator - P register into accumulator
SACL y      ; load accumulator into Y
```
- Z = y-i
```
LAC Y       ; load accumulator with y
SUBS c      ; accumulator - c is loaded into accumulator
SACL z      ; load accumulator into Z
LACK 1      ; load accumulator with 1
ADDS c      ; add c + accumulator and load into accumulator
SACL c      ; load accumulator into C
```

b**. Test program used to model ASM code:**

```
#include <stdio.h>

int main ()
{
int x,y,z;
while(i<64)
   {
   printf("enter the value of x:");
   scanf("%d,"&x);
   printf("enter the value of x:");
   scanf("%d,"&y);
   printf("enter the value of x:");
   scanf("%d,"&z);
   printf("x=%d,y=%d,z=%d\n,x,y,z");
   y= (2*x) + z - 2;
   x = y +5;
   y = -y;
   z = y - i;
   i++;
}

}
```

c. Assembly language test program source code:

```
ldpk 1

page0           =       0               ; memory page 0
page1           =       1               ; memory page 1
base_page0      =       0x000   ; memory page 0
base_page1      =       0x080   ; memory page 1

ptr             =       21      ;
rcc_ptr  =      22      ;

pow1    =       60
lack 0
sacl pow1
pow2    =       61
lack 0
sacl pow2
pow3    =       62
```

```
lack 0
sacl pow3
pow4     =          63
lack 0
sacl pow4
pow5     =          64
lack 0
sacl pow5
pow6     =          65
lack 0
sacl pow6
pow7     =          66
lack 0
sacl pow7
pow8     =          67
lack 0
sacl pow8


pow_1    =          79
lack 0
sacl pow_1
pow_2    =          80
lack 0
sacl pow_2
pow_3    =          81
lack 0
sacl pow_3
pow_4    =          82
lack 1
sacl pow_4
pow_5    =          83
lack 0
sacl pow_5
pow_6    =          84
lack 0
sacl pow_6
pow_7    =          85
lack 1
sacl pow_7
pow_8    =          86
lack 0
sacl pow_8



rcc_kick = (0x00e8 & 0x07f)
rcc_len             = (rcc_kick - rcc_697Hz)
rcc_697Hz           = (0x00e0 & 0x07f)
```

```
x = 13
lack 3
sacl x

y = 14
lack 4
sacl y

z = 15
lack 2
sacl z

a = 16
lack 2
sacl a

m = 17
lack 64
sacl m

lar ar0, m

n = 19
lack 15
sacl n
lar ar1,n


c = 18
lack 0
sacl c

;lark ar0, 63
loop1:

lt x
mpyk 2
lac z
apac
subs a
sacl y

lack 5
add y
sacl x

lt y
mpyk 1
lack 0
spac
```

```
sacl y



lac y
subs c
sacl z

lack 1
adds c
sacl c

banz loop1, *-, ar0

_out_sp_l1:     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_page1)
                        sar             ar0,ptr
d_out_sp_l:     lar             ar0,ptr
                        zals    *+
                        sar             ar0,ptr
                        lar     ar0,rcc_ptr
                        sacl    *+,ar1
                        sar             ar0,rcc_ptr
                        banz    d_out_sp_l,*-,ar0

lark ar0, 120
loop2:
banz loop2, *-, ar0



d_out_sp_l2:    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_page1)
                        sar             ar0,ptr
d_out_sp_l_1:   lar             ar0,ptr
                        zals    *+
                        sar             ar0,ptr
                        lar     ar0,rcc_ptr
                        sacl    *+,ar1
                        sar             ar0,rcc_ptr
                        banz    d_out_sp_l_1,*-,ar0

lark ar0, 120
loop3:
banz loop3, *-, ar0
```

```
d_out_sp_l3:    lark    ar1,rcc_len                 ; length of rcc register block
                        lark    ar0,(rcc_697Hz+base_page1)      ; starting address
                        sar             ar0,rcc_ptr
                        lark    ar0,(pow_1+base_page1)
                        sar             ar0,ptr
d_out_sp_l_2:   lar             ar0,ptr
                        zals    *+
                        sar             ar0,ptr
                        lar     ar0,rcc_ptr
                        sacl    *+,ar1
                        sar             ar0,rcc_ptr
                        banz    d_out_sp_l_2,*-,ar0



;lark ar0, 120
;loop4:
;banz loop4, *-, ar0


d_out_sp_l4:    lark    ar1,rcc_len                 ; length of rcc register block
                        lark    ar0,(rcc_697Hz+base_page1)      ; starting address
                        sar             ar0,rcc_ptr
                        lark    ar0,(pow_1+base_page1)
                        sar             ar0,ptr
d_out_sp_l_3:   lar             ar0,ptr
                        zals    *+
                        sar             ar0,ptr
                        lar     ar0,rcc_ptr
                        sacl    *+,ar1
                        sar             ar0,rcc_ptr
                        banz    d_out_sp_l_3,*-,ar0


;lark ar0, 120
;loop5:
;banz loop5, *-, ar0



d_out_sp_l5:    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_page1)
                        sar             ar0,ptr
d_out_sp_l_4:   lar             ar0,ptr
                        zals    *+
                        sar             ar0,ptr
                        lar     ar0,rcc_ptr
```

```
                              sacl      *+,ar1
                              sar                 ar0,rcc_ptr
                              banz      d_out_sp_l_4,*-,ar0



;lark ar0, 120
;loop6:
;banz loop6, *-, ar0


d_out_sp_l6:    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_page1)
                              sar                 ar0,ptr
d_out_sp_l_5:   lar                     ar0,ptr
                              zals      *+
                              sar                 ar0,ptr
                              lar    ar0,rcc_ptr
                              sacl      *+,ar1
                              sar                 ar0,rcc_ptr
                              banz      d_out_sp_l_5,*-,ar0


;lark ar0, 120
;loop7:
;banz loop7, *-, ar0

end:
```

d**. Assembly language test program merged listing:**

```
  tdspasm, RCS v1.1.1.1
  Listing for module: "dab8730_test"
  Prepared: Fri Dec  9 03:58:55 EST 2016
*******************************************
          0000 6e01          ldpk     1
          0001 7e00          lack     0
          0002 503c          sacl     pow1
          0003 7e00          lack     0
          0004 503d          sacl     pow2
          0005 7e00          lack     0
          0006 503e          sacl     pow3
          0007 7e00          lack     0
          0008 503f          sacl     pow4
          0009 7e00          lack     0
          000a 5040          sacl     pow5
          000b 7e00          lack     0
          000c 5041          sacl     pow6
```

```
000d 7e00              lack      0
000e 5042              sacl      pow7
000f 7e00              lack      0
0010 5043              sacl      pow8
0011 7e00              lack      0
0012 504f              sacl      pow_1
0013 7e00              lack      0
0014 5050              sacl      pow_2
0015 7e00              lack      0
0016 5051              sacl      pow_3
0017 7e01              lack      1
0018 5052              sacl      pow_4
0019 7e00              lack      0
001a 5053              sacl      pow_5
001b 7e00              lack      0
001c 5054              sacl      pow_6
001d 7e01              lack      1
001e 5055              sacl      pow_7
001f 7e00              lack      0
0020 5056              sacl      pow_8
0021 7e03              lack      3
0022 500d              sacl      x
0023 7e04              lack      4
0024 500e              sacl      y
0025 7e02              lack      2
0026 500f              sacl      z
0027 7e02              lack      2
0028 5010              sacl      a
0029 7e40              lack      64
002a 5011              sacl      m
002b 3811              lar       ar0,m
002c 7e0f              lack      15
002d 5013              sacl      n
002e 3913              lar       ar1,n
002f 7e00              lack      0
0030 5012              sacl      c
0031              loop1:
0031 400d              lt        x
0032 8002              mpyk      2
0033 200f              lac       z
0034 7f8f              apac
0035 6310              subs      a
0036 500e              sacl      y
0037 7e05              lack      5
0038 000e              add       y
0039 500d              sacl      x
003a 400e              lt        y
003b 8001              mpyk      1
003c 7e00              lack      0
003d 7f90              spac
```

```
003e 500e           sacl    y
003f 200e           lac     y
0040 6312           subs    c
0041 500f           sacl    z
0042 7e01           lack    1
0043 6112           adds    c
0044 5012           sacl    c
0045 f490           banz    loop1,*-,ar0
0046 0031
0047 7108   d_out_sp_l1: lark    ar1,rcc_len
0048 70e0           lark    ar0,(rcc_697Hz+base_page1)
0049 3016           sar     ar0,rcc_ptr
004a 70bc           lark    ar0,(pow1+base_page1)
004b 3015           sar     ar0,ptr
004c 3815   d_out_sp_l: lar     ar0,ptr
004d 66a8           zals    *+
004e 3015           sar     ar0,ptr
004f 3816           lar     ar0,rcc_ptr
0050 50a1           sacl    *+,ar1
0051 3016           sar     ar0,rcc_ptr
0052 f490           banz    d_out_sp_l,*-,ar0
0053 004c
0054 7078           lark    ar0,120
0055        loop2:
0055 f490           banz    loop2,*-,ar0
0056 0055
0057 7108   d_out_sp_l2: lark    ar1,rcc_len
0058 70e0           lark    ar0,(rcc_697Hz+base_page1)
0059 3016           sar     ar0,rcc_ptr
005a 70bc           lark    ar0,(pow1+base_page1)
005b 3015           sar     ar0,ptr
005c 3815   d_out_sp_l_1: lar     ar0,ptr
005d 66a8           zals    *+
005e 3015           sar     ar0,ptr
005f 3816           lar     ar0,rcc_ptr
0060 50a1           sacl    *+,ar1
0061 3016           sar     ar0,rcc_ptr
0062 f490           banz    d_out_sp_l_1,*-,ar0
0063 005c
0064 7078           lark    ar0,120
0065        loop3:
0065 f490           banz    loop3,*-,ar0
0066 0065
0067 7108   d_out_sp_l3: lark    ar1,rcc_len
0068 70e0           lark    ar0,(rcc_697Hz+base_page1)
0069 3016           sar     ar0,rcc_ptr
006a 70cf           lark    ar0,(pow_1+base_page1)
006b 3015           sar     ar0,ptr
006c 3815   d_out_sp_l_2: lar     ar0,ptr
006d 66a8           zals    *+
006e 3015              sar      ar0,ptr
006f 3816              lar      ar0,rcc_ptr
0070 50a1              sacl     *+,ar1
0071 3016              sar      ar0,rcc_ptr
0072 f490              banz     d_out_sp_l_2,*-,ar0
0073 006c
0074 7108    d_out_sp_l4: lark     ar1,rcc_len
0075 70e0              lark     ar0,(rcc_697Hz+base_page1)
0076 3016              sar      ar0,rcc_ptr
0077 70cf              lark     ar0,(pow_1+base_page1)
0078 3015              sar      ar0,ptr
0079 3815   d_out_sp_l_3: lar      ar0,ptr
007a 66a8              zals     *+
007b 3015              sar      ar0,ptr
007c 3816              lar      ar0,rcc_ptr
007d 50a1              sacl     *+,ar1
007e 3016              sar      ar0,rcc_ptr
007f f490              banz     d_out_sp_l_3,*-,ar0
0080 0079
0081 7108    d_out_sp_l5: lark     ar1,rcc_len
0082 70e0              lark     ar0,(rcc_697Hz+base_page1)
0083 3016              sar      ar0,rcc_ptr
0084 70bc              lark     ar0,(pow1+base_page1)
0085 3015              sar      ar0,ptr
0086 3815   d_out_sp_l_4: lar      ar0,ptr
0087 66a8              zals     *+
0088 3015              sar      ar0,ptr
0089 3816              lar      ar0,rcc_ptr
008a 50a1              sacl     *+,ar1
008b 3016              sar      ar0,rcc_ptr
008c f490              banz     d_out_sp_l_4,*-,ar0
008d 0086
008e 7108    d_out_sp_l6: lark     ar1,rcc_len
008f 70e0              lark     ar0,(rcc_697Hz+base_page1)
0090 3016              sar      ar0,rcc_ptr
0091 70bc              lark     ar0,(pow1+base_page1)
0092 3015              sar      ar0,ptr
0093 3815   d_out_sp_l_5: lar      ar0,ptr
0094 66a8              zals     *+
0095 3015              sar      ar0,ptr
0096 3816              lar      ar0,rcc_ptr
0097 50a1              sacl     *+,ar1
0098 3016              sar      ar0,rcc_ptr
0099 f490              banz     d_out_sp_l_5,*-,ar0
009a 0093
009b            end:
```

e**. Assembly language test program symbol table:**

```
 tdspasm, RCS v1.1.1.1
 Symbol listing for module: "dab8730_test"
 Prepared: Fri Dec  9 03:58:55 EST 2016
 * <symbol> = <hex> (<octal>) (<decimal>) <R,A>
 * where: R == relocatable, A == absolute
******************************************
          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
            a = 0x0010 (0000020) (   16) 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
            c = 0x0012 (0000022) (   18) A
   d_out_sp_l = 0x004c (0000114) (   76) R
  d_out_sp_l1 = 0x0047 (0000107) (   71) R
  d_out_sp_l2 = 0x0057 (0000127) (   87) R
  d_out_sp_l3 = 0x0067 (0000147) (  103) R
  d_out_sp_l4 = 0x0074 (0000164) (  116) R
  d_out_sp_l5 = 0x0081 (0000201) (  129) R
  d_out_sp_l6 = 0x008e (0000216) (  142) R
 d_out_sp_l_1 = 0x005c (0000134) (   92) R
 d_out_sp_l_2 = 0x006c (0000154) (  108) R
 d_out_sp_l_3 = 0x0079 (0000171) (  121) R
 d_out_sp_l_4 = 0x0086 (0000206) (  134) R
 d_out_sp_l_5 = 0x0093 (0000223) (  147) R
          end = 0x009b (0000233) (  155) R
        loop1 = 0x0031 (0000061) (   49) R
        loop2 = 0x0055 (0000125) (   85) R
        loop3 = 0x0065 (0000145) (  101) R
            m = 0x0011 (0000021) (   17) A
            n = 0x0013 (0000023) (   19) A
          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
       page1 = 0x0001 (0000001) (    1) A
        pow1 = 0x003c (0000074) (   60) A
        pow2 = 0x003d (0000075) (   61) A
        pow3 = 0x003e (0000076) (   62) A
        pow4 = 0x003f (0000077) (   63) A
        pow5 = 0x0040 (0000100) (   64) A
        pow6 = 0x0041 (0000101) (   65) A
        pow7 = 0x0042 (0000102) (   66) A
        pow8 = 0x0043 (0000103) (   67) A
       pow_1 = 0x004f (0000117) (   79) A
       pow_2 = 0x0050 (0000120) (   80) A
       pow_3 = 0x0051 (0000121) (   81) A
       pow_4 = 0x0052 (0000122) (   82) A
       pow_5 = 0x0053 (0000123) (   83) A
       pow_6 = 0x0054 (0000124) (   84) A
       pow_7 = 0x0055 (0000125) (   85) A
       pow_8 = 0x0056 (0000126) (   86) A
         ptr = 0x0015 (0000025) (   21) A
   rcc_697Hz = 0x0060 (0000140) (   96) A
    rcc_kick = 0x0068 (0000150) (  104) A
     rcc_len = 0x0008 (0000010) (    8) A
     rcc_ptr = 0x0016 (0000026) (   22) A
           x = 0x000d (0000015) (   13) A
           y = 0x000e (0000016) (   14) A
           z = 0x000f (0000017) (   15) A
```

f. **Assembly language test program object file:**

```
//  tdspasm, RCS v1.1.1.1
//  Object for module: "dab8730_test"
//  Prepared: Fri Dec  9 03:58:55 EST 2016
//*******************************************
@0000 6e01
@0001 7e00
@0002 503c
@0003 7e00
@0004 503d
@0005 7e00
@0006 503e
@0007 7e00
@0008 503f
@0009 7e00
@000a 5040
@000b 7e00
```

```
@000c 5041
@000d 7e00
@000e 5042
@000f 7e00
@0010 5043
@0011 7e00
@0012 504f
@0013 7e00
@0014 5050
@0015 7e00
@0016 5051
@0017 7e01
@0018 5052
@0019 7e00
@001a 5053
@001b 7e00
@001c 5054
@001d 7e01
@001e 5055
@001f 7e00
@0020 5056
@0021 7e03
@0022 500d
@0023 7e04
@0024 500e
@0025 7e02
@0026 500f
@0027 7e02
@0028 5010
@0029 7e40
@002a 5011
@002b 3811
@002c 7e0f
@002d 5013
@002e 3913
@002f 7e00
@0030 5012
@0031 400d
@0032 8002
@0033 200f
@0034 7f8f
@0035 6310
@0036 500e
@0037 7e05
@0038 000e
@0039 500d
@003a 400e
@003b 8001
@003c 7e00
@003d 7f90
```

```
@003e 500e
@003f 200e
@0040 6312
@0041 500f
@0042 7e01
@0043 6112
@0044 5012
@0045 f490
@0046 0031
@0047 7108
@0048 70e0
@0049 3016
@004a 70bc
@004b 3015
@004c 3815
@004d 66a8
@004e 3015
@004f 3816
@0050 50a1
@0051 3016
@0052 f490
@0053 004c
@0054 7078
@0055 f490
@0056 0055
@0057 7108
@0058 70e0
@0059 3016
@005a 70bc
@005b 3015
@005c 3815
@005d 66a8
@005e 3015
@005f 3816
@0060 50a1
@0061 3016
@0062 f490
@0063 005c
@0064 7078
@0065 f490
@0066 0065
@0067 7108
@0068 70e0
@0069 3016
@006a 70cf
@006b 3015
@006c 3815
@006d 66a8
@006e 3015
@006f 3816
```

```
@0070 50a1
@0071 3016
@0072 f490
@0073 006c
@0074 7108
@0075 70e0
@0076 3016
@0077 70cf
@0078 3015
@0079 3815
@007a 66a8
@007b 3015
@007c 3816
@007d 50a1
@007e 3016
@007f f490
@0080 0079
@0081 7108
@0082 70e0
@0083 3016
@0084 70bc
@0085 3015
@0086 3815
@0087 66a8
@0088 3015
@0089 3816
@008a 50a1
@008b 3016
@008c f490
@008d 0086
@008e 7108
@008f 70e0
@0090 3016
@0091 70bc
@0092 3015
@0093 3815
@0094 66a8
@0095 3015
@0096 3816
@0097 50a1
@0098 3016
@0099 f490
@009a 0093
```

Section5- Summary and conclusion:

A: Project Summary:

The project is about designing RTL module for ARB and moving the designed module to the DTMF block and performs some functional analysis to check the working of DTMF receiver. This project gave a very good understanding in how RTL Verilog is used to module a block and how the DTMF receiver works in real world. In top of that assembly level instructions are used to perform few arithmetic functions which gave a hand in understanding assembly language and how it could be in different possibilities. Overall, Project was helpful in a many ways.

B. Conclusion:

1- What went well:

Honestly I had hard time in this project. I was not a not a great programmer, I had to go through Verilog basics and working in RTL level was hard at start. I feel this project moved me from knowing nothing in Verilog to some position where I could design a module with given specification. So yes, My RTL code went well. I did not find any difficulty in understanding the assembly language, It was pretty easy and went good and I got the output as expected. The part where I had to emulate 2 tone of quiet signal followed by # key was also pretty decent. I just had to go through the dtmf.asm file and understand what was going on. I would everything went easy on me expect the test bench.

2- What did not go well:

Well I was not satisfied with my test bench on Tdsp_check condition. It worked on certain instance but on some instance the error was not reported on the first occurrence of Tdsp_breq but on few cycles later my tdsp_error went high and error was reported. I had figured what why was that and that was because I made by conditions depend on dma_grant so on some instance where dma_grant didn't coincide with tdsp_breq those were skipped. So I tried to change my condition and tried to make it work which was real pain.

3- What I leant:

I learned Verilog and got some idea about how assembly language works. Performing some arithmetic operations using assembly language made me analyze the way how machine takes the values and how it gets stored in register and how they get processed one after other and how they are moved from register to dma. This project also gave me knowledge about DTMF receiver and how this DTMF receiver works. I'm pretty much interested in designing module which is used in blocks and that block performs a function which in turns gets to be an application. Pretty interesting, I hope I would learn more in advanced design of digital system.