# Parking Spot Finder
# Final Report

Yitong Ding, Kaijie Shen, Qiankun Xie, Yingjie Zhou

December 7, 2015

# Contents


**1 Introduction** **4**
1.1 Problem . . . . . 4
1.2 Solution . . . . . 4

**2 Proposed System** **4**
2.1 Overview . . . . . 4
2.2 functionality and user interface . . . . . 5
2.2.1 Shortest Path Calculation . . . . . 5
2.2.2 Real-time Location Calculation . . . . . 7
2.2.3 user interface . . . . . 8
2.3 design parameters . . . . . 9
2.4 external connected devices . . . . . 9
2.4.1 Accelerometer . . . . . 9
2.4.2 Gyroscope . . . . . 10
2.4.3 LED Display . . . . . 12
2.5 I/O interface . . . . . 12
2.6 Storage: memory size and interfaces . . . . . 12
2.7 Memory Model Implementation . . . . . 15
2.8 Accelerator to Memory Interfaces . . . . . 16

**3 Design Exploration** **16**
3.1 Computational Workload Comparison . . . . . 16

**4 Test Bench Development** **17**
4.1 How To Set Up . . . . . 17
4.2 Behavioural Model for External Devices . . . . . 18
4.3 Debug Experience . . . . . 18

**5 Simulation Results** **19**
5.1 Design Compiler Synthesis Reports . . . . . 19
5.2 Encounter Place and Route Reports . . . . . 20
5.3 simulation screen capture and explanation . . . . . 23

**6 Conclusion and Future Work** **23**

**7 Individual Contribution** **24**

**8 Appendix** **25**
8.1 C code and script for accelerator design . . . . . 25
8.2 C code for programs running in M0 . . . . . 31
8.3 System Wraper File . . . . . 32
8.4 Vivado scripts . . . . . 38
8.5 design compiler and encounter scripts . . . . . 38
8.6 test bench files . . . . . 42
8.7 Makefile . . . . . 44

# 1 Introduction

## 1.1 Problem

In cities, one problem that begs to be solved is how to find a parking spot during garage parking quickly and easily, especially when the parking garage is almost full and only a few extremely elusive place left. Do you feel so hard to navigate the maze of parking lot and finally find somewhere to park your car? Do you feel so helpless and frustrated to drive aimlessly in a park lot after 10 minutes parking place search and get nowhere? Do you have a bad impression on a shopping mall and will not go there again because of the terrible parking experience? For customers, it's time, gas, money waste and even leading to a bad mood for the whole day. For merchants, it is possible to lose some part of customers that means loss of money.

## 1.2 Solution

Our device is a small portable garage parking spot finder, which is aimed to make divers' parking more convenient. It can help you locate an available parking spot and do real-time path calculation. To be specific, when a car enter the parking garage, at each corner the small device can tell the driver to turn left or right, and finally lead the diver to a parking place. Besides, based on the same mechanism when drivers finish shopping, this device can also help drivers to locate and find their cars.

# 2 Proposed System

## 2.1 Overview

This device looks like a small cube with a small LED display that is used to show the direction (left arrow, right arrow, straight arrow) at each corner inside the parking garage. It also has a battery charger interface that the driver can even charge it inside their car. On one side of that cube, we use antiskid material so that drivers can easily put it above panel without slip. The main function of this device is lead drivers to a parking place in a parking garage by showing different direction arrows on the LED display. Besides the cube can save divers' information inside, like access permit, parking pay account information, in this case the cube also can replace traditional parking

card and save more time and effort when enter and leave a parking garage with these advanced functions: 1) Access control card system (put these device inside drivers' car, toll gate will automatically open if drivers have access permit to this building) 2) Automatic park fee pay system (the cube will record the time of entrance then when the car leave it can calculate the total parking fee based on the parking time and charge the fee from the drivers' parking account). 3) The driver can check parking account balance on the LED display.

## 2.2 functionality and user interface

### 2.2.1 Shortest Path Calculation

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest path tree.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. In our project, the parking lot is like a graph. With a fixed entrance and fixed destination, we can use this algorithm to find a shortest path.

For example, in the graph below, we need to find a path from point a to point f. Using Dijkstra's algorithm, we can get a shortest path: $a => d => g => h => f$.

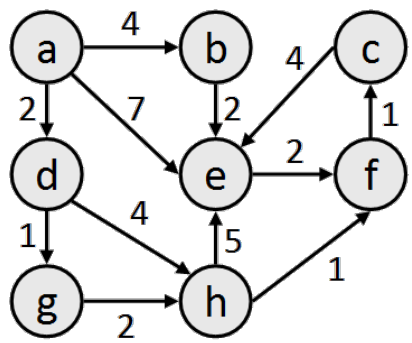



Figure 1: Example of Dijkstra's algorithm

We use matrix to simulate the map of parking lot like figure below.

$$\begin{bmatrix} \infty & 3 & 5 & \infty & 8 & 1 & \infty & \infty \\ 3 & \infty & 2 & \infty & \infty & \infty & 1 & \infty \\ 5 & 2 & \infty & 1 & \infty & \infty & \infty & 2 \\ \infty & \infty & 1 & \infty & 4 & \infty & \infty & \infty \\ 8 & \infty & \infty & 4 & \infty & 6 & \infty & 1 \\ 1 & \infty & \infty & \infty & 6 & \infty & 5 & \infty \\ \infty & 1 & \infty & \infty & \infty & 5 & \infty & 1 \\ \infty & \infty & 2 & \infty & 1 & \infty & 1 & \infty \end{bmatrix}$$

Figure 2: Matrix Example

Entry ij means the distance of point i and point j. Also, Entry ij and Entry ji will always have the same value.The infinity represents point i and point j are disconnected.

Finally, we use DFS to implement this algorithm.

```
function Dijkstra(Graph, source):
create vertex set Q

for each vertex v in Graph:              // Initialization
dist[v] <= INFINITY                    // Unknown distance from source to v
prev[v] <= UNDEFINED                  // Previous node in optimal path from source
add v to Q                             // All nodes initially in Q (unvisited nodes)
dist[source] <= 0                          // Distance from source to source

while Q is not empty:
u <= vertex in Q with min dist[u]     // Source node will be selected first
remove u from Q

for each neighbor v of u:             // where v is still in Q.
alt <= dist[u] + length(u, v)
if alt < dist[v]:                  // A shorter path to v has been found
dist[v] <= alt
prev[v] <= u
```

```
return dist[], prev[]
```

### 2.2.2 Real-time Location Calculation

Assume we have fixed entrance and fixed destination. So we can use the data from gyroscope and accelerometer to calculation the real-time location. Then we compare the real-time location with correct path to make direction decision while can show turn right, turn left and go straight.

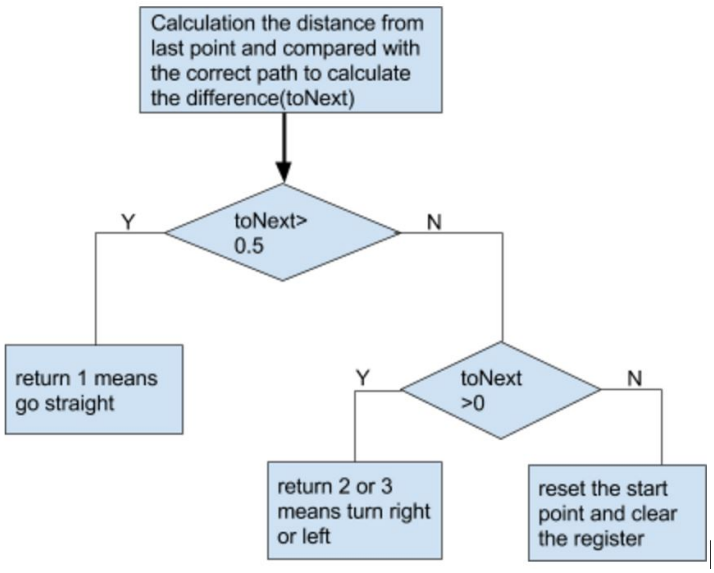



Figure 3: Real-time Location Calculation

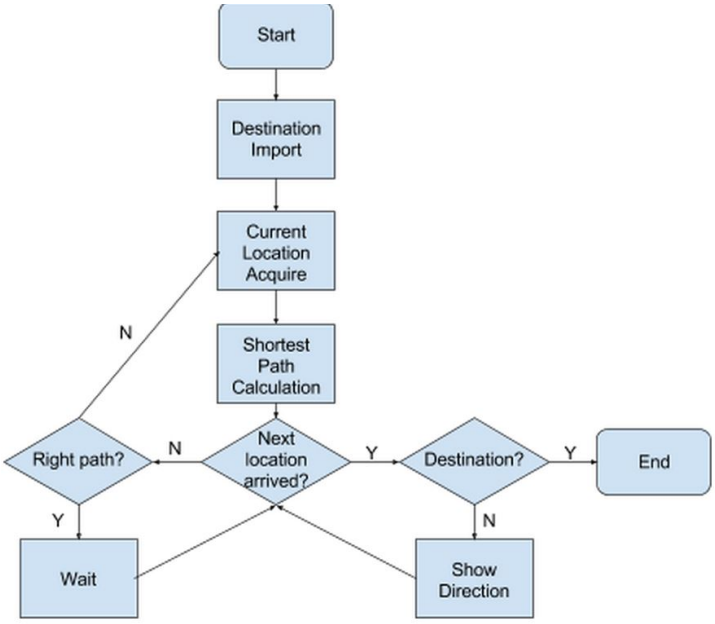



Figure 4: Operational Flow

### 2.2.3 user interface

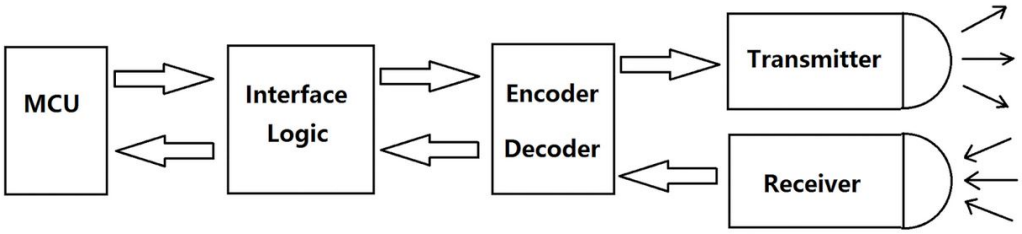



Figure 5: IR and NFC Interface

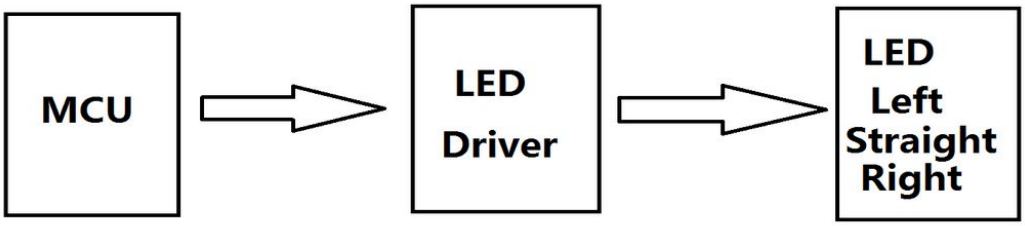



Figure 6: LED Interface

## 2.3 design parameters

- Clock frequency: 200 kHz
- Sensor data input frequency: 20 Hz
- Hardware Accelerator Compute frequency: 20Hz
- Memory size: $65536 * 32$ bits ($2^{18}$ bytes)

## 2.4 external connected devices

In this project, we need altogether three peripherals: accelerometer, gyroscope and LED display. The accelerometer provides the car's accelerator data so that we can achieve the distance the car moves. The gyroscope provides data to calculate the angle the car turns. Based on data of these two sensors, we can calculate the car's position. Finally, the LED display shows the direction the driver need to go: go straight, turn left or turn right.

### 2.4.1 Accelerometer

The accelerometer we choose is ADXL345, it is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to $\pm 16$ g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

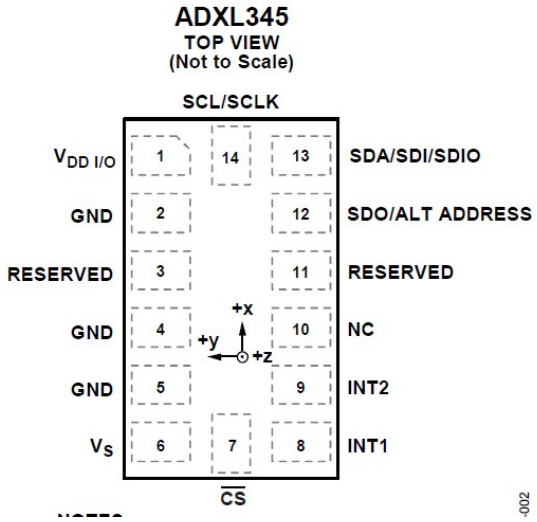



Figure 7: ADXL345 top view

Each 16-bit data were stored in two 8-bit registers. To achieve these data, we use I2C interface to read from certain registers.

This is part of the register map we can use to access the accelerator data:

| | | | | | |
|---|---|---|---|---|---|
| 0x32 | 50 | DATAX0 | R | 00000000 | X-Axis Data 0 |
| 0x33 | 51 | DATAX1 | R | 00000000 | X-Axis Data 1 |
| 0x34 | 52 | DATAY0 | R | 00000000 | Y-Axis Data 0 |
| 0x35 | 53 | DATAY1 | R | 00000000 | Y-Axis Data 1 |
| 0x36 | 54 | DATAZ0 | R | 00000000 | Z-Axis Data 0 |
| 0x37 | 55 | DATAZ1 | R | 00000000 | Z-Axis Data 1 |

Figure 8: ADXL345 register

In our project, the car only needs X-Axis data because the Y-Axis and Z-Axis data are always 0.

### 2.4.2 Gyroscope

The gyroscope we use is ITG-3200. It is a single-chip, digital-output, 3-axis MEMS gyro IC. It features three 16-bit analog-to-digital converters (ADC-s) for digitizing the gyro outputs, a user-selectable internal low-pass filter bandwidth, and an I2C interface.

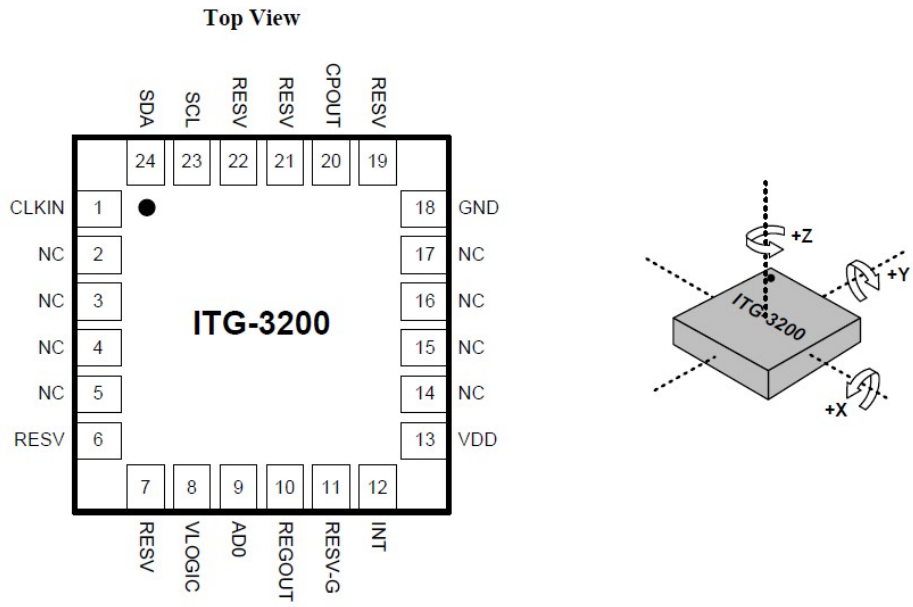



Figure 9: ITG-3200 top view

Similar to accelerometer, the gyroscope data are also stored in certain registers which can be accessed by I2C interface.

**Serial Interface**

| Pin Number | Pin Name | Pin Description |
|---|---|---|
| 8 | VLOGIC | Digital IO supply voltage. VLOGIC must be ≤ VDD at all times. |
| 9 | AD0 | $I^2C$ Slave Address LSB |
| 23 | SCL | $I^2C$ serial clock |
| 24 | SDA | $I^2C$ serial data |

Figure 10: ITG-3200 interface

| | | | | |
|---|---|---|---|---|
| 1D | 29 | GYRO_XOUT_H | R | GYRO_XOUT_H |
| 1E | 30 | GYRO_XOUT_L | R | GYRO_XOUT_L |
| 1F | 31 | GYRO_YOUT_H | R | GYRO_YOUT_H |
| 20 | 32 | GYRO_YOUT_L | R | GYRO_YOUT_L |
| 21 | 33 | GYRO_ZOUT_H | R | GYRO_ZOUT_H |
| 22 | 34 | GYRO_ZOUT_L | R | GYRO_ZOUT_L |

Figure 11: ITG-3200 register

In this project, we also only need the GYRO_XOUT data because the car only turn left or right in one plane.

### 2.4.3 LED Display

The LED display gives instruction to the driver what he/she should do next, whether turn left or right or just go straight on.

## 2.5 I/O interface

$I^2C$ (Inter-Integrated Circuit), pronounced I-squared-C, is a multi-master, multi-slave, single-ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors). It is typically used for attaching lower-speed peripheral ICs to processors and micro controllers. Alternatively $I^2C$ is spelled I2C (pronounced I-two-C) or IIC (pronounced I-I-C).

## 2.6 Storage: memory size and interfaces

Memory size is set to $65536 * 32$ bits, that is $2^{18}$ bytes. The size of memory is over designed for future use of updating algorithm that runs on Cortex M0 and updating of the graph of the parking lots, since it takes a lot of memory space to save 2 huge array of a parking lot.

Here, we use CoreAssembler to design the bus, peripherals and to synthesize the subsystem. The peripherals in our project are accelerometer, gyroscope and LED display, which all can transmitting and receiving data using I2C protocol, so the subsystem we design contain AHB, APB and I2C.

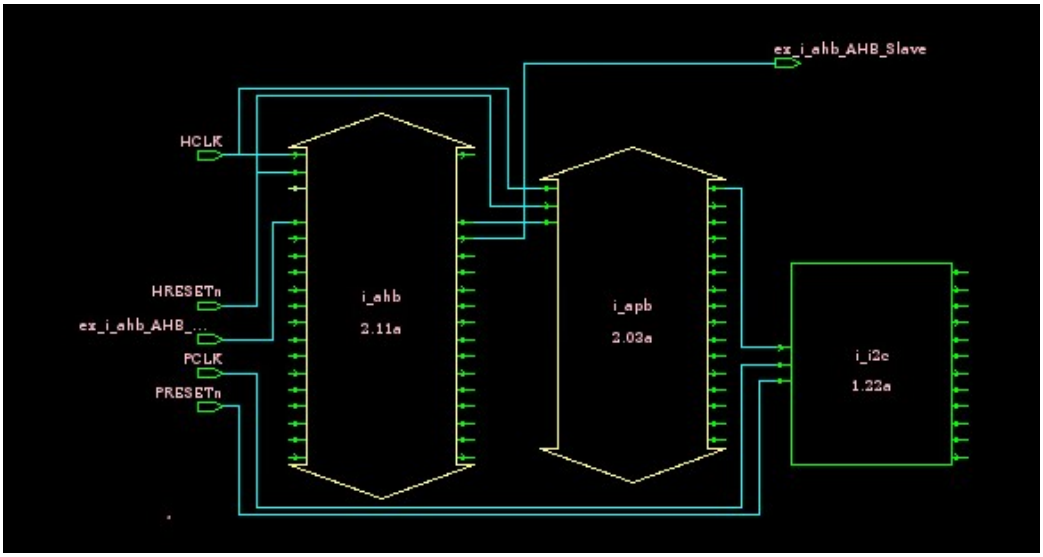



Figure 12: AMBA bus top view

The exported AHB master will be connected to M0, one AHB slave was connected to APB and another was exported for accessing memory.

All the peripherals don't need very high speed, so the I2C was set standard mode instead of fast mode or high speed mode.

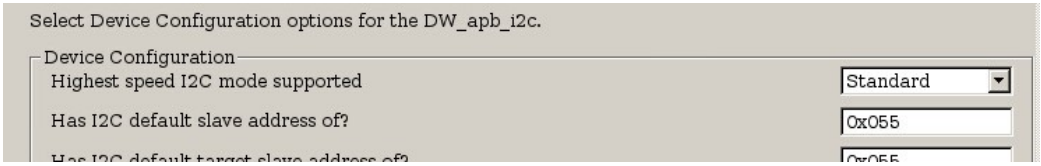



Figure 13: I2C mode

The ic_clk period was set 500 ns.

Figure 14: I2C clock

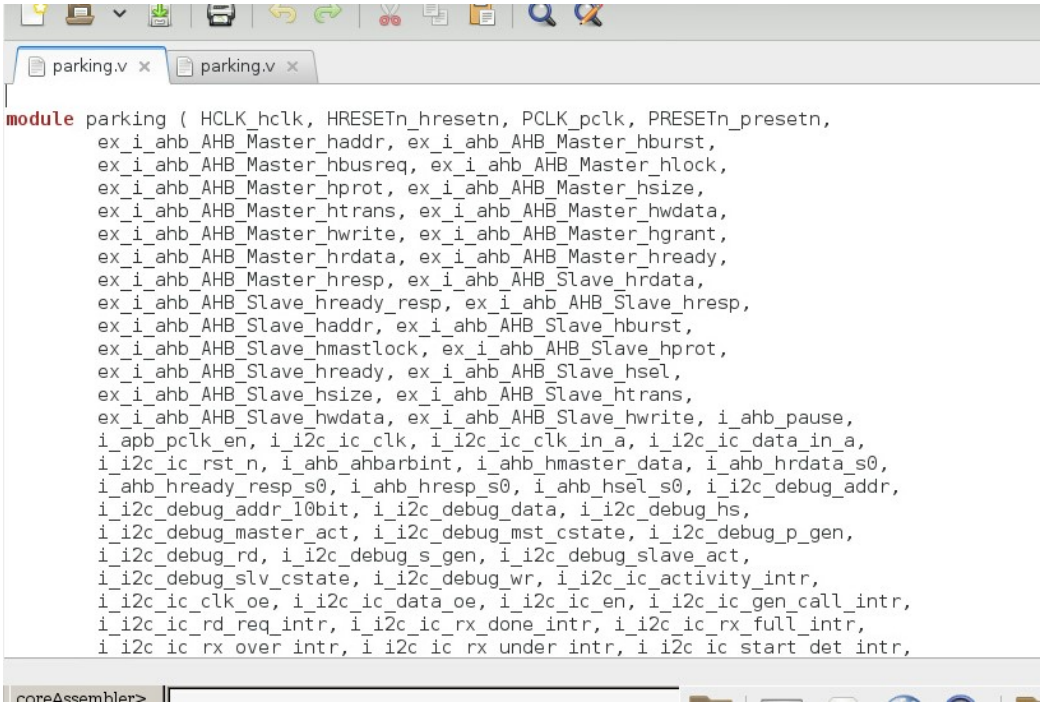
After specifying AHB, APB an I2C components, we have the subsystem RTL:

```
parking.v ×   parking.v ×
// This file was automatically generated by coreAssembler (version J-2015.03-SP1).
//
// FILENAME : /home/warehouse/shenkaijie/Documents/coreAssembler/design/parking/src/parking.v
// DATE : 11/28/15 20:39:05
// ABSTRACT : Verilog top-level subsystem RTL.
//

module parking (// Ports for Interface HCLK
                HCLK_hclk,
                // Ports for Interface HRESETn
                HRESETn_hresetn,
                // Ports for Interface PCLK
                PCLK_pclk,
                // Ports for Interface PRESETn
                PRESETn_presetn,
                // Ports for Interface ex_i_ahb_AHB_Master
                ex_i_ahb_AHB_Master_haddr,
                ex_i_ahb_AHB_Master_hburst,
                ex_i_ahb_AHB_Master_hbusreq,
                ex_i_ahb_AHB_Master_hlock,
                ex_i_ahb_AHB_Master_hprot,
                ex_i_ahb_AHB_Master_hsize,
                ex_i_ahb_AHB_Master_htrans,
                ex_i_ahb_AHB_Master_hwdata,
                ex_i_ahb_AHB_Master_hwrite,
coreAssembler>
```

Figure 15: AHB RTL

Finally, by using the technology library provided, the synthesized subsystem netlist was achieve:



```
module parking ( HCLK_hclk, HRESETn_hresetn, PCLK_pclk, PRESETn_presetn,
        ex_i_ahb_AHB_Master_haddr, ex_i_ahb_AHB_Master_hburst,
        ex_i_ahb_AHB_Master_hbusreq, ex_i_ahb_AHB_Master_hlock,
        ex_i_ahb_AHB_Master_hprot, ex_i_ahb_AHB_Master_hsize,
        ex_i_ahb_AHB_Master_htrans, ex_i_ahb_AHB_Master_hwdata,
        ex_i_ahb_AHB_Master_hwrite, ex_i_ahb_AHB_Master_hgrant,
        ex_i_ahb_AHB_Master_hrdata, ex_i_ahb_AHB_Master_hready,
        ex_i_ahb_AHB_Master_hresp, ex_i_ahb_AHB_Slave_hrdata,
        ex_i_ahb_AHB_Slave_hready_resp, ex_i_ahb_AHB_Slave_hresp,
        ex_i_ahb_AHB_Slave_haddr, ex_i_ahb_AHB_Slave_hburst,
        ex_i_ahb_AHB_Slave_hmastlock, ex_i_ahb_AHB_Slave_hprot,
        ex_i_ahb_AHB_Slave_hready, ex_i_ahb_AHB_Slave_hsel,
        ex_i_ahb_AHB_Slave_hsize, ex_i_ahb_AHB_Slave_htrans,
        ex_i_ahb_AHB_Slave_hwdata, ex_i_ahb_AHB_Slave_hwrite, i_ahb_pause,
        i_apb_pclk_en, i_i2c_ic_clk, i_i2c_ic_clk_in_a, i_i2c_ic_data_in_a,
        i_i2c_ic_rst_n, i_ahb_ahbarbint, i_ahb_hmaster_data, i_ahb_hrdata_s0,
        i_ahb_hready_resp_s0, i_ahb_hresp_s0, i_ahb_hsel_s0, i_i2c_debug_addr,
        i_i2c_debug_addr_10bit, i_i2c_debug_data, i_i2c_debug_hs,
        i_i2c_debug_master_act, i_i2c_debug_mst_cstate, i_i2c_debug_p_gen,
        i_i2c_debug_rd, i_i2c_debug_s_gen, i_i2c_debug_slave_act,
        i_i2c_debug_slv_cstate, i_i2c_debug_wr, i_i2c_ic_activity_intr,
        i_i2c_ic_clk_oe, i_i2c_ic_data_oe, i_i2c_ic_en, i_i2c_ic_gen_call_intr,
        i_i2c_ic_rd_req_intr, i_i2c_ic_rx_done_intr, i_i2c_ic_rx_full_intr,
        i_i2c_ic_rx_over_intr, i_i2c_ic_rx_under_intr, i_i2c_ic_start_det_intr,
```

Figure 16: AHB netlis

The memory interface with AMBA bus is a warper, whose pins are shown as below:

```
module ram_ahbif(
  input hclk,
  input hresetn,
  //ahb slave interface
  input hsel_s,
  input [19:0] haddr_s,
  input [2:0] hburst_s,
  input [1:0] htrans_s,
  input [31:0] hwdata_s,
  input hwrite_s,
  output [1:0] hresp_s,
  output [31:0] hrdata_s,
  output hready_s,
  //ram interface
  input [31:0] ram_rdata,
  output [17:0] ram_addr,
  output [31:0] ram_wdata,
  output ram_write
);
```

## 2.7 Memory Model Implementation

In our design, memory block is divided in to two parts: one is the RAM for communication of between Cortex M0 and hardware accelerator, and storage

of the assembly code running on Cortex M0; one is for storage of the arrays and vectors that been used in the hardware accelerator. The first part is the main RAM block that connect to the AMBA bus, who need a wraper to connect. The second part is accelerator built-in, e.g. generated by Vivado and already have well formed structure and interfaces.

### 2.8 Accelerator to Memory Interfaces

The accelerator to memory interface was originally designed as directly write and read, that is treat the main RAM as registers, and Vivado generated well structured interfaces with internal RAM and ROM for accelerator itself. However, during the compiling, we found that the Design Complier does not support large size (or even small size) of RAM and ROM, which would takes forever to compile. As a result, we have to remove those part and assign pins for those memory block in compiling step. But, with a better sever, we would still like to use the memory block generated by Vivado as it has the best performance.

## 3 Design Exploration

### 3.1 Computational Workload Comparison

For computational workload simulation on Cortex M0, we used a simplified version of code to only simulate the Dijkstra's algorithm, as the original C code is a way too heave workload for Cortex M0 to do.

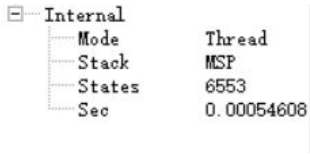



Figure 17: Dijkstra's Algorithm Workload on M0

As the map with 9 vertices shown above, we tested the code with Keil MDK-ARM, and the result are shown below. Notice that the number of vertices in the test are far more less than the real case number in a parking lot. The computation complexity is proportion to the number of vertices.

**Timing (ns)**

**Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| default | 20.00 | 16.57 | 2.50 |

**Latency (clock cycles)**

**Summary**

| Latency | | Interval | | |
|---|---|---|---|---|
| min | max | min | max | Type |
| 11552 | 11558 | 11553 | 11559 | none |

**Detail**

**Instance**

| | | Latency | | Interval | | |
|---|---|---|---|---|---|---|
| Instance | Module | min | max | min | max | Type |
| grp_top_calculateDir_fu_67 | top_calculateDir | 11518 | 11524 | 11518 | 11524 | none |

**Loop**

N/A

Figure 18: Vivado before optimization

**Performance Estimates**

**Timing (ns)**

**Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| default | 20.00 | 16.57 | 2.50 |

**Latency (clock cycles)**

**Summary**

| Latency | | Interval | | |
|---|---|---|---|---|
| min | max | min | max | Type |
| 11514 | 11520 | 11515 | 11521 | none |

**Detail**

**Instance**

**Loop**

Figure 19: Vivado after optimization

For Vivado optimization, as we can see from the two figures above, the optimization was not very effective since loops contains in the algorithm,

which we used in the hardware accelerator, are most not able to un-loop since the data dependence sequence. Anyway, the optimization we applied saved tens of clock cycles for the computation.

# 4 Test Bench Development

## 4.1 How To Set Up

The simulation was done in ModuleSim. The most unconvinence part of the simulation is that out application is more about a real world problem solver, so that the frequency of input is significantly smaller than the frequency of the system's clock. To simulate the system with reasonably computational time consume, the clock frequency was set to 200 kHz, which is just sufficient for the hardware accelrator to caculate the data that been delivered at 20 Hz, as each computational cycle take about 8,000 clock cycles. In this case, the simulation should be run for less than 2 minutes.

## 4.2 Behavioural Model for External Devices

The external device was set to deliver the input data once every 0.05 second, that is 20 Hz. Since the floating point operation problem was solved the in the very last minute, the simulation was been done without the data from gyroscope, which requires the value of $\pi$. The data was delivered as the correct behaviour of the car, who follow the instruction of the device correctly, that is one second of acceleration, one second for breaking, one second for turning, and rest of the time for uniform linear motion.

## 4.3 Debug Experience

The most head blowing debugging problem that we encountered was the DSP unit that Vivado generate all the time. The problem was, for the DSP units, the file been generated was in the form of $*_ip.tcl$ files, which cannot been recognized by the simulation tool, and result in unable to run the simulation. Before we found out the real reason behind it, we tried a few attempts:

1. Use device without DSP unit. However, there is not such device in the library been provided. The very few one contains 40 DSP units, and more than 1000 units for the most cases.

2. Use $-max_dsp0$ argument. However, it didn't work no matter been put in any Vivado processing step.

3. Remove $sqrt()$ function from the code. The DSP units been generated was reduced, but still some.

After data diggingly research, we found that the real reason behind this problem was that we used floating operation in the system, which cannot be easily implanted in the FPGA so that the compiler would transform it into DSP units to simplify the complexity of the system. The way to solve this problem is to use $AP_FIXED.h$ header file which defined a decimals data type with fixed decimal point. However, the header file was not defined well, which requires the data type of opration with $AP_FIXED$ type must be $AP_FIXED$ type, and decimals it need to be configured manually on bit level, which is no doubt very complex. A easier way we used is replace $float$ data type with $int$ type, which will eliminate floating point operation. However, $pi$ was used in gyroscope data processing, and that is why we have to disable the gyroscope data input while doing simulation.

# 5 Simulation Results

## 5.1 Design Compiler Synthesis Reports

```
Number of ports:                          125
Number of nets:                         10863
Number of cells:                        10459
Number of combinational cells:           9372
Number of sequential cells:              1087
Number of macros/black boxes:               0
Number of buf/inv:                       1167
Number of references:                      30

Combinational area:             440203.147316
Buf/Inv area:                    32758.198666
Noncombinational area:          238994.925034
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                679198.072350
Total area:                 undefined
```

Figure 20: Design Compiler Area Report

```
clock HCLK (rise edge)                                  50000.00   50000.00
clock network delay (ideal)                                 0.00   50000.00
u_cortexm0ds/u_logic/Pdi2z4_reg/ck (drsp_1)                 0.00   50000.00 r
library setup time                                       -267.12   49732.88
data required time                                                 49732.88
-------------------------------------------------------------------------
data required time                                                 49732.88
data arrival time                                                  -23008.45
-------------------------------------------------------------------------
slack (MET)                                                        26724.43
```

Figure 21: Design Compiler Timing Report

```
  Cell Internal Power  =   2.3622 mW   (88%)
  Net Switching Power  = 311.7471 uW   (12%)
                         ---------
Total Dynamic Power    =   2.6739 mW  (100%)

Cell Leakage Power     =   3.2582 uW
```

Figure 22: Design Compiler Power Report for All Parts

```
  Cell Internal Power  =   1.8531 mW   (92%)
  Net Switching Power  = 154.2096 uW    (8%)
                         ---------
Total Dynamic Power    =   2.0073 mW  (100%)

Cell Leakage Power     =   2.7166 uW
```

Figure 23: Design Compiler Power Report for Cortex M0

As we can see from the power reports, most of the power was consumed by the Cortex M0 - nearly 80%. We would really like to remove the M0, but not to keep the complexity of the project.

## 5.2 Encounter Place and Route Reports

```
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:04.1  MEM: 43.8M)
```

Figure 24: Encounter Geometry Verification

```
Begin Summary
  Found no problems or warnings.
End Summary

End Time: Wed Dec  2 22:21:41 2015
Time Elapsed: 0:00:01.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.5  MEM: 0.000M)
```

Figure 25: Encounter Connection Verification

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 11.574  | 17.040  | 11.574  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  1901   |  1079   |   848   |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |     94 (94)      |  -12.329   |     94 (94)      |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |     19 (19)      |   -2069    |     19 (19)      |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 100.000%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

Figure 26: Encounter Timing Report

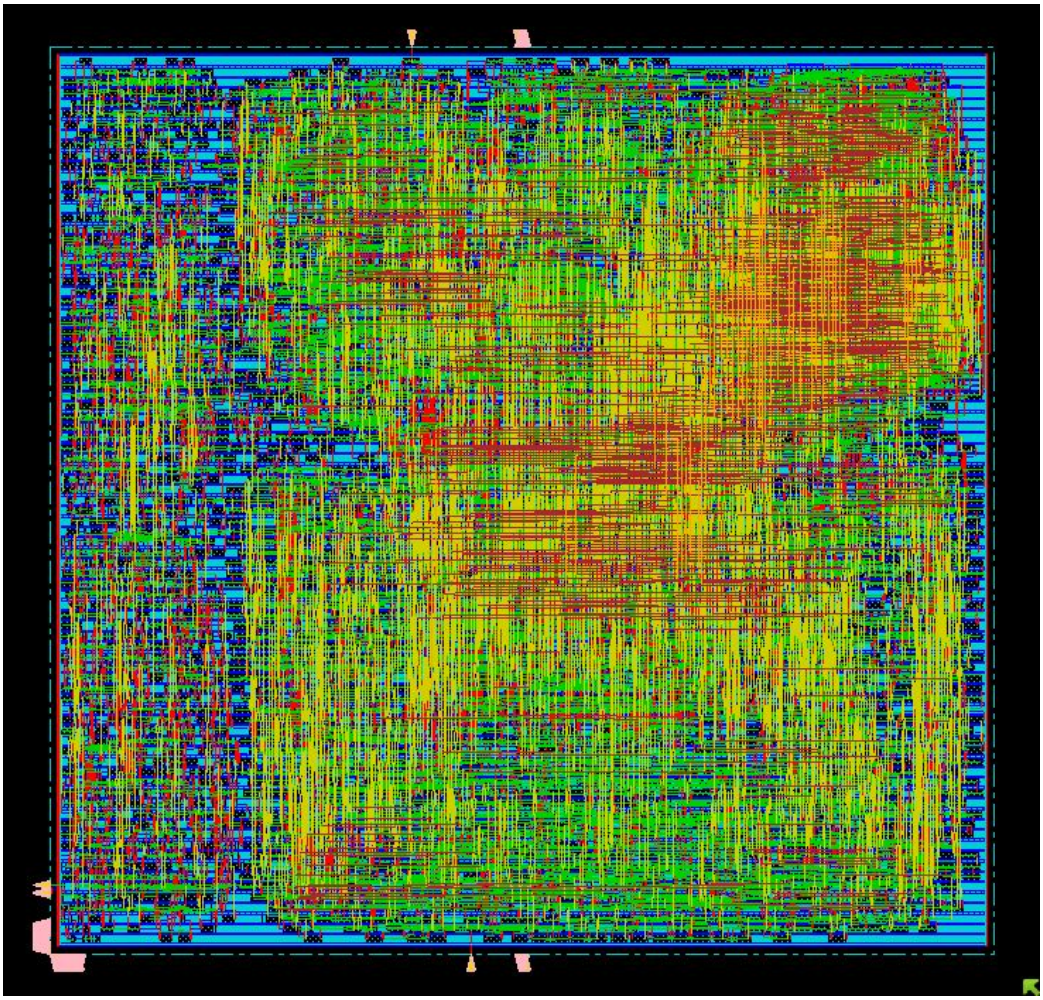


Figure 27: Floorplan Generated by Encounter

## 5.3 simulation screen capture and explanation

Figure 28: Simulation Screen Capture

The simulation result is shown as screen capture above. ap_start is generated by a external unit that takes clock signal and trigger the hardware accelerator for one clock cycle at 20 Hz - same as the frequency of data input from the sensors. This task could be done by M0, but at a much higher cost.

As we can see, the data was delivered about 0.01 second before the next data is delivered, which perfectly met the timing requirement of the system, even with a save margin.

The output of the system is number 2 right now, which indicate turn right and 1 indicate go straight when not turning. It will only valid for one clock cycle, but will been written into the RAM directly. The validation is indicated by the signal next_dirc_ap_vld, as the idle signal goes down, done and ready signal goes up.

# 6 Conclusion and Future Work

To sum up, the design goal is nicely accomplished. The requirement is been met functionally and timingly, also a little compromised in function.

For future work, optimization on the Dijkstra's algorithm is definitely a important part, as the algorithm is $O(n^2)$ complicated, e.g. it will cost much more time to compute the path with a large parking lot graph. What's more, error path correction function can be implanted.

# 7 Individual Contribution

| Design Step | Yitong Ding | Kaijie Shen | Qiankun Xie | Yingjie Zhou |
|---|---|---|---|---|
| Proposal | 25% | 25% | 25% | 25% |
| C Programming | 65% | 0% | 15% | 20% |
| Vivado Compiling | 70% | 10% | 10% | 10% |
| Simulation | 60% | 30% | 5% | 5% |
| CoreAssembler | 0% | 100% | 0% | 0% |
| System Wrap Up | 100% | 0% | 0% | 0% |
| Design Compiler | 100% | 0% | 0% | 0% |
| Encounter | 100% | 0% | 0% | 0% |
| Final Report | 60% | 20% | 15% | 5% |

# 8 Appendix

## 8.1 C code and script for accelerator design

```
#ifndef _header
#define _header

#define MAX 38
#define INF          (~(0x1<<31))
#define N 1
#define E 2
#define S 3
#define W 4
#define X -1
#define din_f   int
#define din_i   int
#define dout_f  int
#define dout_i  int
#define num_arrive 0
#define num_stright 1
#define num_turn_right 2
#define num_turn_left 3
#define num_turn_back 4
#define num_error -1

void dijkstra(int (*matrix)[MAX], int vs, int prev[MAX], int dist[MAX]);

int find_dict_now(int theta, int init_dict);
int find_dirc(int (*dirMatrix)[MAX], int last_last_point, int last_point, int next_point
    );

void calculateDir(
  din_f (*matrix)[MAX],
  din_i (*dirMatrix)[MAX],
  din_i start_point,
  din_i end_point,
  din_f distance,
  din_f theta,
  din_i init_dict,
  dout_i *next_dirc,
  dout_i *last_last_point,
  dout_i *last_point);

void top(int accel_x, int accel_y, int yaw, int start_point, int end_point, int *
    next_dirc);

void odometers(int accel_x, int accel_y, int yaw, int *last_theta, int *first_velocity,
    int *total_distance, int *theta);

#endif
```

```
#include "Header.h"
#include "math.h"

/*Input: accel_x, accel_y, yaw*/
void top(int accel_x, int accel_y, int yaw, int start_point, int end_point, int *
    next_dirc)
{
  int matrix[MAX][MAX] = {
    0, INF, INF, INF, INF, INF, INF, INF, 1, INF, INF, INF, INF, 1, INF, INF, INF,
     INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
      INF, INF, INF,
    INF, 0, INF, INF, INF, INF, INF, INF, INF, 1, 1, INF, INF, INF, INF, INF, INF,
     INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     INF, INF, INF,
    INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, INF, INF, INF, INF
     , INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF,
      INF, INF, INF,
    INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     INF, INF, INF, INF, INF, INF, 1, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1,
     INF, INF, INF,
```

```
INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 1, INF, INF, INF, INF
 , INF, 1, INF,
INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF,
INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF, 1,
INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1,
 INF, INF, INF, 1,
1, INF, INF, INF, INF, INF, INF, INF, INF, 0, 1, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, 1, INF, INF, INF, INF, INF, INF, INF, 1, 0, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, 1, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, 1, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, 1, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, 1, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, 1,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1,
 0, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  1, 0, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, 1, 0, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, 1, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, 0, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF, INF, 1, 0, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, 1, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, 0, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF, INF, INF, INF, INF, 1, 0, 1, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, INF, INF, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, 1, INF, INF, INF, INF, INF, INF
 , INF, INF, INF,
INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
  INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, 1, INF, INF, INF, INF, INF,
  INF, INF, INF,
INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
 INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, INF, INF, INF, INF, INF
```

```
    , INF, INF, INF,
   INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
    INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, 1, INF, INF, INF
    , INF, INF, INF,
   INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, 1, INF, INF,
     INF, INF, INF,
   INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, 1, INF,
     INF, INF, INF,
   INF, INF, INF, INF, INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF,
    INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 1, 0, INF
    , INF, INF, INF,
   INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
    INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     0, 1, INF, INF,
   INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF
    , 1, 0, 1, INF,
   INF, INF, INF, INF, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
    INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
     INF, 1, 0, INF,
   INF, INF, INF, INF, INF, INF, INF, 1, 1, INF, INF, INF, INF, INF, INF, INF, INF, INF
    , INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
    INF, INF, INF, 0
};
int dirMatrix[MAX][MAX] = {
   0, X, X, X, X, X, X, X, X, E, X, X, X, X, S, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, 0, X, X, X, X, X, X, X, X, W, E, X, X, X, X, X, X, X, X, X, S, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, 0, X, X, X, X, X, X, X, X, X, X, W, X, X, X, X, X, X, X, X, X, X, X, X, S,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, S, X, X,
    X, X, X, X, X, X, E, X, X, X,
   X, X, X, X, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, N, S, X, X, X, X, X, W, X,
   X, X, X, X, X, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, E,
   X, X, X, X, X, X, X, X, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, N, X, X, X, W,
   W, X, X, X, X, X, X, X, X, X, 0, E, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, E, X, X, X, X, X, X, X, W, 0, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, W, X, X, X, X, X, X, X, X, X, 0, E, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, X, X, X, X, W, 0, E, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, E, X, X, X, X, X, X, X, X, X, W, 0, X, X, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   N, X, X, X, X, X, X, X, X, X, X, X, X, X, 0, S, X, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, S, X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, S, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, S, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, N, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, S, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, N, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, 0, S, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, S, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, S, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X,
   X, X, X, N, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, 0, S, X, X,
    X, X, X, X, X, X, X, X, X, X,
```

```
    X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, S, X,
     X, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, S, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N, 0, X,
     X, X, X, X, X, X, X, X, X, X,
    X, X, N, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, 0,
     S, X, X, X, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, N,
     0, S, X, X, X, X, X, X, X, X,
    X, X, X, X, S, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     N, 0, X, X, X, X, X, X, X, X,
    X, X, X, X, N, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, 0, S, X, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, N, 0, S, X, X, X, X, X,
    X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, X, N, 0, S, X, X, X, X,
    X, X, X, X, X, X, X, X, S, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, X, X, N, 0, X, X, X, X,
    X, X, X, W, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, X, X, X, X, 0, E, X, X,
    X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, X, X, X, X, W, 0, E, X,
    X, X, X, X, E, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, X, X, X, X, X, W, 0, X,
    X, X, X, X, X, X, X, W, E, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X, X,
     X, X, X, X, X, X, X, X, X, 0
  };
  //data need to be stored
  static int distance = 0;  //total distance moved
  static int theta = 0;     //total theta turned
  static int last_theta = 0;
  static int first_velocity = 0;
  static int total_distance = 0;

  static int init_dict = N;
  static int last_last_point = 5;
  static int last_point = 20;

  odometers(accel_x, accel_y, yaw, &last_theta, &first_velocity, &total_distance, &theta
     );
  //get next direction
  calculateDir(matrix, dirMatrix, start_point, end_point, distance, theta, init_dict,
     next_dirc, &last_last_point, &last_point);

}

/*Input: accel_x, accel_y, yaw; Output: total_distance, theta*/
void odometers(int accel_x, int accel_y, int yaw,
  int *last_theta, int *first_velocity, int *total_distance, int *theta)
{
  int theta_global;   /* global theta from IMU, radians */
  int accel;        /* bot Y position in inches */
  int velocity;
  int imu_theta;      /* change in theta calculate from IMU */
  int distance = 0;
  int t = 5;

  /*calcaulate distance */
  //accel = sqrt(pow(accel_x, 2) + pow(accel_y, 2));
  accel = accel_x/100;
  //accel = sqrt(accel_x[i] * accel_x[i] + accel_y * accel_y);
  velocity = *first_velocity + accel * t/100;
  distance = (*first_velocity + velocity) / 2 * t/100;
  *first_velocity = velocity;
  *total_distance += distance;
  /* read the YAW value from the imu struct and convert to radians */
  //theta_global = ((int)yaw * 2 * pi) / 65536.0;

  /* calculate rotation rate-of-change for stasis detector */
  imu_theta = theta_global - *last_theta;
  *last_theta = theta_global;
  *theta = theta_global;
  //*theta -= (int)(((int)(*theta / (2 * pi))) * 2 * pi);
  //*theta -= (int)(((int)(*theta / (2 * pi))) * 2 * pi);
  /*if (*theta < -pi)
  {
    *theta += 2 * pi;
```

```
  }
  else
  {
    if (*theta > pi)
      *theta -= 2 * pi;
  }
  */
}


void calculateDir(
  din_f (*matrix)[MAX],
  din_i (*dirMatrix)[MAX],
  din_i start_point,
  din_i end_point,
  din_f distance,
  din_f theta,
  din_i init_dict,
  dout_i *next_dirc,
  dout_i *last_last_point,
  dout_i *last_point)
{
  int sptSet[MAX] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
  int dist[MAX] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
  dijkstra(matrix, end_point, sptSet, dist);

  int total_distance = dist[start_point];
  //int dirc_now = find_dict_now(theta, init_dict);

  if (total_distance == distance) //arrive the end_point
  {
    next_dirc = num_arrive;
  }
  else if (total_distance - distance < dist[sptSet[*last_point]])
  {
    *next_dirc = num_stright;
    *last_last_point = *last_point;
    *last_point = sptSet[*last_point];
  }
  else if (total_distance - distance == dist[sptSet[*last_point]])
  {
    *next_dirc = find_dirc(dirMatrix, *last_last_point, *last_point, sptSet[*last_point
     ]);//nextdirction
  }
  else
  {
    *next_dirc = num_stright;
  }

}

int find_dirc(int (*dirMatrix)[MAX], int last_last_point, int last_point, int next_point
     )
{
  int last_dirc = dirMatrix[last_last_point][last_point];
  int next_dirc = dirMatrix[last_point][next_point];
  if (last_dirc == next_dirc)
  {
    return num_stright;
  }
  else if (next_dirc - last_dirc == 1 || next_dirc - last_dirc == -3)
  {
    return num_turn_right;
  }
  else if (next_dirc - last_dirc == -1 || next_dirc - last_dirc == 3)
  {
    return num_turn_left;
  }
  else if (next_dirc - last_dirc == 2 || next_dirc - last_dirc == -2)
  {
    return num_turn_back;
  }
  else
  {
    return num_stright;
```

```
  }
}
/*
int find_dict_now(int theta, int init_dict)
{
  int dirc_now;
  if (theta <= pi / 4 || theta > -pi / 4)
  {
    return init_dict;
  }
  else if (theta <= 3 / 4 * pi || theta > pi / 4)
  {
    dirc_now = init_dict + 1;
    if (dirc_now > 4)
    {
      dirc_now -= 4;
    }
    return dirc_now;
  }
  else if (theta > -3 / 4 * pi || theta <= -pi / 4)
  {
    dirc_now = init_dict - 1;
    if (dirc_now < 1)
    {
      dirc_now += 4;
    }
    return dirc_now;
  }
  else
  {
    dirc_now = init_dict + 2;
    if (dirc_now > 4)
    {
      dirc_now -= 4;
    }
    return dirc_now;
  }
}
*/
void dijkstra(int (*matrix)[MAX], int vs, int prev[MAX], int dist[MAX])
{
  int i, j, k;
  int min;
  int tmp;
  int flag[MAX];
  //char vexs[MAX] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i' };
  int vexnum = MAX;

loop1: for (i = 0; i < vexnum; i++)
{
  flag[i] = 0;
  prev[i] = 0;
  dist[i] = matrix[vs][i];
}

      flag[vs] = 1;
      dist[vs] = 0;

   loop2: for (i = 1; i < vexnum; i++)
   {
     min = INF;
   loop3: for (j = 0; j < vexnum; j++)
   {
     if (flag[j] == 0 && dist[j]<min)
     {
       min = dist[j];
       k = j;
     }
   }

       flag[k] = 1;

    loop4: for (j = 0; j < vexnum; j++)
    {
      tmp = (matrix[k][j] == INF ? INF : (min + matrix[k][j]));
      if (flag[j] == 0 && (tmp  < dist[j]))
      {
```

```
        dist[j] = tmp;
        prev[j] = k;
      }
    }
  }

}
```

## 8.2 C code for programs running in M0

```
   //-----------------------------------------------------------------------------
// The confidential and proprietary information contained in this file may
// only be used by a person authorised under and to the extent permitted
// by a subsisting licensing agreement from ARM Limited.
//
//             (C) COPYRIGHT 2010 ARM Limited.
//                 ALL RIGHTS RESERVED
//
// This entire notice must be reproduced on all copies of this file
// and copies of this file may only be made by a person if such person is
// permitted to do so under the terms of a subsisting license agreement
// from ARM Limited.
//
//      SVN Information
//
//      Checked In          : 2010-08-03 14:15:21 +0100 (Tue, 03 Aug 2010)
//
//      Revision            : 144883
//
//      Release Information : AT510-MN-80001-r0p0-00rel0
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Cortex-M0 DesignStart C program example
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Include standard C library and RealView Compiler header.
//-----------------------------------------------------------------------------

#include <stdio.h>
#include <time.h>
#include <rt_misc.h>

//-----------------------------------------------------------------------------
// The following code implements all the functions required for semihosting
// using the Cortex-M0 DesignStart testbench, mapping output from C to the
// console output of the testbench.
//-----------------------------------------------------------------------------

// Disable the inclusion of semihosting support; see the "ARM Compiler
// toolchain Using ARM C and C++ Libraries and Floating-Point Support"
// (ARM DUI 0475) for further details.
#pragma import(__use_no_semihosting)

// Define where the top of memory is.
#define TOP_OF_RAM 0x20000U

//-----------------------------------------------------------------------------
// Define location of C stack and heap
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Implement the minimum number of functions required to support
// standard C input/output operations without a debugger attached.
//-----------------------------------------------------------------------------

// Define the location of the output console in the DesignStart testbench.
volatile unsigned char *console = (volatile unsigned char *) 0x40000000U;

// Implement a simple structure for C's FILE handle.
struct __FILE { int handle; };
FILE __stdout;
FILE __stdin;
```

```c
// Implement file IO handling, only console output is supported.
time_t time(time_t* t)          { static time_t clock = 0; return clock++; }
int fputc(int ch, FILE *f)      { *console = ch; return ch; }
int ferror(FILE *f)             { return 0; }
int fgetc(FILE *f)              { return -1; }
int __backspace(FILE *stream)   { return 0; }
void _ttywrch(int ch)           { fputc(ch,&__stdout); }

// Writing 0xD to the console causes the DesignStart testbench to finish.
void _sys_exit(void)            { fputc(0x0D,&__stdout); while(1); }

//-----------------------------------------------------------------------------
// Implement the vector table in its own area to facilitate linking first
//-----------------------------------------------------------------------------

extern void __main(void);       // Use C-library initialization function.

__attribute__ ((section("vectors")))
  static void (* const vector_table[])(void) =
{
  (void (*)(void)) TOP_OF_RAM, // Initial value for stack pointer.
  __main,                      // Reset handler is C initialization.
  0,                           // No HardFault handler, just cause lockup.
  0,                           // No NMI handler, just cause lockup.
  0//...                       // Additional handlers would be listed here.
};

//-----------------------------------------------------------------------------
// Simple "Hello World" program.
//-----------------------------------------------------------------------------

int main(void) {
  int * ADD_AP_START       = (int *) 0x10000U;
  int * ADD_AP_DONE      = (int *) 0x10004U;
  int * ADD_AP_IDLE      = (int *) 0x10008U;
  int * ADD_AP_READY       = (int *) 0x1000cU;
  int * ADD_ACCEL_X      = (int *) 0x10010U;
  int * ADD_ACCEL_Y      = (int *) 0x10014U;
  int * ADD_YAW        = (int *) 0x10018U;
  int * ADD_START_POINT    = (int *) 0x1001cU;
  int * ADD_END_POINT      = (int *) 0x10020U;
  int * ADD_NEXT_DIRC      = (int *) 0x10024U;
  int * ADD_NEXT_DIRC_AP_VLD  = (int *) 0x10028U;
  int * i2c_ACCEL_X      = (int *) 0x10030U;
  int * i2c_ACCEL_Y      = (int *) 0x10034U;
  int * i2c_YAW        = (int *) 0x10038U;

  * ADD_START_POINT = 7;
  * ADD_END_POINT = 20;
  while (1)
  {
    * ADD_ACCEL_X = * i2c_ACCEL_X;
    * ADD_ACCEL_Y = * i2c_ACCEL_Y;
    * ADD_YAW    = * i2c_YAW;
    if (*ADD_AP_DONE)
    {
      printf("%d", *ADD_NEXT_DIRC);
    }
  }

  //printf("Hello World!\n");

  //printf("This message was printed from helloworld.c provided\n");
  //printf("with the ARM Cortex-M0 DesignStart processor\n");

  return 0;
}
```

## 8.3 System Wraper File

```c
module top_wraper (
```

```
  input wire HCLK,
  input wire HRESETn,
  input wire i_i2c_ic_clk_in_a ,
  input wire i_i2c_ic_data_in_a ,
  output wire i_i2c_ic_clk_oe ,
  output wire i_i2c_ic_data_oe ,
  output wire i_i2c_ic_en ,
  input wire in_ACCEL_X,
  input wire in_ACCEL_Y,
  input wire in_YAW,
  input wire [31:0] ram_rdata,
  output wire [17:0] ram_addr,
  output wire [31:0] ram_wdata,
  output wire ram_write,
  output wire [31:0] NEXT_DIRC
);

//-----------------------------------------------------------------------------
// Define parameters for clock period and power-on reset delay
//-----------------------------------------------------------------------------

//localparam clk_period = 100;            // Simulation cycles per clock period
//localparam por_delay  = 1001;           // Simulation cycles of power-on-reset
localparam ram_log2   = 18;             // Power of two of RAM words
//localparam addr_tty   = 32'h40000000;   // Address of output console

//-----------------------------------------------------------------------------
// Define registers for clock, reset and memory
//-----------------------------------------------------------------------------

//reg sim_clock;                          // System clock
//reg power_on_reset_n;                   // Power-on reset signal
//reg [31:0] ram [0:(2**ram_log2)-1];     // Storage for AHB memory model

//-----------------------------------------------------------------------------
// Cortex-M0 DesignStart signal list
//-----------------------------------------------------------------------------

// See the AMBA(r)3 AHB-Lite Protocol Specification v1.0 (ARM IHI 0033),
// and the Cortex(tm)-M0 Technical Reference Manual (ARM DDI 0432), for
// further details on the following signals:
//wire        HCLK;                // AHB-Lite interface and CPU master clock
//wire        HRESETn;             // AHB-Lite active-low reset signal

wire [31:0] HADDR;               // AHB-Lite byte address
wire [ 2:0] HBURST;              // AHB-Lite burst type (not used by testbench)
wire        HMASTLOCK;           // AHB-Lite locked transaction (always zero)
wire [ 3:0] HPROT;               // AHB-Lite protection (not used by testbench)
wire [ 2:0] HSIZE;               // AHB-Lite size (# of bits: 0=8, 1=16, 2=32)
wire [ 1:0] HTRANS;              // AHB-Lite perform transaction
wire [31:0] HWDATA;              // AHB-Lite write-data
wire        HWRITE;              // AHB-Lite transaction is write not read
wire [31:0] HRDATA;              // AHB-Lite read-data
wire        HREADY;              // AHB-Lite bus ready signal
wire [1:0]  HRESP;               // AHB-Lite bus error (not used by testbench)
wire    HBUSREQ;       // AHB-Lite HBUSREQ output will be left unconnected
wire    HGRANT;        // AHB-Lite HGRANT input tied to logic '1' (1'b1).


wire    HSEL_S ;
wire [31:0] HADDR_S ;
wire [ 2:0] HBURST_S ;
wire [ 1:0] HTRANS_S ;
wire [31:0] HWDATA_S ;
wire    HWRITE_S ;
wire [ 1:0] HRESP_S ;
wire [31:0] HRDATA_S ;
wire    HREADY_S ;
//need to be assign
wire    HMASTLOCK_S;
wire    HPORT_S;
wire [ 2:0] HSIZE_S;
wire    HREADY_RESP_S;
// See the ARMv6-M Architecture Reference Manual (ARM DDI 0419), and the
// Cortex(tm)-M0 Technical Reference Manual (ARM DDI 0432), for further
// details on the following signals:
```

```
wire        NMI;                // Non-maskable interrupt input (not used by tb)
wire [15:0] IRQ;                // Interrupt inputs (not used by testbench)

wire        TXEV;               // Event output (CPU executed SEV instruction)
wire        RXEV;               // Event input (not used by testbench)

wire        LOCKUP;             // CPU stopped due to multiple software errors
wire        SYSRESETREQ;        // CPU request for system to be reset

wire        SLEEPING;           // CPU is sleeping (not used by testbench)

//-----------------------------------------------------------------------------
// Generate system clock, power-on reset and synchronized AHB reset signals
//-----------------------------------------------------------------------------
/*
// Generate a clock of the appropriate period
initial
  #0 sim_clock = 1'b0;

always @(sim_clock)
  #(clk_period/2) sim_clock <= ~sim_clock;

// Release the active-low power-on reset signal after the given delay
initial begin
  #0 power_on_reset_n = 1'b0;
  #(por_delay) power_on_reset_n = 1'b1;
end

// Synchronize AHB reset, and factor in reset request from the CPU
reg [1:0] rst_sync;
always @(posedge sim_clock or negedge power_on_reset_n)
  if(!power_on_reset_n)
    rst_sync <= 2'b00;
  else
    rst_sync <= {rst_sync[0],~SYSRESETREQ};
*/
//-----------------------------------------------------------------------------
// Connect clock and reset to M0 signals and assign static signals
//-----------------------------------------------------------------------------

//assign HCLK       = sim_clock;    // Assign AHB clock from simulation clock
//assign HRESETn    = rst_sync[1];  // Assign AHB clock from synchronizer
assign HREADY     = 1'b1;         // All devices are zero-wait-state
assign NMI        = 1'b0;         // Do not generate any non-maskable interrupts
assign IRQ        = {16{1'b0}};   // Do not generate any interrupts
assign RXEV       = 1'b0;         // Do not generate any external events
assign HRESP[1:0]   = 2'b00;         // No device in this system generates errors
assign HGRANT     = 1'b1;

//signal for slave stage
assign HMASTLOCK_S  = 1'b0;
assign HPORT_S    = 1'b0;
assign HSIZE_S[2:0] = 3'b000;
assign HREADY_RESP_S= 1'b1;


CORTEXM0DS u_cortexm0ds (
  .HCLK         (HCLK),
  .HRESETn      (HRESETn),
  .HADDR        (HADDR[31:0]),
  .HBURST       (HBURST[2:0]),
  .HMASTLOCK    (HMASTLOCK),
  .HPROT        (HPROT[3:0]),
  .HSIZE        (HSIZE[2:0]),
  .HTRANS       (HTRANS[1:0]),
  .HWDATA       (HWDATA[31:0]),
  .HWRITE       (HWRITE),
  .HRDATA       (HRDATA[31:0]),
  .HREADY       (HREADY),
  .HRESP        (HRESP[0]),
  .NMI          (NMI),
  .IRQ          (IRQ[15:0]),
  .TXEV         (TXEV),
  .RXEV         (RXEV),
  .LOCKUP       (LOCKUP),
  .SYSRESETREQ  (SYSRESETREQ),
  .SLEEPING     (SLEEPING)
```

```
);

//-----------------------------------------------------------------------------
// Connect clock and reset to AHB
//-----------------------------------------------------------------------------


wire i_ahb_pause;
wire i_apb_pclk_en;

//Arbiter Slave Interface Signals (left open)
wire [31:0] i_ahb_hrdata_s0;
wire    i_ahb_hready_resp_s0;
wire [1:0]  i_ahb_hresp_s0;
wire    i_ahb_hsel_s0;
wire    i_ahb_ahbarbint;

//Observability Signals (left open)
wire [3:0]  i_ahb_hmaster_data;

wire i_i2c_debug_addr;
wire i_i2c_debug_addr_10bit;
wire i_i2c_debug_data;
wire i_i2c_debug_hs;
wire i_i2c_debug_master_act;
wire [4:0] i_i2c_debug_mst_cstate;
wire i_i2c_debug_p_gen;
wire i_i2c_debug_rd;
wire i_i2c_debug_s_gen;
wire i_i2c_debug_slave_act;
wire [3:0] i_i2c_debug_slv_cstate;
wire i_i2c_debug_wr;

wire i_i2c_ic_activity_intr;
wire i_i2c_ic_gen_call_intr;
wire i_i2c_ic_rd_req_intr;
wire i_i2c_ic_rx_done_intr;
wire i_i2c_ic_rx_full_intr;
wire i_i2c_ic_rx_over_intr;
wire i_i2c_ic_rx_under_intr;
wire i_i2c_ic_start_det_intr;
wire i_i2c_ic_stop_det_intr;
wire i_i2c_ic_tx_abrt_intr;
wire i_i2c_ic_tx_empty_intr;
wire i_i2c_ic_tx_over_intr;

assign i_ahb_pause          = 1'b0;
assign i_apb_pclk_en        = 1'b1;

assign i_i2c_debug_addr       = 1'b0;
assign i_i2c_debug_addr_10bit   = 1'b0;
assign i_i2c_debug_data       = 1'b0;
assign i_i2c_debug_hs       = 1'b0;
assign i_i2c_debug_master_act   = 1'b0;
assign i_i2c_debug_mst_cstate[4:0]  = 5'b0;
assign i_i2c_debug_p_gen      = 1'b0;
assign i_i2c_debug_rd       = 1'b0;
assign i_i2c_debug_s_gen      = 1'b0;
assign i_i2c_debug_slave_act    = 1'b0;
assign i_i2c_debug_slv_cstate[3:0]  = 4'b0;
assign i_i2c_debug_wr       = 1'b0;

assign i_i2c_ic_activity_intr   = 1'b0;
assign i_i2c_ic_gen_call_intr   = 1'b0;
assign i_i2c_ic_rd_req_intr   = 1'b0;
assign i_i2c_ic_rx_done_intr  = 1'b0;
assign i_i2c_ic_rx_full_intr  = 1'b0;
assign i_i2c_ic_rx_over_intr  = 1'b0;
assign i_i2c_ic_rx_under_intr   = 1'b0;
assign i_i2c_ic_start_det_intr  = 1'b0;
assign i_i2c_ic_stop_det_intr   = 1'b0;
assign i_i2c_ic_tx_abrt_intr  = 1'b0;
assign i_i2c_ic_tx_empty_intr   = 1'b0;
assign i_i2c_ic_tx_over_intr  = 1'b0;

parking AHB (
  // Ports for Interface HCLK
```

```
.HCLK_hclk            (HCLK),
// Ports for Interface HRESETn
.HRESETn_hresetn       (HRESETn),
// Ports for Interface PCLK
.PCLK_pclk            (HCLK),
// Ports for Interface PRESETn
.PRESETn_presetn       (HRESETn),
// Ports for Interface ex_i_ahb_AHB_Master
.ex_i_ahb_AHB_Master_haddr   (HADDR[31:0]),
.ex_i_ahb_AHB_Master_hburst  (HBURST[2:0]),
.ex_i_ahb_AHB_Master_hbusreq(HBUSREQ),
.ex_i_ahb_AHB_Master_hlock   (HMASTLOCK),
.ex_i_ahb_AHB_Master_hprot   (HPROT[3:0]),
.ex_i_ahb_AHB_Master_hsize   (HSIZE[2:0]),
.ex_i_ahb_AHB_Master_htrans  (HTRANS[1:0]),
.ex_i_ahb_AHB_Master_hwdata  (HWDATA[31:0]),
.ex_i_ahb_AHB_Master_hwrite  (HWRITE),
.ex_i_ahb_AHB_Master_hgrant  (HGRANT),
.ex_i_ahb_AHB_Master_hrdata  (HRDATA[31:0]),
.ex_i_ahb_AHB_Master_hready  (HREADY),
.ex_i_ahb_AHB_Master_hresp   (HRESP[1:0]),
// Ports for Interface ex_i_ahb_AHB_Slave
.ex_i_ahb_AHB_Slave_hrdata   (HRDATA_S[31:0]),   //input
.ex_i_ahb_AHB_Slave_hready_resp(HREADY_RESP_S), //input
.ex_i_ahb_AHB_Slave_hresp (HRESP_S[ 1:0]),    //input
.ex_i_ahb_AHB_Slave_haddr (HADDR_S[31:0]),
.ex_i_ahb_AHB_Slave_hburst   (HBURST_S[ 2:0]),
.ex_i_ahb_AHB_Slave_hmastlock(HMASTLOCK_S),
.ex_i_ahb_AHB_Slave_hprot (HPORT_S),
.ex_i_ahb_AHB_Slave_hready   (HREADY_S),
.ex_i_ahb_AHB_Slave_hsel   (HSEL_S),
.ex_i_ahb_AHB_Slave_hsize (HSIZE_S),
.ex_i_ahb_AHB_Slave_htrans   (HTRANS_S[ 1:0]),
.ex_i_ahb_AHB_Slave_hwdata   (HWDATA_S[31:0]),
.ex_i_ahb_AHB_Slave_hwrite   (HWRITE_S),
// Ports for Manually exported pins
.i_ahb_pause          (i_ahb_pause),       //input//
.i_apb_pclk_en          (i_apb_pclk_en),     //input//
.i_i2c_ic_clk         (HCLK),          //input
.i_i2c_ic_clk_in_a       (i_i2c_ic_clk_in_a),  //input from the top
.i_i2c_ic_data_in_a      (i_i2c_ic_clk_in_a),  //input from the top
.i_i2c_ic_rst_n         (HRESETn),         //input
.i_i2c_ic_clk_oe        (i_i2c_ic_clk_oe),    //output to the top
.i_i2c_ic_data_oe       (i_i2c_ic_data_oe),   //output to the top
.i_i2c_ic_en          (i_i2c_ic_en),       //output to the top
.i_ahb_ahbarbint        (i_ahb_ahbarbint),
.i_ahb_hmaster_data      (i_ahb_hmaster_data),
.i_ahb_hrdata_s0        (i_ahb_hrdata_s0),
.i_ahb_hready_resp_s0    (i_ahb_hready_resp_s0),
.i_ahb_hresp_s0         (i_ahb_hresp_s0),
.i_ahb_hsel_s0          (i_ahb_hsel_s0),
.i_i2c_debug_addr       (i_i2c_debug_addr),
.i_i2c_debug_addr_10bit    (i_i2c_debug_addr_10bit),
.i_i2c_debug_data       (i_i2c_debug_data),
.i_i2c_debug_hs         (i_i2c_debug_hs),
.i_i2c_debug_master_act    (i_i2c_debug_master_act),
.i_i2c_debug_mst_cstate    (i_i2c_debug_mst_cstate),
.i_i2c_debug_p_gen       (i_i2c_debug_p_gen),
.i_i2c_debug_rd         (i_i2c_debug_rd),
.i_i2c_debug_s_gen       (i_i2c_debug_s_gen),
.i_i2c_debug_slave_act     (i_i2c_debug_slave_act),
.i_i2c_debug_slv_cstate    (i_i2c_debug_slv_cstate),
.i_i2c_debug_wr         (i_i2c_debug_wr),
.i_i2c_ic_activity_intr    (i_i2c_ic_activity_intr),
.i_i2c_ic_gen_call_intr    (i_i2c_ic_gen_call_intr),
.i_i2c_ic_rd_req_intr    (i_i2c_ic_rd_req_intr),
.i_i2c_ic_rx_done_intr     (i_i2c_ic_rx_done_intr),
.i_i2c_ic_rx_full_intr     (i_i2c_ic_rx_full_intr),
.i_i2c_ic_rx_over_intr     (i_i2c_ic_rx_over_intr),
.i_i2c_ic_rx_under_intr    (i_i2c_ic_rx_under_intr),
.i_i2c_ic_start_det_intr   (i_i2c_ic_start_det_intr),
.i_i2c_ic_stop_det_intr    (i_i2c_ic_stop_det_intr),
.i_i2c_ic_tx_abrt_intr     (i_i2c_ic_tx_abrt_intr),
.i_i2c_ic_tx_empty_intr    (i_i2c_ic_tx_empty_intr),
.i_i2c_ic_tx_over_intr     (i_i2c_ic_tx_over_intr)
);
```

```
//-------------------------------------//
//AHB to RAM wrapper
//-------------------------------------//
/*
wire [31:0] ram_rdata ;
wire [31:0] ram_addr  ;
wire [31:0] ram_wdata ;
wire    ram_write ;
*/
ram_ahbif U_ram_ahbif (
  .hclk      (HCLK ),
  .hresetn   (HRESETn ),
  .hsel_s    (HSEL_S ),
  .haddr_s   (HADDR_S[31:0] ),
  .hburst_s  (HBURST_S[ 2:0] ),
  .htrans_s    (HTRANS_S[ 1:0] ),
  .hrdata_s    (HRDATA_S[31:0] ),
  .hwdata_s    (HWDATA_S[31:0] ),
  .hwrite_s    (HWRITE_S ),
  .hready_s    (HREADY_S ),
  .hresp_s   (HRESP_S[ 1:0] ),
  .ram_rdata   (ram_rdata ),
  .ram_addr    (ram_addr ),
  .ram_wdata   (ram_wdata ),
  .ram_write   (ram_write )
);



//-----------------------------------------//
//hardware accelerater
//-----------------------------------------//
wire        AP_START;      //in
wire        AP_DONE;       //out
wire        AP_IDLE;       //out
wire        AP_READY;      //out
wire  [31:0]  ACCEL_X;       //in
wire  [31:0]  ACCEL_Y;       //in
wire  [31:0]  YAW;         //in
wire  [31:0]  START_POINT;     //in
wire  [31:0]  END_POINT;       //in
//wire  [31:0]  NEXT_DIRC;       //out to the top
wire      NEXT_DIRC_AP_VLD; //out

//assign ram address for simulation data input
localparam  ADD_in_ACCEL_X     = 32'h00010030;
localparam  ADD_in_ACCEL_Y     = 32'h00010034;
localparam  ADD_in_YAW       = 32'h00010038;
//assign ram address for the input and output ports of hardware accelerater
localparam  ADD_AP_START     = 32'h00010000;      //in
localparam  ADD_AP_DONE      = 32'h00010004;      //out
localparam  ADD_AP_IDLE      = 32'h00010008;      //out
localparam  ADD_AP_READY     = 32'h0001000c;      //out
localparam  ADD_ACCEL_X      = 32'h00010010;      //in
localparam  ADD_ACCEL_Y      = 32'h00010014;      //in
localparam  ADD_YAW        = 32'h00010018;      //in
localparam  ADD_START_POINT    = 32'h0001001c;      //in
localparam  ADD_END_POINT    = 32'h00010020;      //in
localparam  ADD_NEXT_DIRC    = 32'h00010024;      //out
localparam  ADD_NEXT_DIRC_AP_VLD= 32'h00010028;      //out

top park_guild (
  .ap_clk        (HCLK),
  .ap_rst        (HRESETn),
  .ap_start      (AP_START),
  .ap_done       (AP_DONE),
  .ap_idle       (AP_IDLE),
  .ap_ready      (AP_READY),
  .accel_x       (ACCEL_X),
  //.accel_y       (ACCEL_Y),
  .yaw         (YAW),
  .start_point     (START_POINT),
  .end_point       (END_POINT),
  .next_dirc       (NEXT_DIRC),
  .next_dirc_ap_vld (NEXT_DIRC_AP_VLD)
);
```

```
ap_start_timer ap_start_timer(
  .HCLK(HCLK),
  .HRESETn(HRESETn),
  .ap_start(AP_START)
);
/*
always @(posedge HCLK) begin
  if (ram_write) begin
    ram[ram_addr] = ram_wdata;
  end
  ram[ADD_AP_DONE][0]          = AP_DONE;
  ram[ADD_AP_IDLE][0]          = AP_IDLE;
  ram[ADD_AP_READY][0]         = AP_READY;
  ram[ADD_NEXT_DIRC]           = NEXT_DIRC;
  ram[ADD_NEXT_DIRC_AP_VLD][0]   = NEXT_DIRC_AP_VLD;
  ram[ADD_in_ACCEL_X]          = in_ACCEL_X;
  ram[ADD_in_ACCEL_Y]          = in_ACCEL_Y;
  ram[ADD_in_YAW]            = in_YAW;
end
assign ram_rdata = ram[ram_addr];
assign ACCEL_X     = ram[ADD_ACCEL_X];
assign ACCEL_Y     = ram[ADD_ACCEL_Y];
assign YAW       = ram[ADD_YAW];
assign START_POINT  = ram[ADD_START_POINT];
assign END_POINT  = ram[ADD_END_POINT];
*/
endmodule
```

## 8.4 Vivado scripts

```
##############################################################
## This file is generated automatically by Vivado HLS.
## Please DO NOT edit it.
## Copyright (C) 2014 Xilinx Inc. All rights reserved.
##############################################################
open_project dirct_new_3
set_top top
add_files new_dir.c
add_files -tb new_dir_tb.c
open_solution "solution2"
set_part {xq7z045rf900-2i}
create_clock -period 20 -name default
source "./directives.tcl"
csim_design -setup
csynth_design
cosim_design
export_design -format ip_catalog
```

```
#set_directive_unroll -factor 2 "dijkstra/loop1"
set_directive_loop_flatten "dijkstra/loop1"
set_directive_loop_flatten "dijkstra/loop3"
#set_directive_inline -region -recursive top
```

## 8.5 design compiler and encounter scripts

```
    # Use multiple cores
set_host_options -max_cores 16

#####################################
# Setup library                     # No modifications needed
#####################################

set lib_path ". /group/guyeon/cktcad/kits/arm/arm/tsmc/cln40g"

# allows files to be read in without specifying the directory path
set search_path ". /project/linuxlab/synopsys/syncore/libraries /project/linuxlab/
     synopsys/syncore/minpower/syn /project/linuxlab/synopsys/syncore/dw/syn_ver /project
     /linuxlab/synopsys/syncore/dw/sin_ver /project/linuxlab/cadence/vendors/VTVT/
     vtvt_tsmc180/Synopsys_Libraries/libs"
```

```
# Technology cell symbol library
set symbol_lib "vtvt_tsmc180.sdb"

# Technology cell files
set target_library "vtvt_tsmc180.db"

# Used during design linking
set link_library "* dw_foundation.sldb vtvt_tsmc180.db"

# Library of designware components
set synthetic_library "dw_foundation.sldb"

# Directory where DC placed intermediate files
define_design_lib WORK -path ./WORK

# removing high drive inverter
# set_dont_use inv_4

#######################################
# Read in Verilog Source Files        #
#######################################

# Give the list of your verilog files
set my_verilog_files [list AHB_RAM_wraper.v ap_start_timer.v CORTEXM0DS.v
    cortexm0ds_logic.v my_testbench.v top.v top_calculateDir.v top_dijkstra.v]

set my_verilog_files2 [list ./parkinggate/components/i_i2c/src/DW_apb_i2c_cc_constants.v
     ./parkinggate/components/i_i2c/src/DW_apb_i2c_bcm_params.v ./parkinggate/components
    /i_i2c/src/DW_apb_i2c_bcm57.v ./parkinggate/components/i_i2c/src/DW_apb_i2c_bcm21.v
    ./parkinggate/components/i_i2c/src/DW_apb_i2c_bcm41.v ./parkinggate/components/i_i2c
    /src/DW_apb_i2c_bcm06.v ./parkinggate/components/i_i2c/src/DW_apb_i2c_biu.v ./
    parkinggate/components/i_i2c/src/DW_apb_i2c_regfile.v ./parkinggate/components/i_i2c
    /src/DW_apb_i2c_fifo.v ./parkinggate/components/i_i2c/src/DW_apb_i2c_intctl.v ./
    parkinggate/components/i_i2c/src/DW_apb_i2c_rx_filter.v ./parkinggate/components/
    i_i2c/src/DW_apb_i2c_clk_gen.v ./parkinggate/components/i_i2c/src/
    DW_apb_i2c_mstfsm.v ./parkinggate/components/i_i2c/src/DW_apb_i2c_sync.v ./
    parkinggate/components/i_i2c/src/DW_apb_i2c_slvfsm.v ./parkinggate/components/i_i2c/
    src/DW_apb_i2c_tx_shift.v ./parkinggate/components/i_i2c/src/DW_apb_i2c_rx_shift.v .
    /parkinggate/components/i_i2c/src/DW_apb_i2c_toggle.v ./parkinggate/components/i_i2c
    /src/DW_apb_i2c_dma.v ./parkinggate/components/i_i2c/src/DW_apb_i2c.v ./parkinggate/
    components/i_i2c/src/DW_apb_i2c-undef.v ./parkinggate/components/i_apb/src/
    DW_amba_constants.v ./parkinggate/components/i_apb/src/DW_apb_cc_constants.v ./
    parkinggate/components/i_apb/src/DW_apb_constants.v ./parkinggate/components/i_apb/
    src/DW_apb_dcdr.v ./parkinggate/components/i_apb/src/DW_apb_slcr.v ./parkinggate/
    components/i_apb/src/DW_apb_ahbsif.v ./parkinggate/components/i_apb/src/
    DW_apb_prdmux.v ./parkinggate/components/i_apb/src/DW_apb_psel.v ./parkinggate/
    components/i_apb/src/DW_apb_deslcr.v ./parkinggate/components/i_apb/src/DW_apb_mux.v
     ./parkinggate/components/i_apb/src/DW_apb_rdretime.v ./parkinggate/components/i_apb
    /src/DW_apb.v ./parkinggate/components/i_apb/src/DW_apb-undef.v ./parkinggate/
    components/i_ahb/src/DW_amba_constants.v ./parkinggate/components/i_ahb/src/
    DW_ahb_cc_constants.v ./parkinggate/components/i_ahb/src/DW_ahb_constants.v ./
    parkinggate/components/i_ahb/src/DW_ahb_bcm_params.v ./parkinggate/components/i_ahb/
    src/DW_ahb_bcm02.v ./parkinggate/components/i_ahb/src/DW_ahb_bcm01.v ./parkinggate/
    components/i_ahb/src/DW_ahb_bcm53.v ./parkinggate/components/i_ahb/src/
    DW_ahb_arbif.v ./parkinggate/components/i_ahb/src/DW_ahb_dcdr.v ./parkinggate/
    components/i_ahb/src/DW_ahb_dfltslv.v ./parkinggate/components/i_ahb/src/
    DW_ahb_ebt.v ./parkinggate/components/i_ahb/src/DW_ahb_gctrl.v ./parkinggate/
    components/i_ahb/src/DW_ahb_mask.v ./parkinggate/components/i_ahb/src/DW_ahb_mux.v .
    /parkinggate/components/i_ahb/src/DW_ahb_cntc.v ./parkinggate/components/i_ahb/src/
    DW_ahb_gating.v ./parkinggate/components/i_ahb/src/DW_ahb_wtps.v ./parkinggate/
    components/i_ahb/src/DW_ahb_arb.v ./parkinggate/components/i_ahb/src/DW_ahb.v ./
    parkinggate/components/i_ahb/src/DW_ahb-undef.v ./parkinggate/src/parking.v]
# Set the top module of your design
set my_toplevel top_wraper

# Translates HDL to intermediate format
analyze -f verilog $my_verilog_files
analyze -f verilog $my_verilog_files2
# This command does the same work of analyze+elaborate
# read_verilog $my_verilog_files
# read_verilog /home/warehouse/yitongding/Documents/ese566/dirdecider_DC/parkinggate/
    export/parking.v

# Buids generic technology database
elaborate $my_toplevel
```

```
# Designate the design to synthesize
current_design $my_toplevel

##########################
## Verilog (?) Compiler settings # No modifications needed
##########################

# to make DC not use the assign statement in its output netlist
set verilogout_no_tri true

# assume this means DC will ignore the case of the letters in net and module names
#set verilogout_ignore_case true

# unconnected nets will be marked by adding a prefix to its name
set verilogout_unconnected_prefix "UNCONNECTED"

# show unconnected pins when creating module ports
set verilogout_show_unconnected_pins true

# make sure that vectored ports don't get split up into single bits
set verilogout_single_bit false

# generate a netlist without creating an EDIF schematic
set edifout_netlist_only true

#######################
# Define constraints
#######################

# set the clock period in ps
set CLK_PERIOD 50000

# setting the approximate skew
set CLK_SKEW [expr 0.025 * $CLK_PERIOD]

# constraint design area units depends on the technology library
# set MAX_AREA 20000.0
# set_max_area $MAX_AREA

# power constraints
# set MAX_LEAKAGE_POWER 0.0
# set_max_leakage_power $MAX_LEAKAGE_POWER
# set MAX_DYNAMIC_POWER 0.0
# set_max_dynamic_power $MAX_DYNAMIC_POWER

# make sure ports aren't connected together
set_fix_multiple_port_nets -all

# setting the port of clock
create_clock -period  $CLK_PERIOD HCLK

## Design Rule Constraints

set DRIVINGCELL inv_1
set DRIVE_PIN {Y}
# set input driving cell strength / Max fanout for all design
set_driving_cell -lib_cell $DRIVINGCELL -pin $DRIVE_PIN [all_inputs]

# largest fanout allowed
#set MAX_FANOUT 8
#set_max_fanout $MAX_FANOUT

# models load on output ports
set_load $MAX_OUTPUT_load [all_outputs]
# incase of variable load at each output port
# set_load <loadvalue> [get_ports {<portnames>}]


# set maximum and minimum capacitance
# set_max_capacitance
# set_min_capacitance

# setting operating conditions if allowed by technology library
# set_operating_conditions

# wireload models
# set_wireload_model
```

```
# set_wireload_mode

set MAX_INPUT_DELAY 0.9
set MIN_INPUT_DELAY 0
set OUTPUT_MAX_DELAY 0.4
set OUTPUT_MIN_DELAY -0.4

# models the delay from signal source to design input port
# set_input_delay

# models delay from design to output port
# set_output_delay

# used when you are translating some netlist from one technology to another
link

# used to generate separate instances within the netlist
uniquify

###########################
# Design Compiler settings #
###########################

# completely flatten the hierarchy to allow optimization to cross hierarchy boundaries
ungroup -flatten -all

# check internal DC representation for design consistency
check_design

# verifies timing setup is complete
check_timing

# enable DC ultra optimizations
compile_ultra

# verifies timing setup is complete
check_timing

# report design size and object counts
report_area

# reports design database constraints attributes
report_timing_requirements

################
# Output files #
################

# save design
set filename [format "%s%s"  $my_toplevel ".ddc"]
write -format ddc -hierarchy -output $my_toplevel

# save delay and parasitic data
set filename [format "%s%s"  $my_toplevel ".sdf"]
write_sdf -version 1.0 $filename

# save synthesized verilog netlist
set filename [format "%s%s"  $my_toplevel ".syn.v"]
write -format verilog -hierarchy -output $filename

# this file is necessary for P&R with Encounter
set filename [format "%s%s"  $my_toplevel ".sdc"]
write_sdc $filename

# write milkyway database
if {[shell_is_in_topographical_mode]} {
    write_milkyway -output $my_toplevel -overwrite
}

redirect [format "%s%s" $my_toplevel  _design.repC] { report_design }
redirect [format "%s%s" $my_toplevel  _area.repC] { report_area }
redirect -append [format "%s%s" $my_toplevel  _area.repC] { report_reference }
redirect [format "%s%s" $my_toplevel  _latches.repC] { report_register -level_sensitive
    }
redirect [format "%s%s" $my_toplevel  _flops.repC] { report_register -edge }
redirect [format "%s%s" $my_toplevel  _violators.repC] { report_constraint
    -all_violators }
```

```
redirect [format "%s%s" $my_toplevel _power.repC] { report_power }
redirect [format "%s%s" $my_toplevel _max_timing.repC] { report_timing -delay max
    -nworst 3 -max_paths 20 -greater_path 0 -path full -nosplit}
redirect [format "%s%s" $my_toplevel _min_timing.repC] { report_timing -delay min
    -nworst 3 -max_paths 20 -greater_path 0 -path full -nosplit}
redirect [format "%s%s" $my_toplevel _out_min_timing.repC] { report_timing -to [
    all_outputs] -delay min -nworst 3 -max_paths 1000000 -greater_path 0 -path full
    -nosplit}

quit
```

## 8.6 test bench files

```
`timescale 100ns/10ns

module ParkingSpot_tb;

  reg sim_clock;                          // System clock
reg power_on_reset_n;                    // Power-on reset signal

localparam clk_period = 50;             // Simulation cycles per clock period
localparam por_delay  = 1010;            // Simulation cycles of power-on-reset
localparam one_sec    = 10**6;
//-----------------------------------------------------------------------------
// Generate system clock, power-on reset and synchronized AHB reset signals
//-----------------------------------------------------------------------------

// Generate a clock of the appropriate period
initial
  #0 sim_clock = 1'b0;

always
  #(clk_period/2) sim_clock <= ~sim_clock;

 // Release the active-low power-on reset signal after the given delay
 initial begin
  #0 power_on_reset_n = 1'b0;
  #(por_delay) power_on_reset_n = 1'b1;
end

wire HCLK;
wire HRESETn;

assign HCLK    = sim_clock;
assign HRESETn = power_on_reset_n;

reg [31:0]  ACCEL_X;
reg [31:0]  YAW;
reg [31:0]  start_point;
reg [31:0]  end_point;

wire   ap_start;
wire   ap_done;
wire   ap_idle;
wire   ap_ready;
wire   [31:0] next_dirc;
wire   next_dirc_ap_vld;

top DUT1(
    .ap_clk(HCLK),
    .ap_rst(!HRESETn),
    .ap_start(ap_start),
    .ap_done(ap_done),
    .ap_idle(ap_idle),
    .ap_ready(ap_ready),
    .accel_x(accel_x),
    .yaw(yaw),
    .start_point(start_point),
    .end_point(end_point),
    .next_dirc(next_dirc),
    .next_dirc_ap_vld(next_dirc_ap_vld)
);
```

```
ap_start_timer DUT2(
  .HCLK(HCLK),
  .HRESETn(HRESETn),
  .ap_start(ap_start)
);

  initial begin
  $dumpfile("ParkingSpot.vcd");
  $dumpvars(0,ParkingSpot_tb);

  #(por_delay);
  start_point = 20;
  end_point = 7;

  // first go straight

  ACCEL_X = 1;
  YAW = 0;

  #10000000

  ACCEL_X = 0;
  YAW = 0;

  #50000000

  ACCEL_X = -100;
  YAW = 0;
//right turn
  #10000000

  ACCEL_X = 0;
  YAW = 90;

// second go
  #10000000

  ACCEL_X = 1;
  YAW = 0;

  #10000000

  ACCEL_X = 0;
  YAW = 0;

  #50000000

  ACCEL_X = -100;
  YAW = 0;

//right turn
  #10000000

  ACCEL_X = 0;
  YAW = 90;


//third go

  #10000000

  ACCEL_X = 1;
  YAW = 0;

  #10000000

  ACCEL_X = 0;
  YAW = 0;

  #10000000

  ACCEL_X =-100;
  YAW = 0;

//turn
  #10000000
```

```
  ACCEL_X = 0;
  YAW = 90;
// forth

  #10000000

  ACCEL_X = 1;
  YAW = 0;

  #20000000

  ACCEL_X = 0;
  YAW = 0;

  #10000000

  ACCEL_X = -100;
  YAW = 0;

//turn
  #10000000

  ACCEL_X = 0;
  YAW = 90;

//five
  #10000000

  ACCEL_X = 1;
  YAW = 0;

  #20000000

  ACCEL_X = 0;
  YAW = 0;

  #10000000

  ACCEL_X = -100;
  YAW = 0;
  $finish;
  end

endmodule
```

## 8.7 Makefile

we did not use any Makefile.