ELEC6231/VLSI DESIGN PROJECT

CYCLE COMPUTER
FINAL PROJECT REPORT

Team Name: M1
First Examiner: Dr Terrence Mak
Second Examiner: Iain McNally
Student Name:Xinyi Qu,Yuting Gan, Yunfan Ruan
Username: xq1g19,yg2m19,yr1g19

May 26, 2020

Abstract

This project aims to design and implement a cycle computer on a single CMOS chip. In our project, it is utilized AMS 0.35um CMOS technology to test our cycle computer. There are five functions for this design, including odometer, speedometer, trip timer, cadence and variable wheel size. All functions were simulated and the results were correct. The implementation of this design has been verified. Clock and reset were synchronised for full chip design. The constraints were satisfied, including area, time and power. All design rules were passed.

1 PROBLEMS AND GOALS

In the design of the cycle computer, some basic ideas should be discussed to complete this project. Regrading to the design of this cycle computer, the functionality and implementation should be consider. The trade-off between the area of the chip and the complexity of the design is hard to decide.Furthermore, the result has small deviation compared to the theoretical values.

During the simulation of the this cycle computer, some problems were encountered. Initially,When reading the results via LED, one result string was missing due to the incorrect decoder.when we test the behavioural model, the results were displayed disordered with respect to the order of the mode, and the digit 3 which displays the character of the mode was wrong. Moreover, there are only 3 digits to display the results hence the residual fraction part will be ignored or increased. In this case, the calculation results had differences in the simulation results. When implement the design into chip, the gate level was failed at the beginning due to the no reset synchronisation in wrapper. Finally, most problems were solved expect the sign-off.

This project aims to design a cycle computer and then implement it based on a CMOS chip. The objectives of this project are:

- Write a design proposal to decide the approach, specifications and work division of the design.
- Write a initial behavioural model to test single function and display it on LED screen
- Write a behavioural model in system verilog to satisfy the basic and advanced function of cycle computer.
- Simulate the design to verify the results adjust its accuracy.
- Gate-level design with gate delays to support input and reset synchronisation and scan path.
- Simulate the design to support a variable wheel size.
- Synthesis and place and route to optimize the design.
- Complete cadence design to check design rules.

2 SPECIFICATIONS AND ANALYSIS

2.1 Specifications

This cycle computer provides five basic functions for users, including travelled distance, travelled time, current speed, cadence and adjustable wheel size. To implement this design into a small chip area, it is utilized the hard-only design method. Additionally, the divider was simple that can be used repeatedly among different calculation. Only one divider can be used in calculating current speed and cadence, therefore, the chip area was reduced.

2.2 Result Calculation

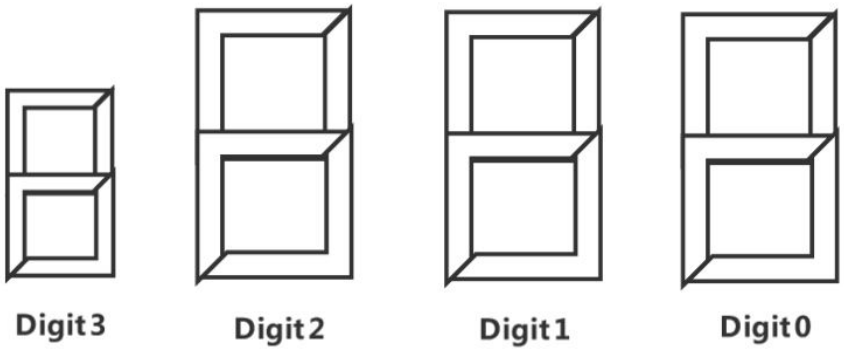

Figure 1: LED Digits

According to the Figure 1, the seven-segment LED has four digits. In this LED screen, Digit3 represents the character of each mode, where d represent the travelled distance, t represents the travelled time, c represents the cadence, v represents the current speed. Digit 2 to Digit 0 represents the value of results. Digit points are followed by each digit number.

With the limitation of number of digit, the results should be round off. Therefore, we provided the calculation algorithm in our User Guide, see in Appendix 7.1. If the fractional part of the result is greater than approximate to 0.6, it will be viewed as 1 otherwise it will be viewed as 0. This algorithm also implemented in calculating the wheel size.

2.3 Synchronisation

In this project, all clock for each gate should be synchronised to the global clock. Regarding to the reset, the active-low reset should be synchronised to clear flip-flop.

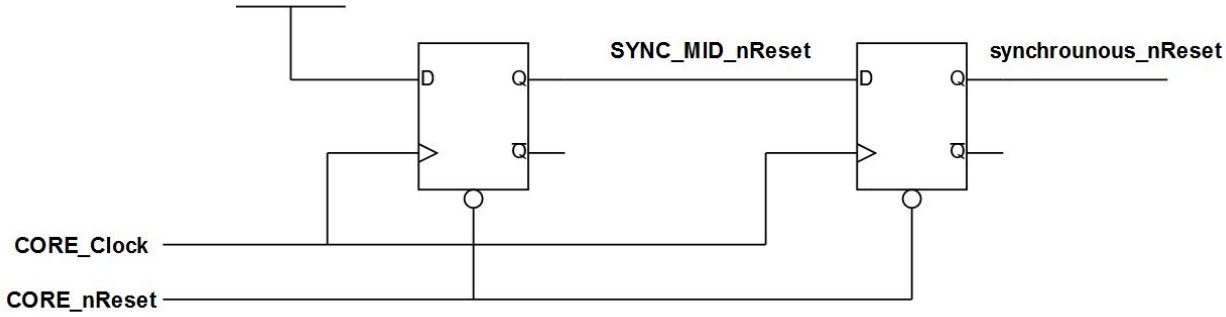

Figure 2: Synchronisation Circuit

According to the circuit, the synchronisation code can be written as

```
1   DFC1 SYNC_DFC1_1 (
2  .D('1'), .Q(SYNC_MID_nReset), .C(CORE_Clock), .RN(
       CORE_nReset)
3 );
4
5 DFC1 SYNC_DFC1_2 (
6    .D(SYNC_MID_nReset), .Q(synchronous_nReset), .C(CORE_Clock), .RN(
       CORE_nReset)
7 );
```

where the CORE_Clock is global clock at 12.8kHz, the CORE_nReset is applied to all gates. The first d-type flip flop is connected to high input signal. The output reset signals can be synchronised by adding these two flip-flops.

3 DESIGN

To achieve the objectives. The hard-only method was used to design some basic functions with small chip area. This cycle computer was designed to realize five basic functions to the cyclist:

- Cycling Distance - Odometer
- Cycling Time - Trip Timer
- Real-time Cycling Speed - Speedometer
- Pedal Cadence - Cadence
- Variable Wheel Size

This cycle computer was implemented on a single chip using AMS 0.35 um CMOS technology. All results can be displayed on the seven-segment LED screen. To achieve the objectives, the architecture diagram was drawn to set input signals, output signals and the logic for different models.

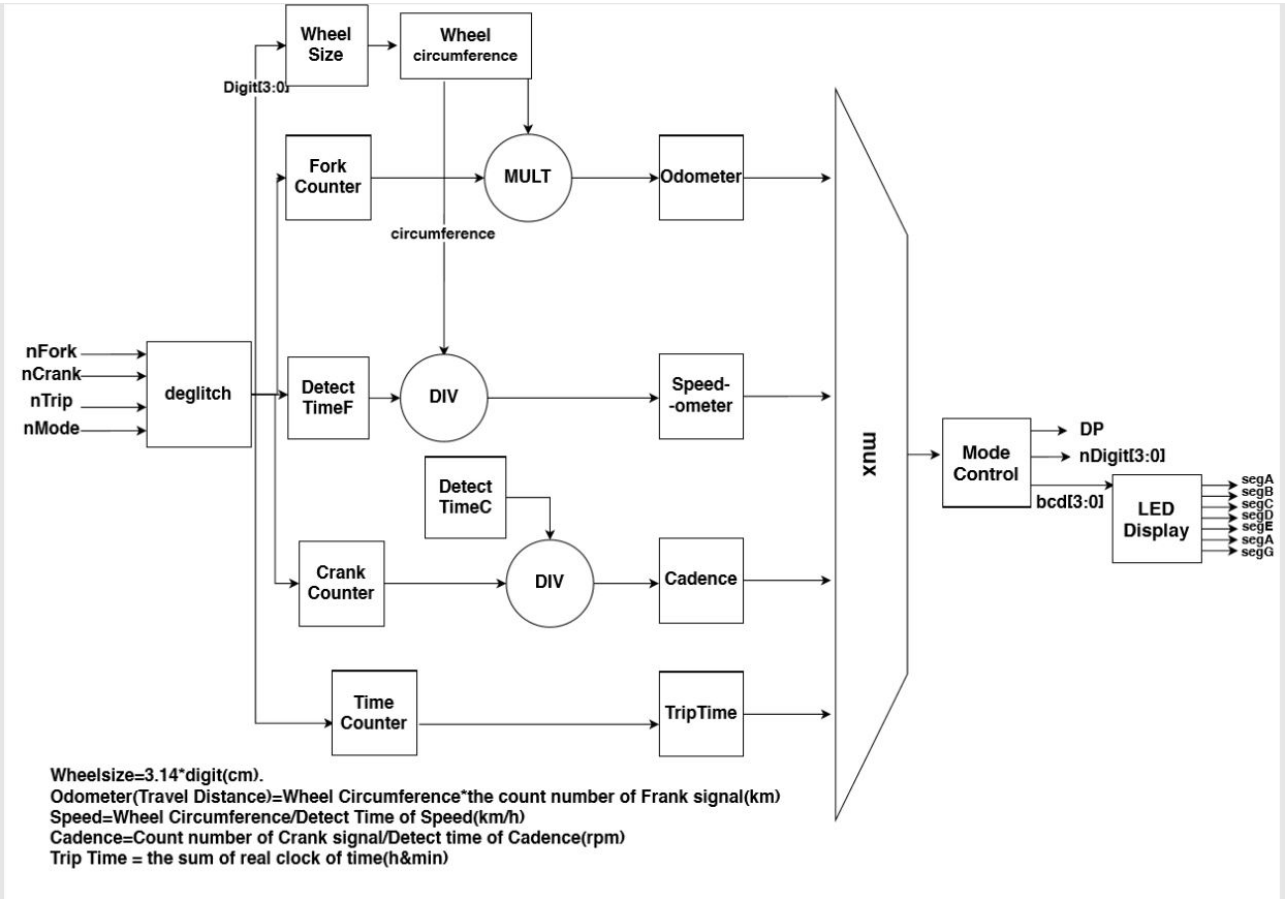

Figure 3: Architecture Diagram for cycle computer

Figure 3 described the architecture of this cycle computer, all models are written in systemverilog code. This architecture diagram displayed the input signal, output signal, five functions, mode control and LED display model, which corresponding to the objectives.

3.1 Behavioural Model

The behavioural model can be divided into parts:

1. Hall-effect signal detection Model, including Fork and Crank signal7.
2. Function models: Five basic functions, including odometer, speedometer, cadence, trip timer and variable wheel size.
3. Compute model: Evaluate the results, specifically, the divider is the essential part for calculating the results.
4. Mode control: Decide which function is using.
5. Decoder: Transfer the input signal (BCD) into seven-segment.
6. LED display: Display results on LED screen
7. deglitch model: Remove glitch when press the buttons.

3.2 RTL

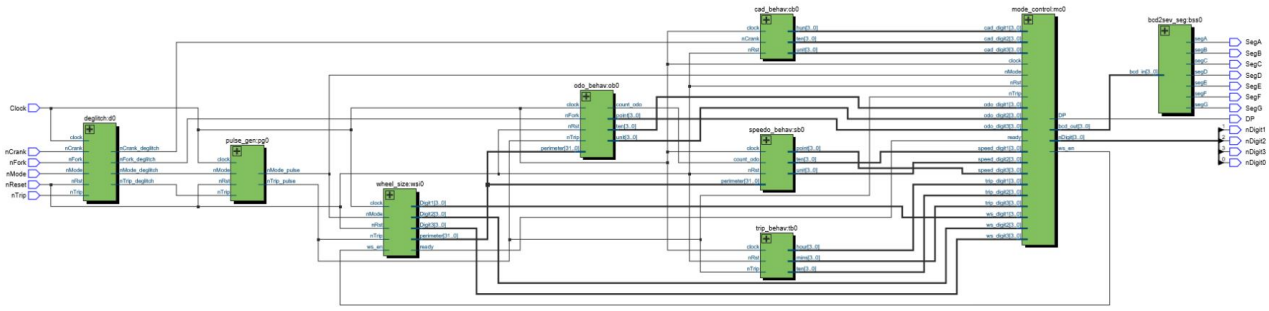

Figure 4: Register Transfer Level

3.3 comp_core

The comp_core is a top-level model which include all models in our design.

```
1 module comp_core (input logic Clock,nReset,nMode,nTrip,nFork,nCrank,
2 output logic SegA,SegB,SegC,SegD,SegE,SegF,SegG,DP,
3 output logic [3:0] nDigit);
```

Regarding to the input signals, Clock and synchronised Reset signals are provided for our design. Mode and Trip are two buttons. Fork and Crank signal is detected from two hall-effect sensors. the character n means it is a active-low signal. The output signals are compromised to seven-segment (SegA to SegG), four digits (nDigt[3:0]) and Digit Point (DP).

The top-level model also connected to other models, the model and its signals are listed below:

```
1  deglitch d0(.nCrank(sync_nCrank_2),.nFork(sync_nFork_2),.nTrip(
       sync_nTrip_2),.nMode(sync_nMode_2),.clock(Clock),.nRst(nReset),.
       nCrank_deglitch(nCrank_deglitch),.nFork_deglitch(nFork_deglitch),.
       nTrip_deglitch(nTrip_deglitch),.nMode_deglitch(nMode_deglitch),.
       readyM(readyM),.readyT(readyT));
2
3  pulse_gen pg0(.nTrip(nTrip_deglitch),.nMode(nMode_deglitch),.clock(
       Clock),.nRst(nReset),.readyM(readyM),.readyT(readyT),.nTrip_pulse(
       nTrip_pulse),.nMode_pulse(nMode_pulse));
4
5  speedo_behav sb0(.perimeter(perimeter),.clock(Clock),.nRst(nReset),.
       nFork(nFork_deglitch),.speedo_point(speedo_point),.speedo_int(
       speedo_int));
6
7  odo_behav ob0(.clock(Clock),.nFork(nFork_deglitch),.nRst(nReset),.nTrip
       (nTrip_pulse),.perimeter(perimeter),.odo_int(odo_int),.odo_point(
       odo_point));
8
9  bcd2sev_seg bss0(.bcd_in(bcd_out),.segA(SegA),.segB(SegB),.segC(SegC),.
       segD(SegD),.segE(SegE),.segF(SegF),.segG(SegG));
10
11 mode_control mc0(.odo_int(odo_int),.speedo_int(speedo_int),.cad_final(
       cad_final),.mins(mins),.odo_point(odo_point),.speedo_point(
       speedo_point),.hour(hour),.ws_digit1(ws_digit1),.ws_digit2(ws_digit2
       ),.ws_digit3(ws_digit3),.clock(Clock),.nRst(nReset),.nMode(
       nMode_pulse),.nTrip(nTrip_pulse),.ready(ready),.bcd_out(bcd_out), .
       nDigit(nDigit),.ws_en(ws_en),.DP(DP));
12
```

```
13 wheel_size wsi0(.clock(Clock),.nRst(nReset),.nTrip(nTrip_pulse),.nMode(
       nMode_pulse),.ws_en(ws_en),.Digit1(ws_digit1),.Digit2(ws_digit2),.
       Digit3(ws_digit3),.perimeter(perimeter),.ready(ready));
14
15 trip_behav tb0(.clock(Clock),.nRst(nReset),.nTrip(nTrip_pulse),.nFork(
       nFork_deglitch),.hour(hour),.mins(mins));
16
17 cad_behav cb0(.nCrank(nCrank_deglitch),.clock(Clock),.nRst(nReset),.
       cad_final(cad_final));
```

3.4 Model Control

To control each operation of this cycle computer, two buttons are provided, including Mode and Trip. This cycle computer can support 4 modes:

- Mode 0: Distance
- Mode 1: Duration
- Mode 2: Cadence
- Mode 3: Speed

Initially, Mode 0 was entered on reset. When we press trip button, the cycle computer is reset. When we press the mode button, the mode can be changed in the sequence of mode 0 to mode 3. When we press mode and trip button simultaneously, the wheel size can be changed.

3.5 Functions

In our design, five functions are designed in behavioural model of odo_behav, trip_behav, cad_behav, speedo_behav,and wheel_size, which corresponding to the odometer, trip timer, cadence, speedometer and variable wheel size respectively.

3.5.1 Wheel size

```
1 assign pi = {8'b00000000,24'b110010001111010111000010}; //value of pi
2 perimeter = {10'b0,22'b0000000010001011111111};//default 2136mm
3 dia = 32'b0;
```

```
peri = 64'b0;
if(ready)
begin
dia = {18'b0,digit1_tot} + {18'b0,digit2_tot} + {18'b0,digit3};
peri = dia * pi;//optimization
perimeter = peri [53:22];
end
```

The perimeter of the wheel is default at 2136mm. When changing the wheel size, the perimeter of the wheel size can be calculated as the diameter multiply the constant π.

3.6 Deglitch

The glitch may occurs when user press the buttons, therefore, the deglitch is provided to solve this problem.

3.7 Pulse Generation

After deglitching signals, the input button signals Mode and Trip are generated by the behavioural model pulse_gen.

3.8 Decoder

To transfer the input signals into seven-segment LED screen, the decoder is provided as behavioural model bcdsev_seg.

4 IMPLEMENTATION

Eventually, this cycle computer design has implemented successfully on a AMS 0.35um CMOS chip.

4.1 Synthesis Results

4.1.1 Area

```
Number of ports:                    2162
Number of nets:                     6620
```

```
Number of cells:                         3738
Number of combinational cells:           3083
Number of sequential cells:               603
Number of macros/black boxes:              21
Number of buf/inv:                        807
Number of references:                      10

Combinational area:               311893.401901
Buf/Inv area:                      31777.201035
Noncombinational area:            239657.603546
Macro/Black Box area:             714840.000000
Net Interconnect area:             94887.000000

Total cell area:                 1266391.005447
Total area:                      1361278.005447
```

4.1.2 Power

```
   Global Operating Voltage = 3
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 = 1pW


  Cell Internal Power  =   7.1546 uW   (62%)
  Net Switching Power  =   4.3605 uW   (38%)
                         ---------
Total Dynamic Power    =  11.5151 uW  (100%)

Cell Leakage Power     =  12.3340 uW
```

```
                 Internal          Switching          Leakage
                                  Total
Power Group      Power             Power              Power
                 Power    (   %   )  Attrs
--------------------------------------------------------------------------------

io_pad           2.1418e-03        1.1698e-03         1.6600e+06
    4.9716e-03  (  20.85%)
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         3.2779e-03        1.2750e-03         4.2653e+06
    8.8182e-03  (  36.97%)
sequential       0.0000            0.0000             0.0000
                0.0000   (   0.00%)
combinational    1.7349e-03        1.9157e-03         6.4087e+06
    1.0059e-02  (  42.18%)
--------------------------------------------------------------------------------

Total            7.1546e-03 mW     4.3605e-03 mW      1.2334e+07 pW
    2.3849e-02 mW
```

4.1.3 Timing Report

```
  data arrival time
     66.07

  clock master_clock (rise edge)                        78125.00
     78125.00
  clock network delay (ideal)                               2.50
     78127.50
```

```
5   clock uncertainty                                    -1.00
        78126.50
6   core_inst/sb0/divd_reg[49]/C (DFSEC1)                 0.00
        78126.50 r
7   library setup time                                   -0.78
        78125.72
8   data required time
        78125.72
9   ------------------------------------------------------------------------------


10  data required time
        78125.72
11  data arrival time
        -66.07
12  ------------------------------------------------------------------------------


13  slack (MET)
        78059.65
```

4.2 Place and Route

The place and route result can be see in Appendix 7.2.

4.3 Full Chip Design

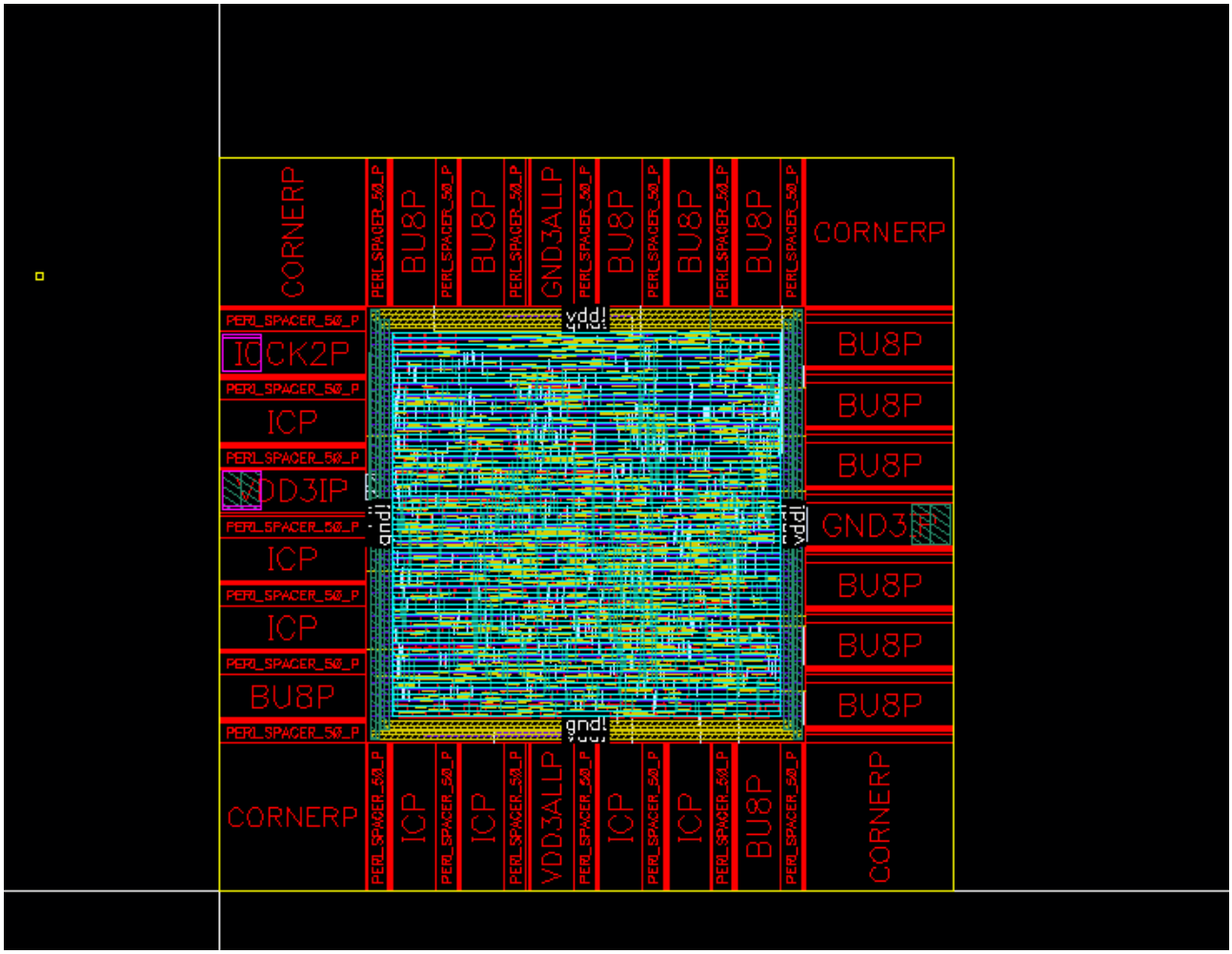

Figure 5: DRC Check in full chip

X dimension in the Virtuoso:1693.725nm

Y dimension in the Virtuoso: 1687.2nm

Total are:2857652.82nm^2

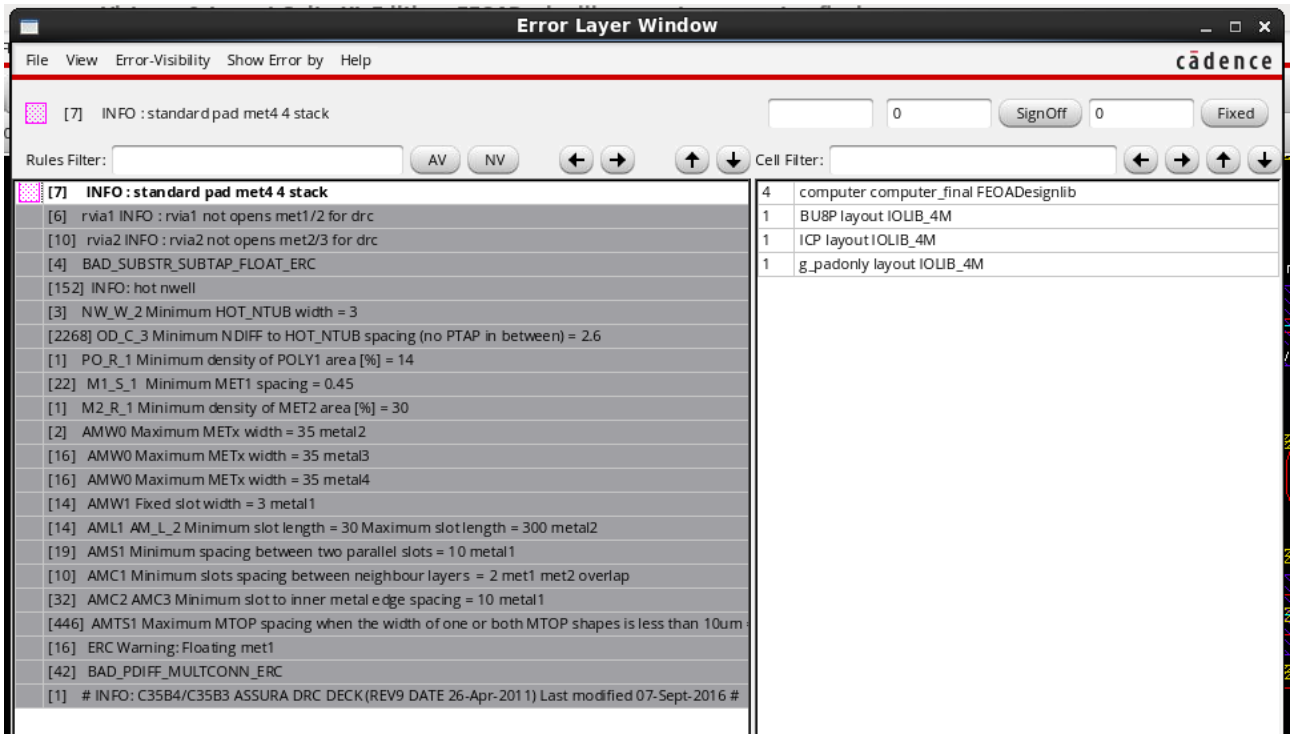

Figure 6: DRC Error List

The virtuoso cadence is used to check the design rules. According to the Figure 6, there is no error when checking design rules but it has several warnings.

5 TESTING STRATEGY AND RESULTS

All functions and performance of cycle computer design were verified. The results can be displayed on LED screen only with 4 digits (Digit 0- Digit 3) and digit points. To test different functions, the stimulus.sv model has been changed to stimulus mode and trip signals for cycle computer.

Use following command to test 100 seconds whether 4 modes can be controlled by two buttons in sequence.

./simulate behavioural 100s +define+basic_mode_change
and in gate-level design:
./simulate -gate gate_level behavioural 100s +define+basic_mode_change

To add stimulus signal and display the results of this cycle computer, the command is provided as

./simulate -gate gate_level behavioural 100s +define+special_stimulus +define+special_monitor

To add delay for each gate, the cycle computer can be simulated as
./simulate -gate -sdf gate_level/computer.sdf gate_level 100s +define+basic_mode_change

In the waveform, some unused waves were removed for better observation, including Button3, DnC, SCLK, SDIN, nButton3, nSCE, press_third_button.

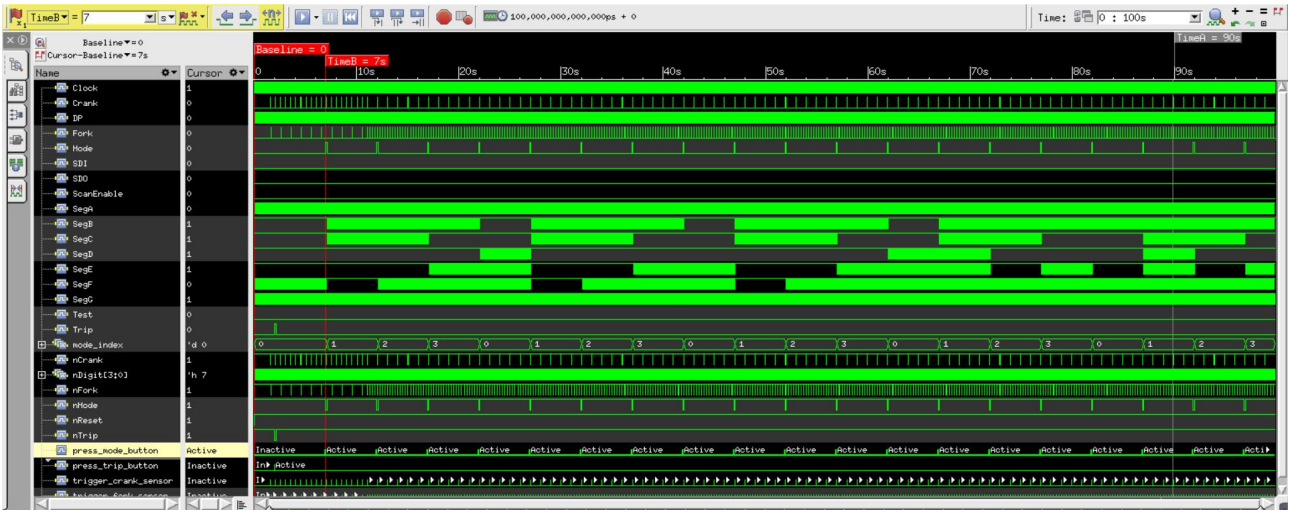

Figure 7: Mode Change

According to the figure, it is can be found that when press_mode_button is active, the nMode becomes to low then to high, the mode_index is changed from 0 to 1 hence the mode is changed from MODE0 to MODE1. Furthermore, once nMode is changed, the mode_index is changed for loop from 0 to 3.

5.1 Character verification

The Digit3 displays the character that represent each function, including d, t, c, v.

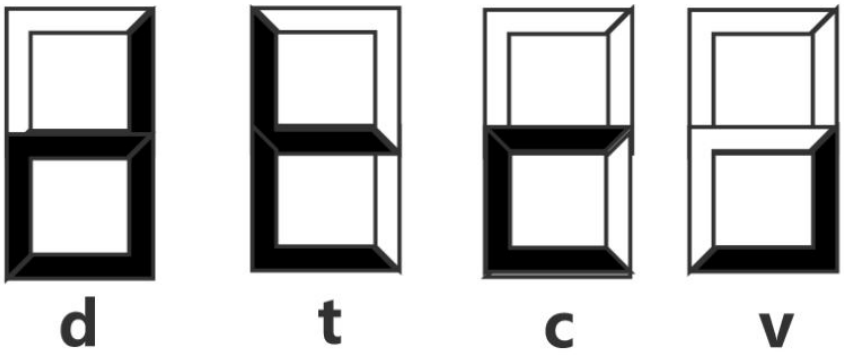

Figure 8: The characterS represent each function

To verify the character, the waveform can be observed in seven-segment (SegA-SegG). For example, the character t can be read from simulation.

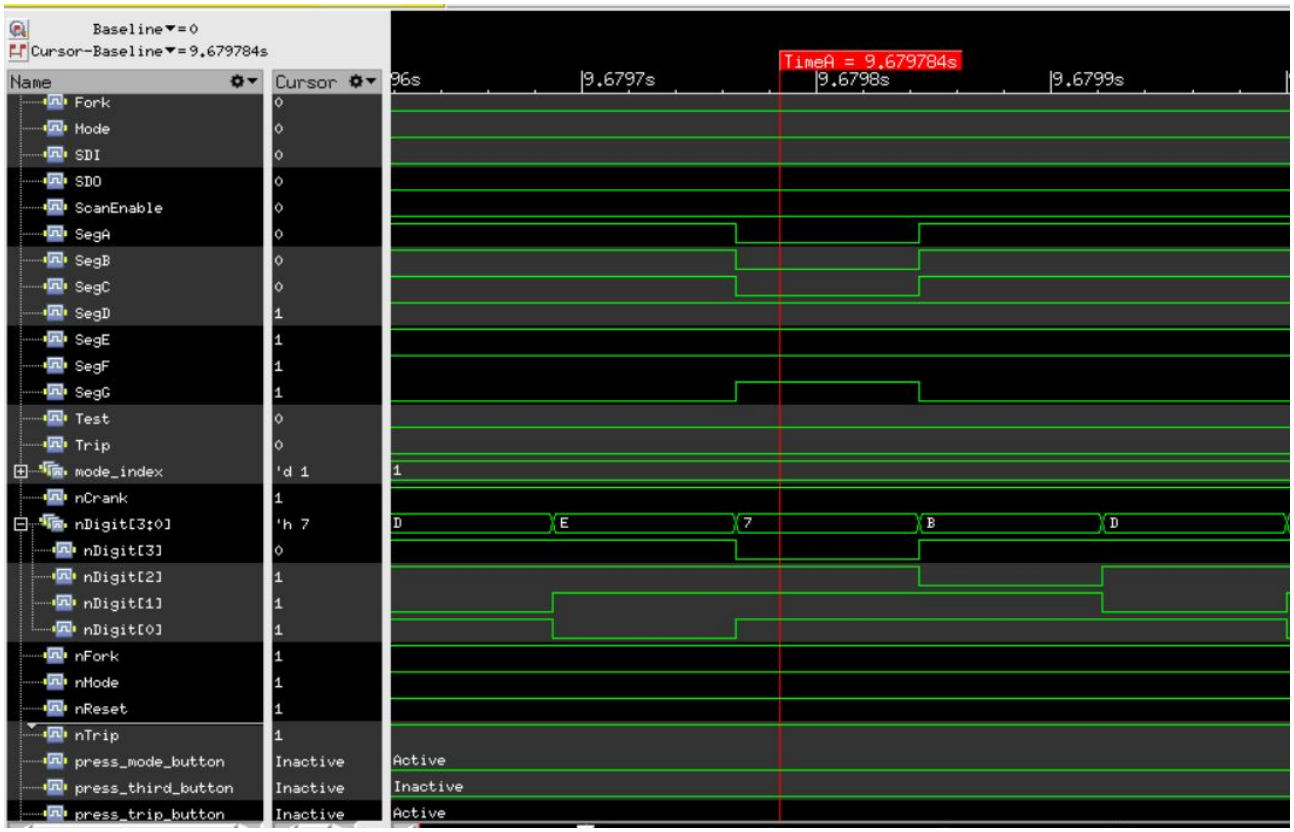

Figure 9: Character t

According to the TimeA cursor,the mode_index is 1, which means it is in Mode1. Then the nDigit[3:0] is 'h7, which means nDigit[3] is 0, others are 1. The digit3 is active-low digits

hence we can read the seven-segment in Digit3. Afterwards, we can observe the SegA to SegG, in this waveform the SegA to SegC are 0, SegD to SegG are 1, the 1 represent highlight in seven-segment LED. Therefore, the t is displayed as Figure 8. Similarly, other characters can be observed in different modes.

6 EVALUATION

Our cycle computer design is excellent in terms of its performances. All functions including odometer, speedometer, cadence and trip timer were performed well with correct simulation values. Only small derivation occurred due to the accuracy related to the limit digits on LED screen. After gate-level design and place and route, the full chip was implemented without design rule violations. However, sign-off part was not finished with 7 warnings.

7 CONCLUSIONS AND FUTURE WORK

To summarise, this cycle computer was designed to achieve all functions in our design proposal. All functions were simulated with correct result values. To implement this design on a chip, AMS 0.35um CMOS technology was used. The synthesis and place and route was complete without design rule violations. Consequently, the full chip design was completed successfully. To improve this design, in the future, some additional functions can be added, such as the maximum speed. Additionally, LCD may be more helpful for user to read the result values since it can provide more accurate number with unit. Furthermore, the chip area might be reduced by simplifying the system verilog code in behavioural model or optimizing the design in synthesis and place and route.

REFERENCES

[1] Iain McNally,(Apr,2018),Reset Synchronization, [Online]. Avaliable: https://secure.ecs.soton.ac.uk/notes/bim/notes/chip/lab/synchronization/synchronization.php

APPENDIX

7.1 User Manual

A user guide is provided for user to understand the specifications of cycle computer, operations and calculation of each function.

1 FUNCTION DESCRIPTION

This cycling computer features tracking cycling moment accurately and precisely. To meet the most essential need of cyclist, it records 4 kinds of data, including travelling distance, travelling duration, cadence and speed. In addition to the basic functions, there is a extra function to change the wheel size. This cycling computer is easy to use with one LED screen and two buttons on it. All tracking data can be displayed on the LED screen.

There are totally five functions in this cycling computer:

- Odometer
- Tripe Timer
- Speedometer
- Cadence Meter
- Variable Wheel Size

2 COMPONENTS

- LED Screen
- Mode changing Buttons
- Two DRV5021 Hall Effect Sensors and magnet

3 INSTALLATION STEPS

1. Mount the magnet as close in the hub.
2. Attach the sensor to the **Fork**.
3. Attach the sensor to the **Crank**.
4. Mount the cycle compute to the handlebar

4 BUTTON OPERATION INSTRUCTION

In this cycle computer design, two buttons are provided: Mode and Trip. Four types of information can be displayed on the LED screen, including travel distance, travel time, pedal cadence and current speed.

- Press **Mode** button to display information in the sequence of:
    1. d-Travel Distance
    2. t-Travel Duration
    3. c-Pedal Cadence
    4. v-Current Speed
- Press **Trip** button to reset the values.
- Press **Mode** and **Trip** at the same time to set the wheel size, or you can
- Press **Mode** button, hold on **Mode** button, then press **Trip** button to set the wheel size

5 LED DISPLAY SPECIFICATION

There are four digits on LED screen, while the digit3 display the character and digit0-digit2 display its values.

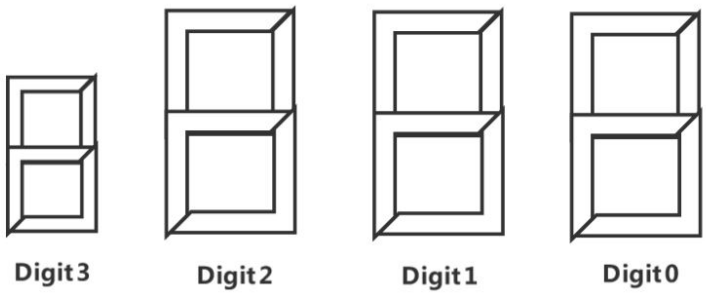

Figure 1: LED Display

The digit3 can display characters d, t, c, v indicate the different functions.

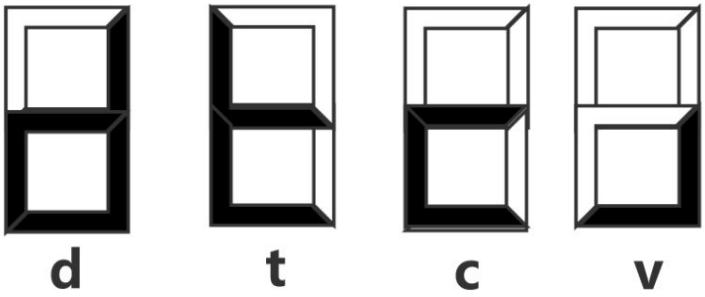

Figure 2: Digit3 Character Display

5.1 Odometer-d

- Function: Travel Distance
- Digit3 character display: d
- Unit: kilometres(km)
- Range: 0-99.9km
- Accuracy: Keep 1 decimal to 0.1km.

5.2 Trip Timer-t

- Function: Duration
- Digit3 character display: t;
  Digit2 display: Time in hour;
  Digit1Digit0 display: Time in minutes.
- Unit: hour (h) and minutes(min)
- Range: 0-9h59min
- Accuracy: Keep the integers to one minute.

5.3 Cadence meter-c

- Function: Pedal Cadence.
- Digit3 character display: c

- Unit: resolutions per unit (rpm)
- Range: 0-999rpm
- Accuracy: Keep the integers to one rpm.

5.4 Speedometer-v

- Function: Current Speed
- Digit3 character display: v
- Unit: kilometres per hour (km/h)
- Range: 0-99.9km/h
- Accuracy: Keep 1 decimal to 0.1km/h.

5.5 Wheel size

- Function: set wheel diameter
- Unit: millimeter (mm)
- Range: 0-999mm
- Accuracy: Keep the integers to one mm.

6 CYCLE COMPUTER PARAMETER CALCULATIONS

In this cycle computer, two sensors are provided to detect the **Frank** and **Crank** signal. Each parameter can be calculated in terms of the count number of the signal detecting.

- Travel Distance=(the count number of the **Frank** signal)*(Wheel Circumstance)
- Travel Duration = Time between the first and the last **Frank** signal detecting during cycling.
- Current Speed = (Wheel Circumstance)/ (Time between two adjacent **Frank** signals detecting).
- Cadence = (1 minute)/(the time between two adjacent **Crank** signal detecting).

- Wheel circumstance =3.14 * (Wheel diameter), where 3.14 is the approximate value of π.

6.1 Accuracy

With the limitation of the number of digits on the LED screen, only three digits can be displayed. Therefore, rounding off the results has been implemented in this cycle computer. Specifically, LED displays each type of value with respect to the accuracy below:

- Distance: Accurate to 0.1km.
- Duration: Accurate to 1min.
- Cadence: Accurate to 1rpm.
- Speed: Accurate to 0.1km/h
- Wheel diameter: Accurate to 1mm.

Regarding the results with accurate to 0.1, the number less than 0.06 is treated as 0.0 while the number greater than or equal to 0.06 is treated as 0.1, that is

$$Result(Distance, Speed) = \begin{cases} 00.0, & Result < around 0.06 \\ 00.1, & Result \geq around 0.06 \end{cases} \tag{1}$$

Similarly, regarding to the results with accurate to 1 unit, the LED displays as:

$$Result(Duration, Cadence) = \begin{cases} 000, & Result < around 0.6 \\ 001, & Result \geq around 0.6 \end{cases} \tag{2}$$

Apart from the LED display accuracy, another approximate value for results is due to the wheel size. When user set the wheel size, only three digits are provided. In this case, user can only input the value from 0 to 999mm for the wheel diameter.Since the wheel diameter may have decimal part, the wheel circumstance calculation result may not too precise hence the results of four functions(odometer, trip timer, cadence meter and speedometer) may not precise as well.

6.2 Wheel Size Setting Table

Press **Mode** and **Trip** button simultaneously to alter the wheel diameter, **or** you can Press **Mode** button, hold on **Mode** button, then press **Trip** button to set the wheel size

- Press MODE button: set the position of **Digit 2**.
- Press TRIP button: Increment digit value (0-9).
- Press MODE button: set the position of **Digit 1**.
- Press TRIP button: Increment digit value (0-9).
- Press MODE button: set the position of **Digit 0**.
- LED displays the input value of wheel diameter.

The default wheel size is 700*28c with circumference of 2136mm, while the size of tire is 700mm, the width of tire is 28mm and the width of rim is c(A is very narrow, D is wide). The diameter can be set from 0 to 999 to change the wheel size. The wheel size number will display on the LED screen when you set the wheel diameter.

$$Wheel\_Circumstance = \pi * Wheel\_Diameter \tag{3}$$

- Adjust Wheel Size: Press Mode and Trip buttons.
- Increment Digit: Press Trip Button.
- Adjust Next Digit: Press Mode button (Digit2->Digit1->Digit0).

Wheel Specification	Wheel Circumstance	Wheel Diameter
700*38c	2180mm	694mm
700*35c	2168mm	690mm
700*32c	2155mm	686mm
700*30c	2145mm	683mm
700*28c	2155mm	680mm
700*25c	2124mm	676mm
700*23c	2105mm	670mm
700*20c	2074mm	661mm
650*23c	1990mm	634mm
700*20c	1945mm	619mm

Table 1: Wheel Size Setting Specification Table

7.2 Circuit Diagram

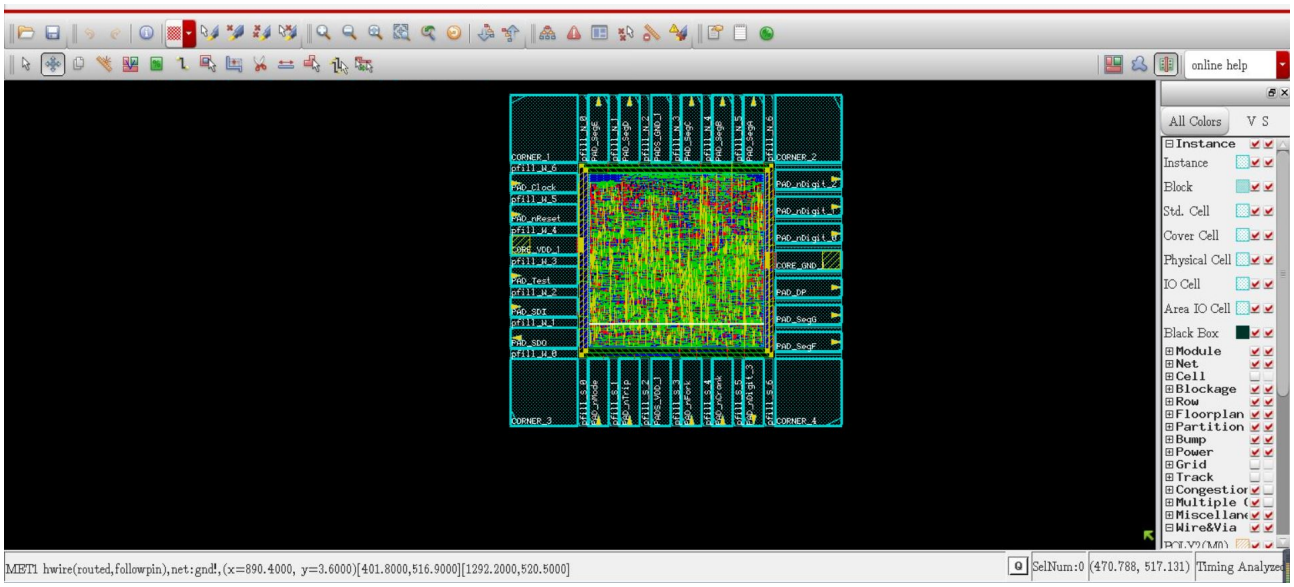

Figure 10: Place and Route Result