# Computer-Aided VLSI System Design
# Homework 4: IoT Data Filtering

TA: 陳泰融 r11943024@ntu.edu.tw
TA: 蔡宇軒 f07943171@ntu.edu.tw

**Due Tuesday, May. 09, 14:00**

## Data Preparation

1. Decompress 1112_hw4.tar with following command

```
tar -xvf 1112_hw4.tar
```

| Folder | File | Description |
|---|---|---|
| 00_TESTBED | testfixture.v | Testbench for IOTDF. |
| 00_TESTBED/pattern1_data | pattern1.dat | Input IoT data for f1~f9 |
| 00_TESTBED/pattern1_data | f1.dat ~ f9.dat | Output golden for function 1 ~ function 9 |
| 00_TESTBED/pattern2_data | | Put the input data and golden which you generate |
| 01_RTL | IOTDF.v | Your design. |
| 01_RTL | rtl_01.f | File list for RTL simulation |
| 01_RTL | runall_rtl | RTL simulation bash file |
| 02_SYN | .synopsys_dc.setup | Configuration file for DC. |
| 02_SYN | IOTDF_DC.sdc | Constraint file for synthesis. |
| 03_GATE | rtl_03.f | File list for gate-level simulation |
| 03_GATE | run_syn | Gate-level simulation bash file |
| 06_POWER | .synopsys_pt.setup | Configuration file for PrimeTime |
| 06_POWER | pt_script.tcl | PrimeTime power measurement script |
| reports | report.txt | Design report form |

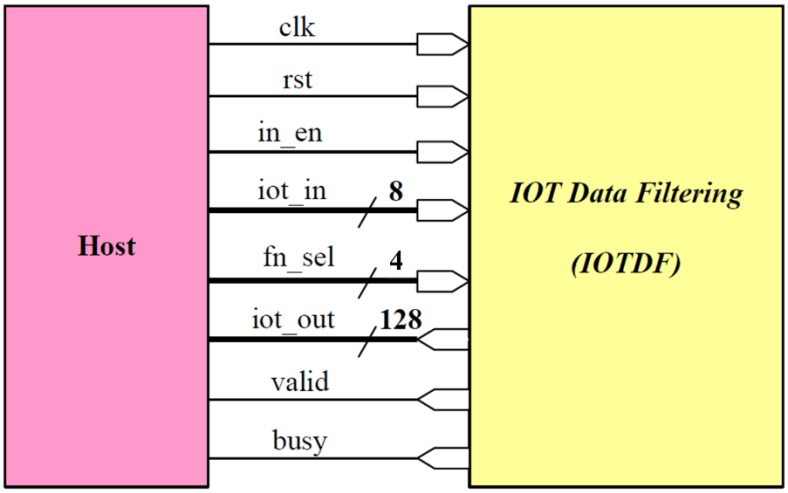
All libraries needed for synthesis, simulation can be found in previous homework.

**Only worst-case library is used in this homework.**

## Introduction

In this homework, you are asked to design a **IoT Data Filtering (IOTDF)**, which can processor large IoT data from the sensors, and output the result in real-time.

## Block Diagram

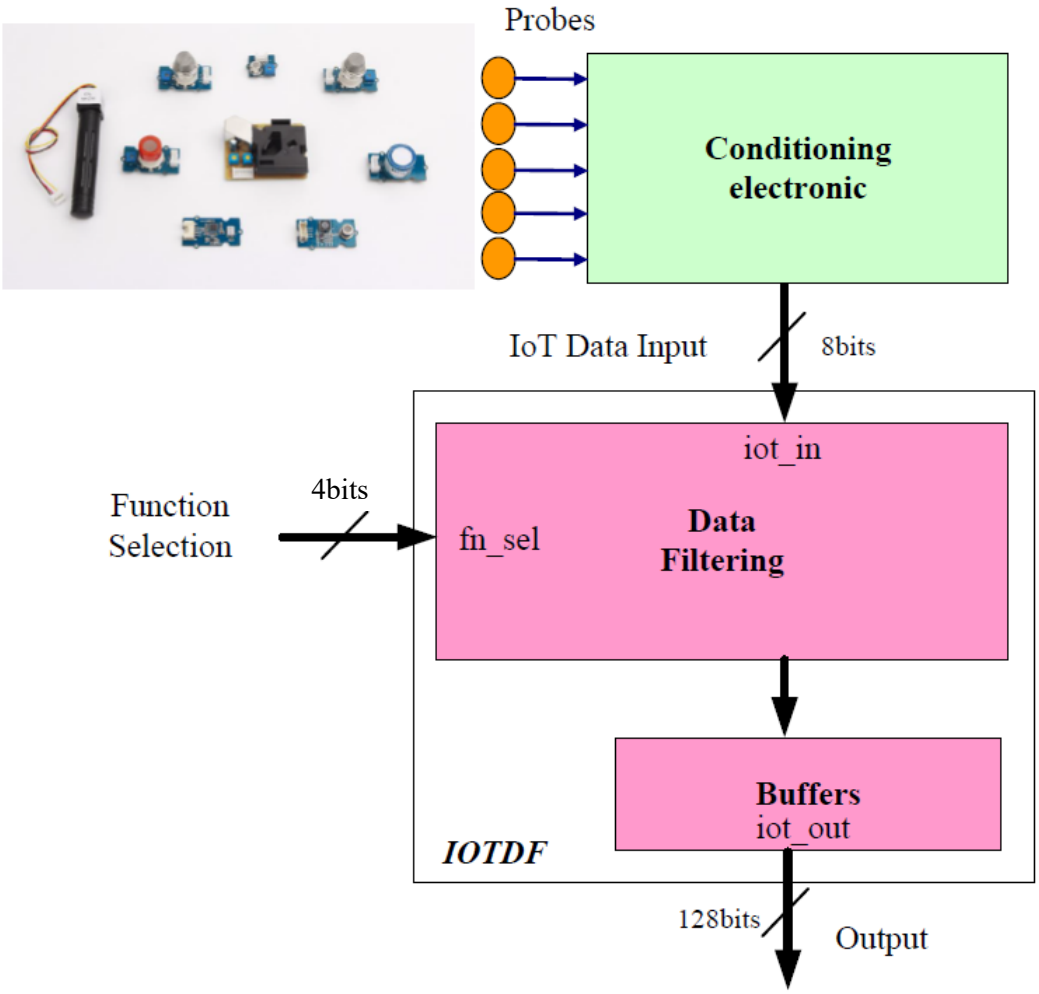



## Specifications

1. Top module name: **IOTDF**
2. Input/output description:

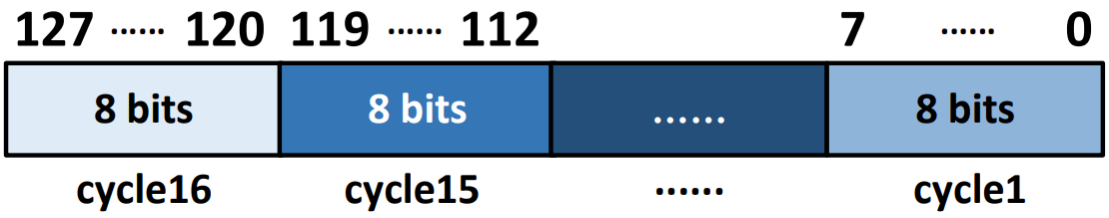
| Signal Name | I/O | Width | Simple Description |
|---|---|---|---|
| clk | I | 1 | Clock signal in the system (positive edge trigger).<br>All inputs are synchronized with the **positive** edge clock.<br>All outputs should be synchronized at clock **rising** edge |
| rst | I | 1 | Active **high** asynchronous reset. |
| in_en | I | 1 | Input enable signal.<br>When **busy** is **low**, **in_en** is turned to **high** for fetching new data. Otherwise, **in_en** is turned to **low** if **busy** is **high**.<br>If all data are received, **in_en** is turned to **low** to the end of the process. |
| iot_in | I | 8 | IoT input signal.<br>Need **16** cycles to transfer one 128-bit data.<br>The number of data is **96** in this homework. |
| fn_sel | I | 4 | Function Select Signal.<br>There are 9 functions supported in IOTDF. For each simulation, only one function is selected for data processing. |

| | | | |
|---|---|---|---|
| iot_out | O | 128 | IoT output signal.<br>One cycle for one data output. |
| busy | O | 1 | IOTDF busy signal<br>(explained in description for in_en) |
| valid | O | 1 | IOTDF output valid signal<br>Set **high** for valid output |




## Design Description

1. The sensor data is a 128-bit unsigned data, which is divided in 16 8-bit partial data for IOTDF fetching. The way for data transferring is as follow. Only 96 data are required to fetch for each function simulation.



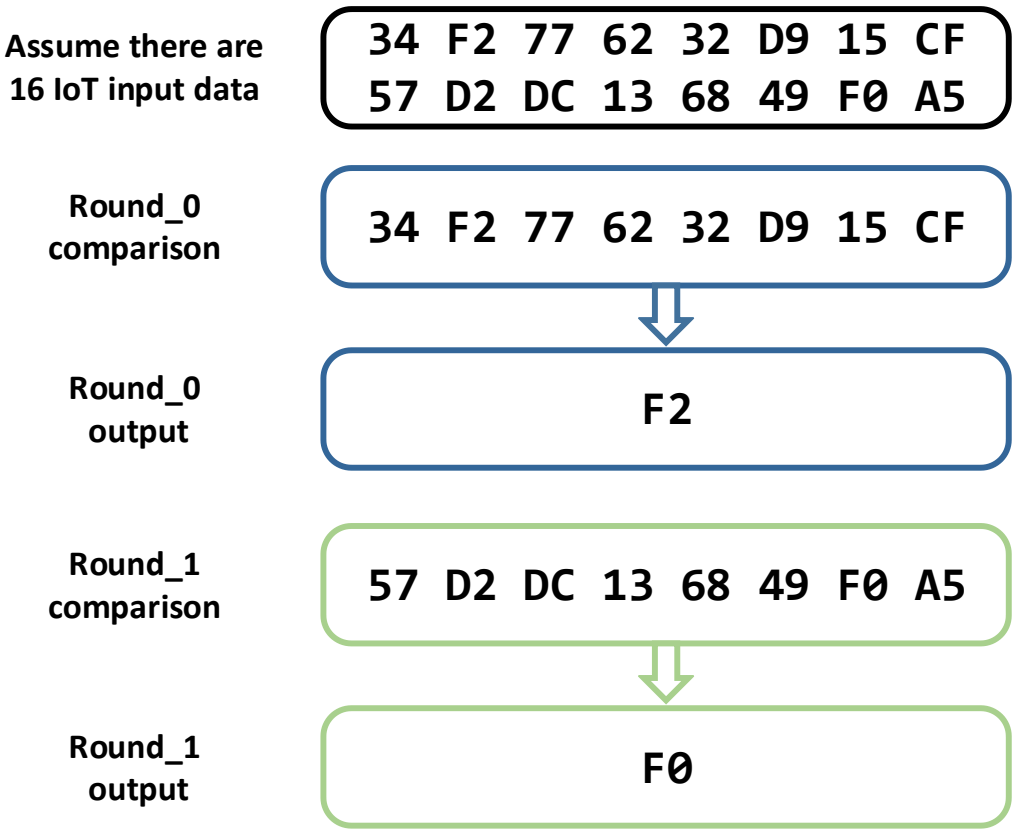
2. Nine functions are asked to design in this homework.

| | Fn_sel | Functions |
|---|---|---|
| F1 | 4'b0001 | Max(N) |
| F2 | 4'b0010 | Min(N) |
| F3 | 4'b0011 | Top2Max(N) |
| F4 | 4'b0100 | Last2Min(N) |
| F5 | 4'b0101 | Avg(N) |
| F6 | 4'b0110 | Extract(low < data < high) |
| F7 | 4'b0111 | Exclude(data<low , high<data) |
| F8 | 4'b1000 | PeakMax(the data is larger than previous output data) |
| F9 | 4'b1001 | PeakMin(the data is smaller than previous output data) |

The details of functions are shown in the following: (8 bit for example)

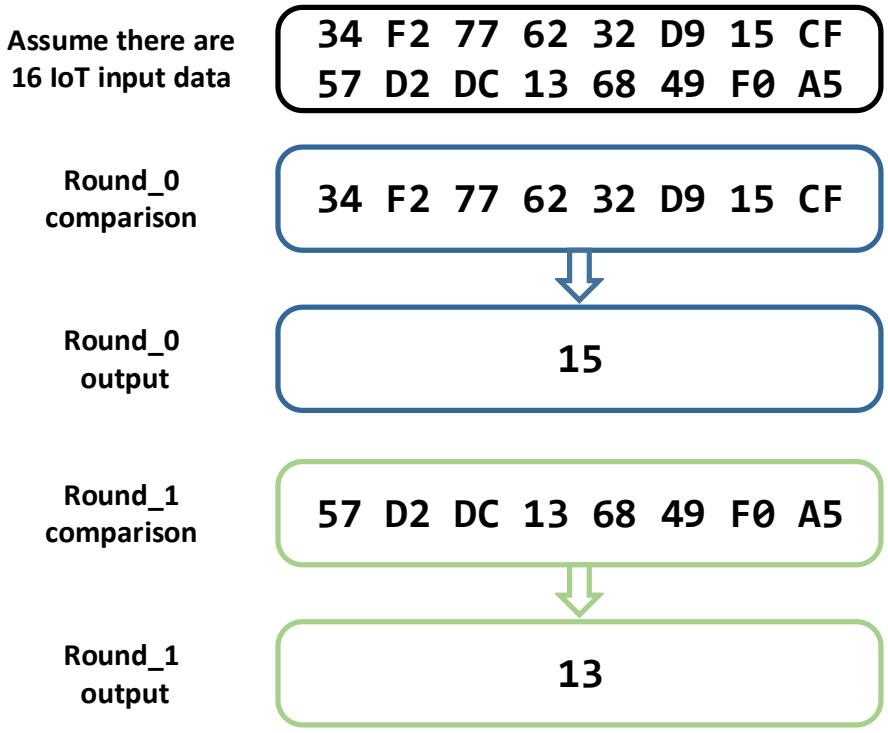
a. F1: Max(N)

- Find the largest data in 8 IoT data for each round.



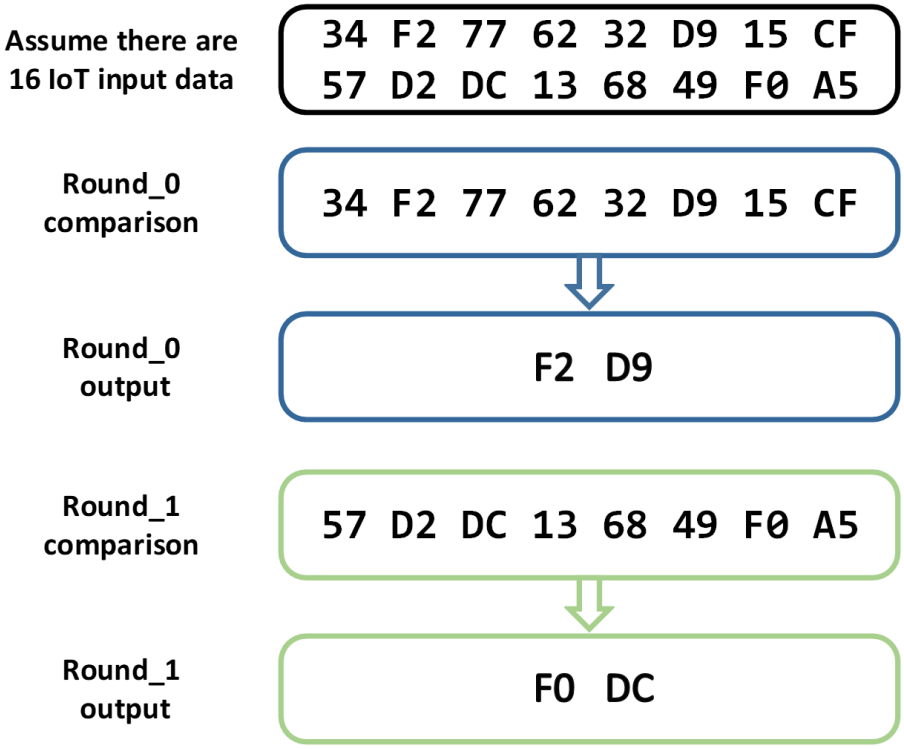
b. F2: Min(N)

- Find the smallest data in 8 IoT data for each round.



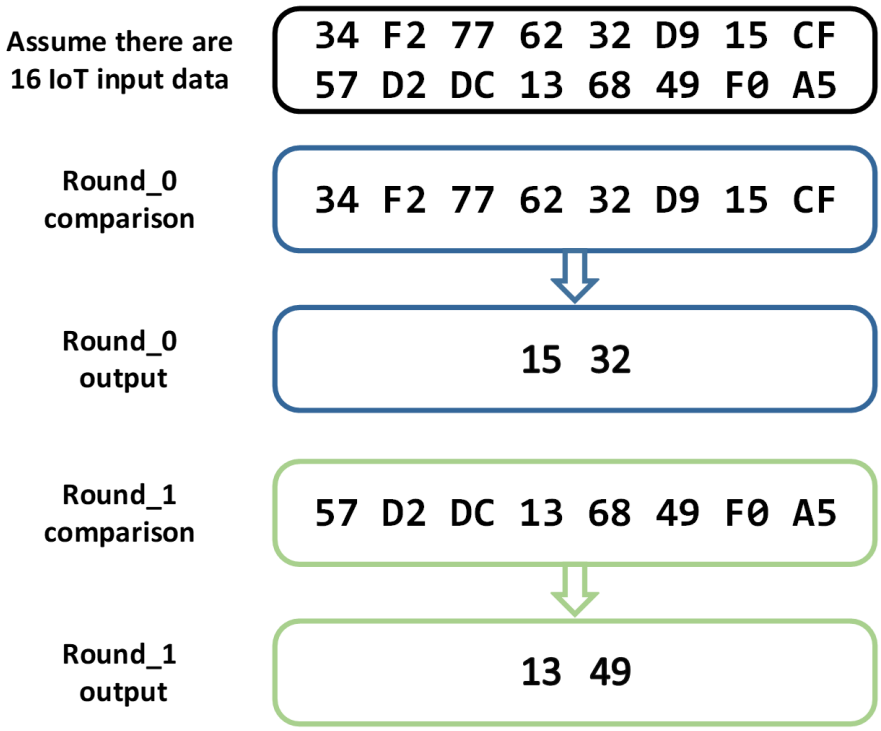

c. F3: Top2Max(N)

- Find the two largest values in 8 IoT data for each round.
- Output the largest first, then output the second largest.



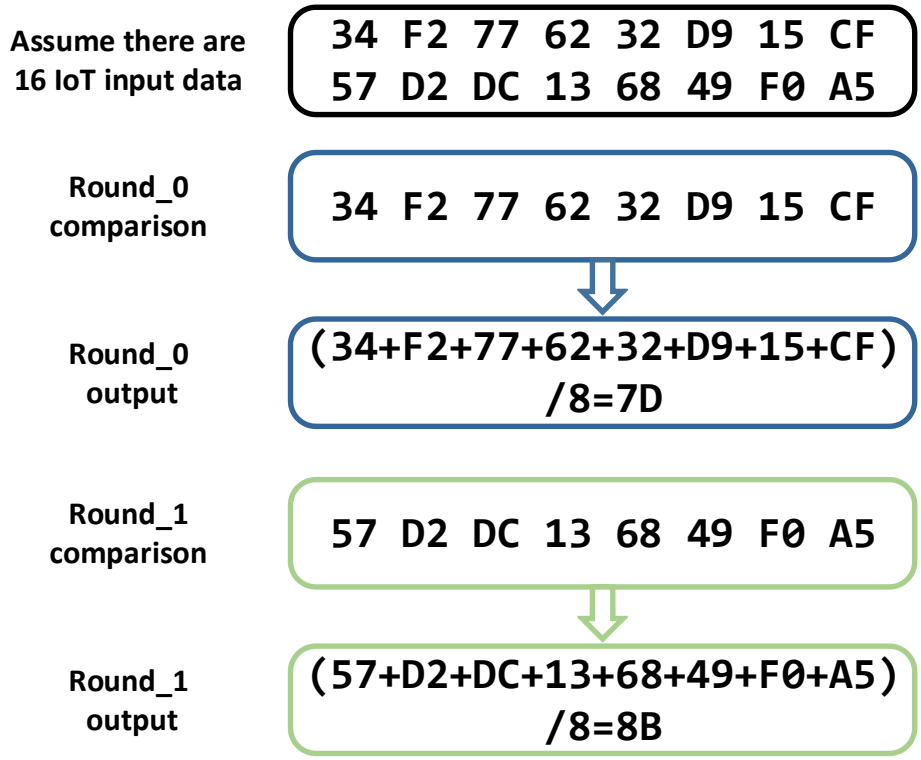
d. F4: Last2Min(N)

- Find the two smallest values in 8 IoT data for each round.
- Output the smallest first, then output the second smallest.



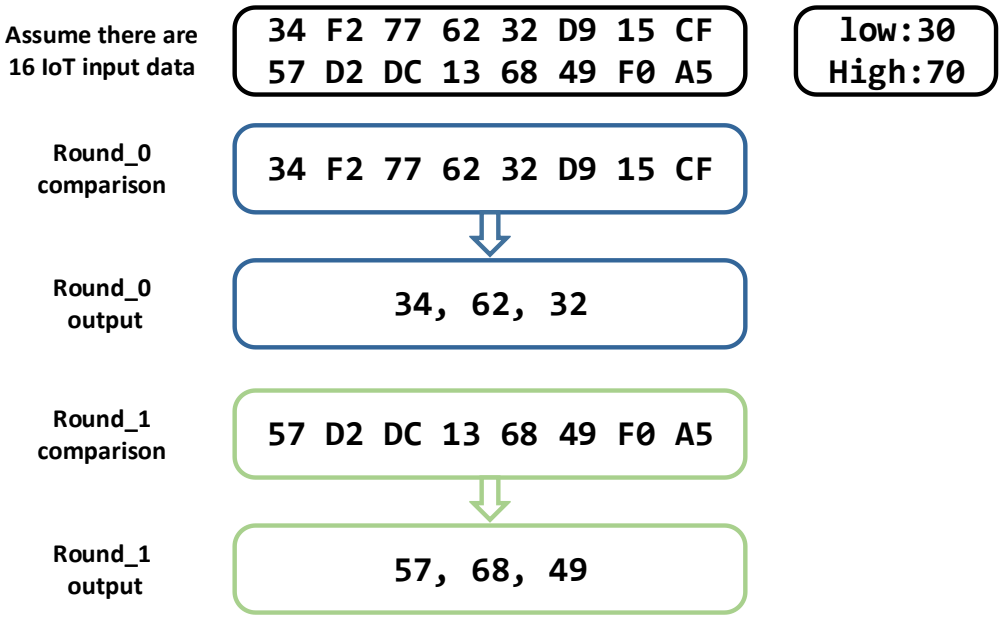

e. F5: Avg(N)

- Find the average in 8 IoT data for each round.
- Round down the output if the result is not integer



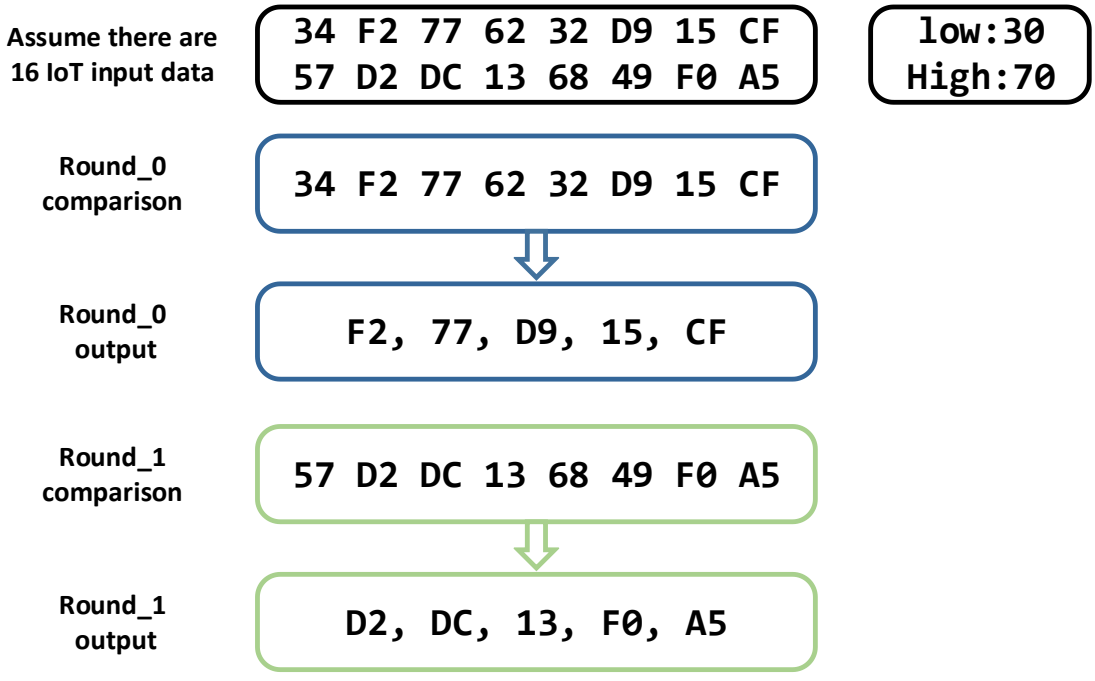
f. F6: Extract(low < data < high)

- Find the data between the known “low” value and the known “high” value.
- For the homework, the “low” and “high” value are set as follow:
  - Low: 128’h 6FFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF
  - High: 128’h AFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF



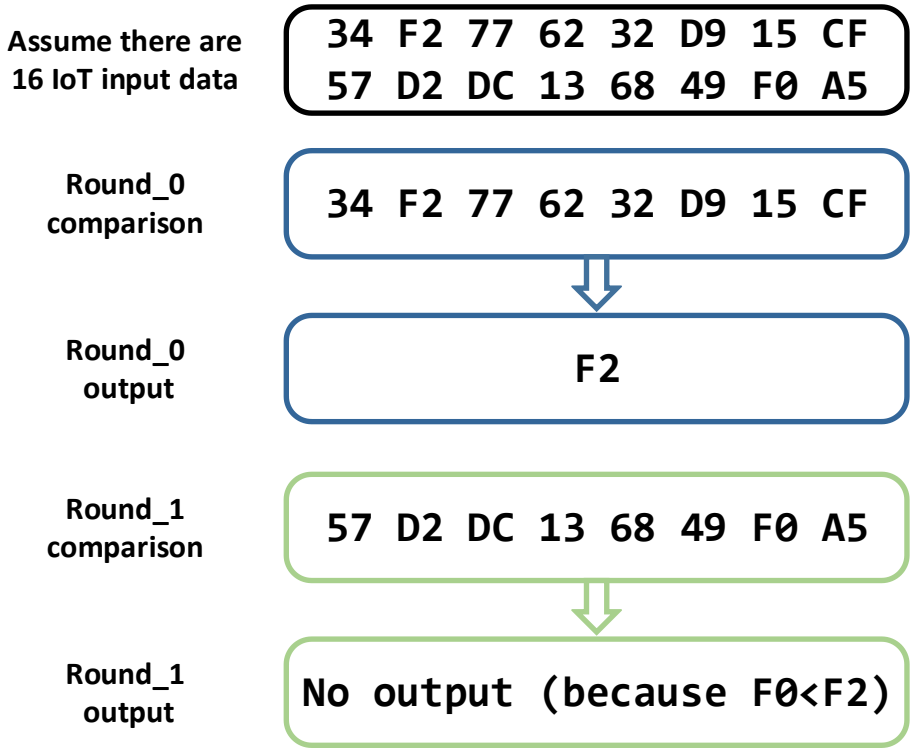

g. F7: Exclude(data<low , high<data)

- Find the data which is smaller than the known “low” value, or larger than the known “high” value.
- For the homework, the “low” and “high” value are set as follow:
  - Low: 128’h 7FFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF
  - High: 128’h BFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF



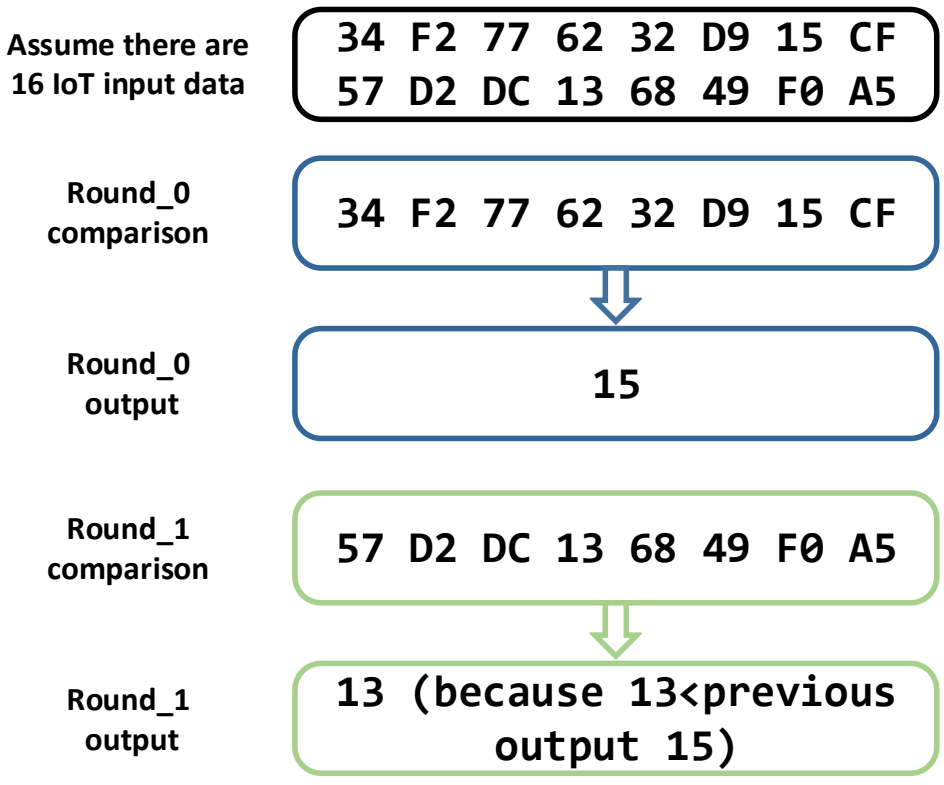
h. F8: PeakMax(the data is larger than previous output data)

- Find the largest data in round_0 first. For the rest of the round, output the data which is larger than previous output data.

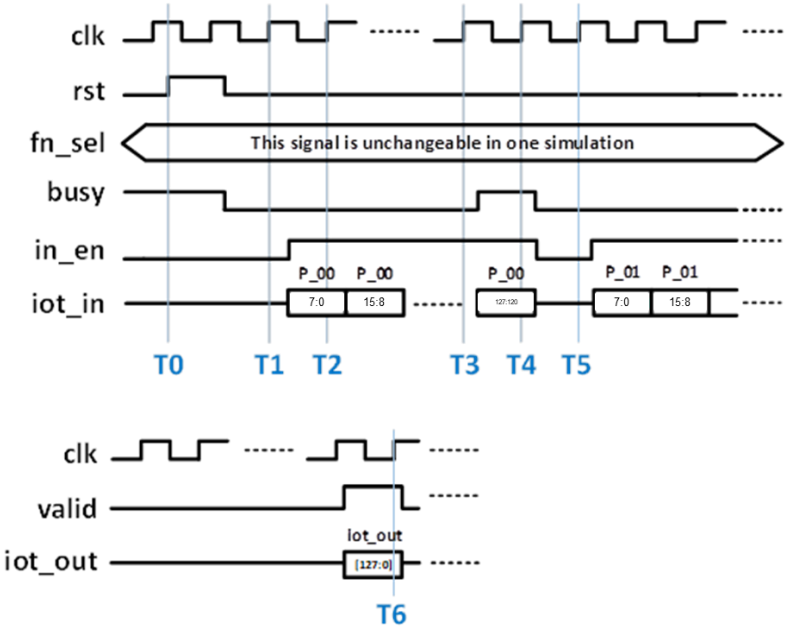



i. F9: PeakMin(the data is smaller than previous output data)

- Find the smallest round_0 first. For the rest of the round, output the data which is smaller than previous output data.



## Timing Diagram




1. IOTDF is initialized between T0~T1.
2. in_en is set to high and start to input IoT data P_00[7:0] if busy is low at T1.
3. in_en is kept to high and input IoT data P_00[15:8] if busy is low at T2.
4. in_en is kept to high and input IoT data P_00[127:120] if busy is low at T3.
5. in_en is set to low and IoT data is set to 0 (stop streaming in data) if busy is high at T4.
6. There are 16 cycles between T1~T4 for one IoT data. You can set busy to high to stop steaming in data if you want.
7. You have to set valid to high if you want to output iot_out.
8. The whole processing time can't exceed 1000000 cycles.

## Hint

1. Clock gating can help reduce the power.
2. With registers optimization/sharing, the area will be much lower.
3. Pipeline can help cut down on process time.

## Submission

1. Create a folder named **studentID_hw4**, and put all below files into the folder

```
r11943024_hw4
        ├── 01_RTL
        │       ├── IOTDF.v (and other verilog files)
        │       └── rtl_01.f (Remember to include all your verilog files)
        ├── 02_SYN
        │       ├── IOTDF_syn.area
        │       └── IOTDF_syn.timing
        ├── 03_GATE
        │       ├── IOTDF_syn.sdf
        │       └── IOTDF_syn.v
        ├── 06_POWER
        │       └── F1_9.power
        └── reports
                └── report.txt
```

Note: Use **lower case** for the letter in your student ID. (Ex. r06943027_hw4)

2. Content of the report.txt

   Record the power and processing time of gate-level simulation.

```
StudentID: r11943024
Clock period: 5.0 (ns)
Area: 30000.00 (um^2)
----------------------------
f1 time: 10016.50 (ns)
f1 power: 0.9197 (mW)
----------------------------
f2 time: 10016.50 (ns)
f2 power: 0.9197 (mW)
----------------------------
f3 time: 10023.00 (ns)
f3 power: 0.9197 (mW)
----------------------------
f4 time: 10023.00 (ns)
f4 power: 0.9197 (mW)
```

```
----------------------------
f5 time: 10016.50 (ns)
f5 power: 0.9197 (mW)
----------------------------
f6 time: 10773.00 (ns)
f6 power: 0.9197 (mW)
----------------------------
f7 time: 10016.50 (ns)
f7 power: 0.9197 (mW)
----------------------------
f8 time: 10003.50 (ns)
f8 power: 0.9197 (mW)
----------------------------
f9 time: 10003.50 (ns)
f9 power: 0.9197 (mW)
```

3. Compress the folder **studentID_hw4** in a **tar file** named **StudentID_hw4_v*k*.tar** (***k* is the number of version, *k* =1,2,…)**

```
tar -cvf StudentID_hw4_vk.tar StudentID_hw4
```

   TA will only check the last version of your homework.

4. Submit to NTU cool

## Grading Policy

1. TA will use **runall_rtl** and **runall_syn** to run your code at RTL and gate-level

simulation.

2. Simulation (60%)

| | Score |
|---|---|
| RTL simulation | 40% |
| Gate-level simulation | 20% |
| Hidden pattern (Gate-level) | 10% |

3. Performance (40%)

Score for performance to be considered:

**Score = (Power1 × Time1 + … + Power9 × Time9)*Area**

**Unit: power(mW), Time(ns), Area($um^2$)**

Baseline = $3.5*10^9$

Caution: Need to pass hidden to get the score of this part

| | Score |
|---|---|
| Baseline | 10% |
| Ranking (Need to pass Baseline) | 20% |

4. Precise explanation of the terms **Power**, **Time**, and **Area**

- Area: Cell area from synthesis report (ex. 16590ns below)

```
Library(s) Used:

    slow (File: /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db/slow.db)

Number of ports:                          815
Number of nets:                          3171
Number of cells:                         2823
Number of combinational cells:           2209
Number of sequential cells:               513
Number of macros/black boxes:               0
Number of buf/inv:                        442
Number of references:                     172

Combinational area:              15261.323552
Buf/Inv area:                     1534.449575
Noncombinational area:           12993.597111
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 28254.920663
Total area:                 undefined
```

- Time: processing time from simulation (ex. 16590ns below)

```
P04:  ** Correct!! ** , iot_out=ea11441ea823a0ade3852d25c71f750a
P05:  ** Correct!! ** , iot_out=eeccfe0ea62b02ad92c025e86bd303db
P06:  ** Correct!! ** , iot_out=d495a168f2987fa5e06673e0f67f6028
P07:  ** Correct!! ** , iot_out=fd799525a5793d4c74d1b81b5df28d34
P08:  ** Correct!! ** , iot_out=df5e9b3eaaf1de60ef68672fe6d01dcc
P09:  ** Correct!! ** , iot_out=e8314ae60ff5850bb1feccdf549dc780
P10:  ** Correct!! ** , iot_out=f6ca503e3ecabf188b00992f75a29e03
P11:  ** Correct!! ** , iot_out=cd7944c200cea18c4eab0328c544fc0f

----------------------------------------------------

Congratulations! All data have been generated successfully!

Total cost time:  10016.50 ns
-------------------------PASS-------------------------
```

- Power: Use below command to analyze the power. (Need to source the following .cshrc file first!) (ex. 0.8326 mW below)

```
Unix% source /usr/cad/synopsys/CIC/primetime.cshrc
Unix% pt_shell -f ./pt_script.tcl | tee pp.log
```

```
  Net Switching Power  = 1.200e-04   (14.41%)
  Cell Internal Power  = 6.971e-04   (83.73%)
  Cell Leakage Power   = 1.548e-05   ( 1.86%)
                         ---------
Total Power            = 8.326e-04  (100.00%)

X Transition Power     = 1.812e-06
Glitching Power        =    0.0000

Peak Power             =    0.3794
Peak Time              =     6.500
```

## Reference

[1] IC Design Contest, 2019.