Computer-Aided VLSI System Design

Homework 4: IoT Data Filtering

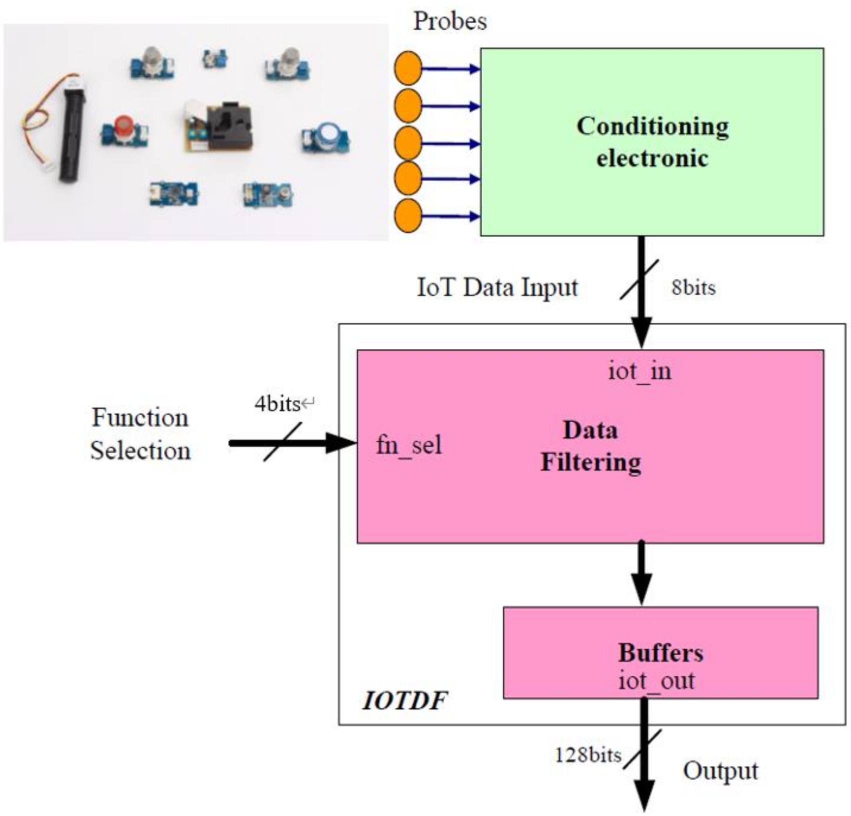
Graduate Institute of Electronics Engineering, National Taiwan University

Goal

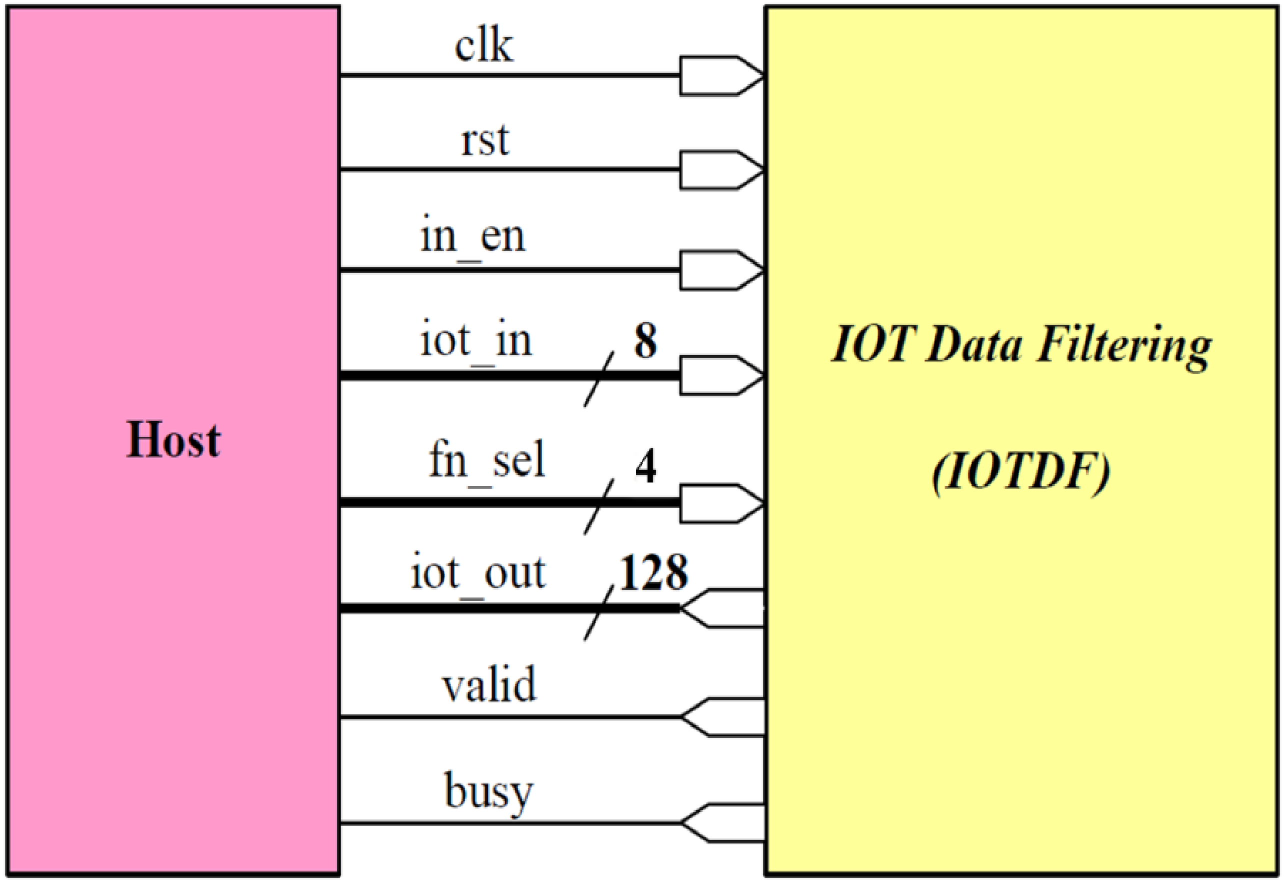
- In this homework, you will learn
  - Generate patterns for testing
  - Optimizing the trade-off between power consumption, execution speed, and required area
  - Use primetime to estimate power
  - Learn to design a suitable architecture for processing data with long bit lengths

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.

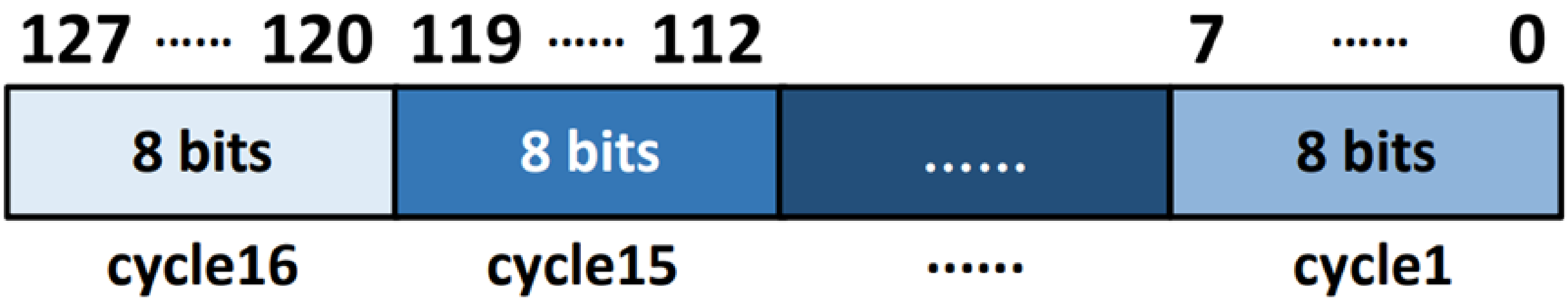


Block Diagram



Design Description

- 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.

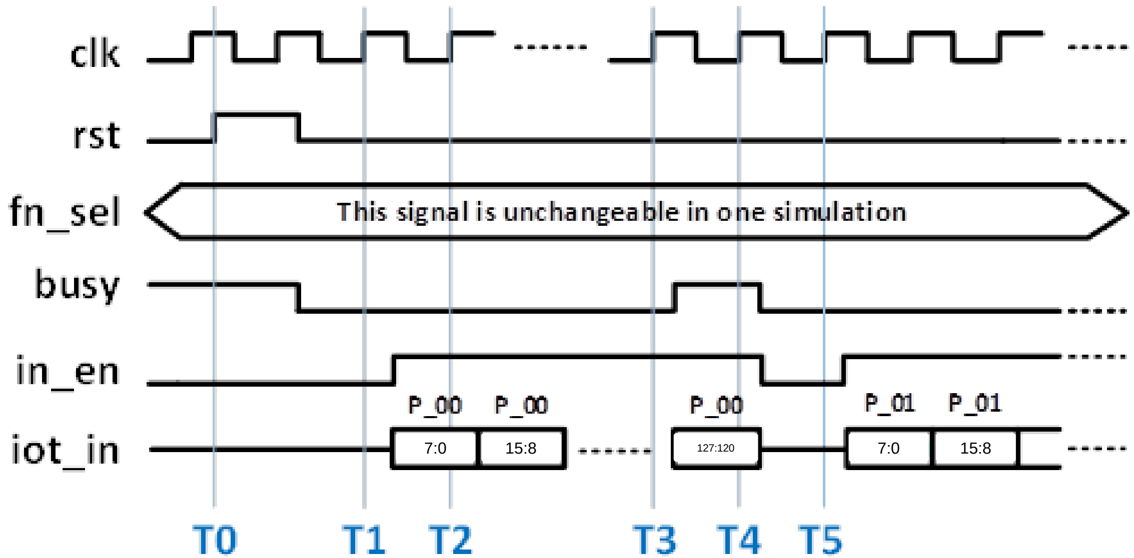


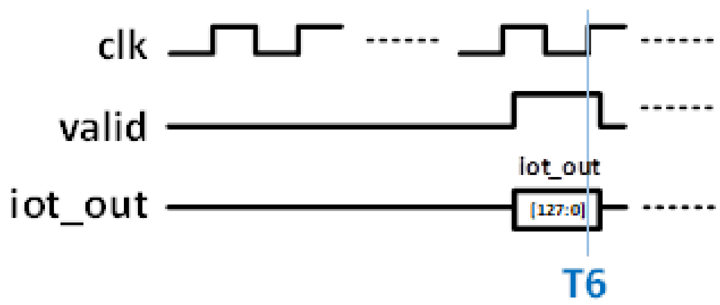
Input/Output

Signal Name	I/O	Width	Simple Description
clk	I	1	Clock signal in the system (positive edge trigger). All inputs are synchronized with the positive edge clock. All outputs should be synchronized at clock rising edge
rst	I	1	Active high asynchronous reset.
in_en	I	1	Input enable signal. 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. If all data are received, in_en is turned to low to the end of the process.
iot_in	I	8	IoT input signal. Need 16 cycles to transfer one 128-bit data. The number of data is 96 in this homework.
fn_sel	I	4	Function Select Signal. 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. One cycle for one data output.
busy	O	1	IOTDF busy signal (explained in description for in_en)
valid	O	1	IOTDF output valid signal Set high for valid output

Specification (1)

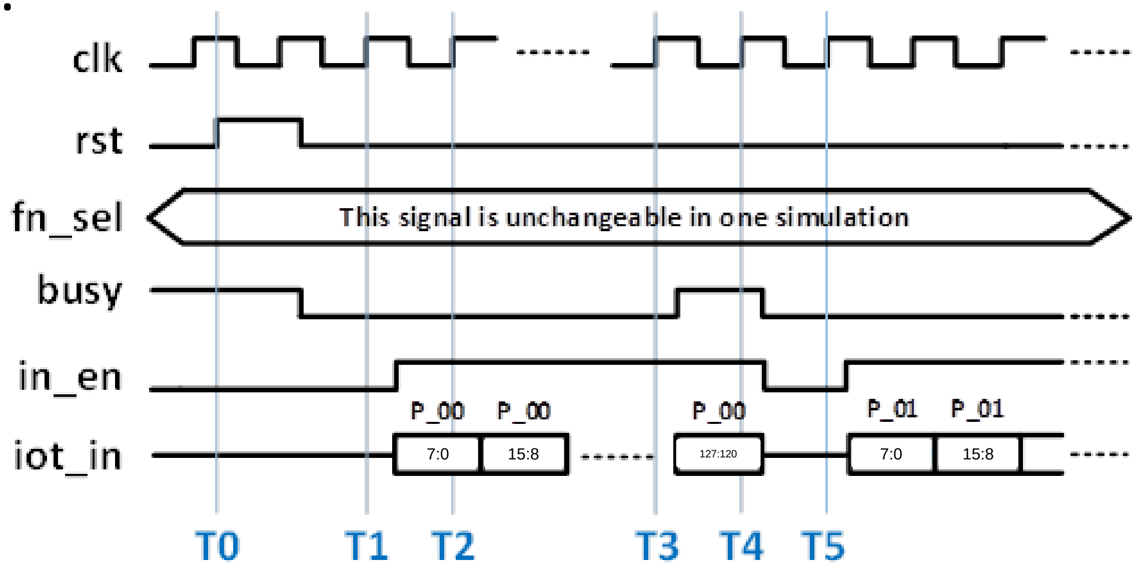
- IOTDF is initialized between T0~T1..

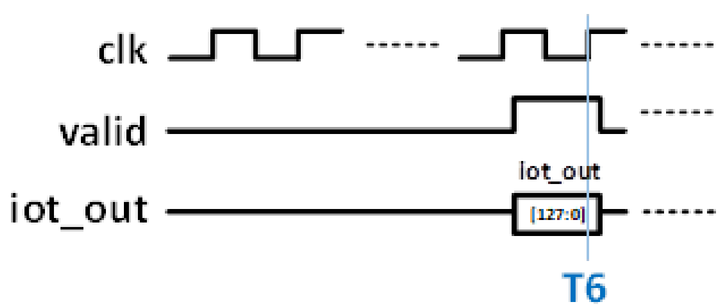




Specification (2)

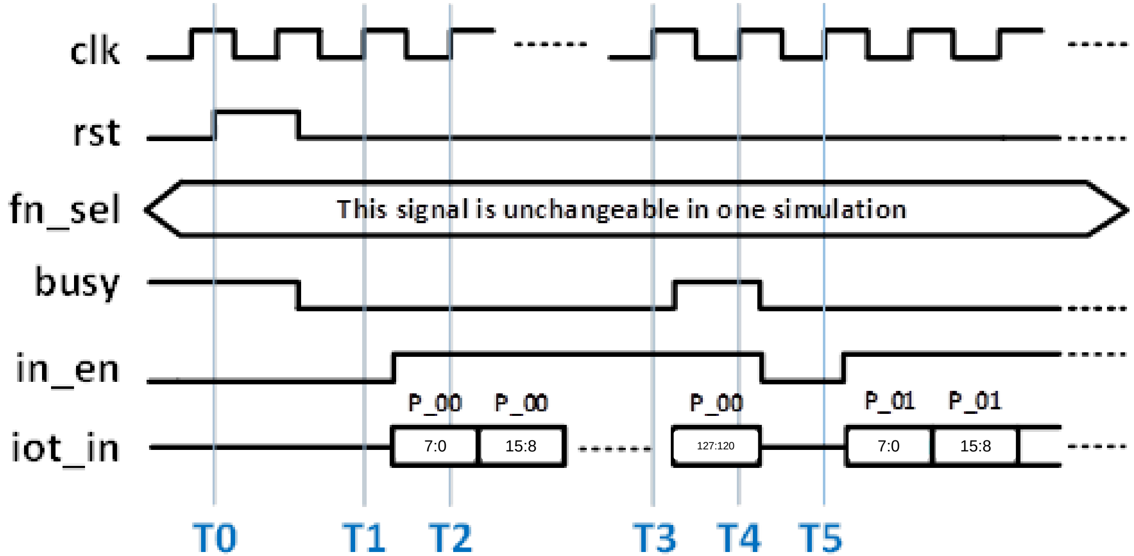
- in_en is set to high and start to input IoT data P_00[7:0] if busy is low at T1.

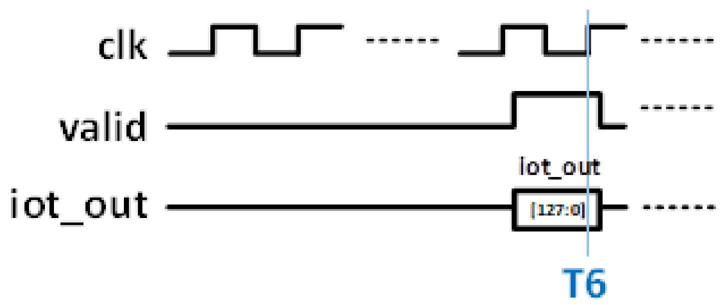




Specification (3)

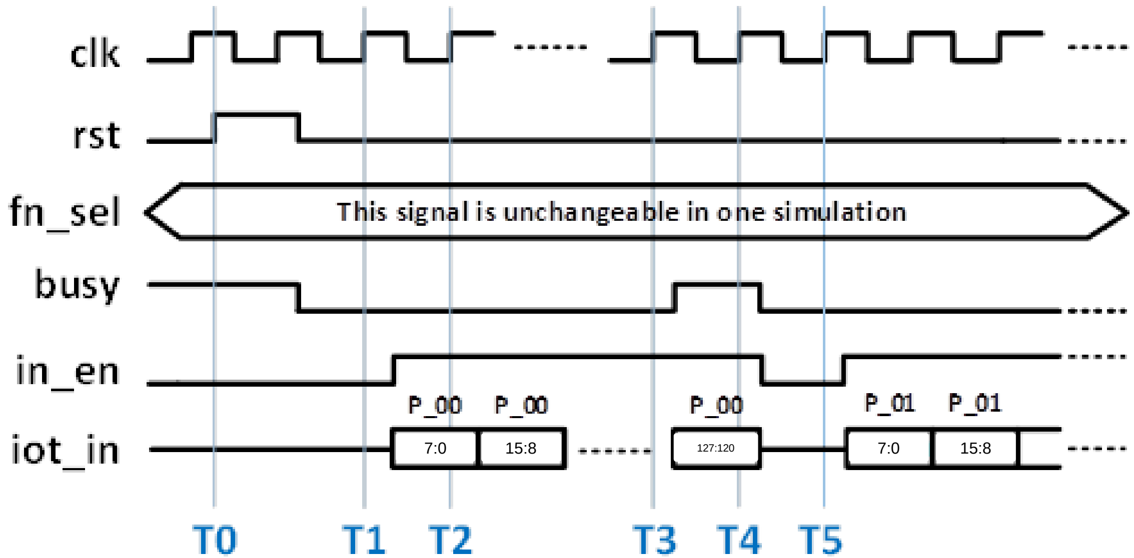
- in_en is kept to high and input IoT data P_00[15:8] if busy is low at T2.

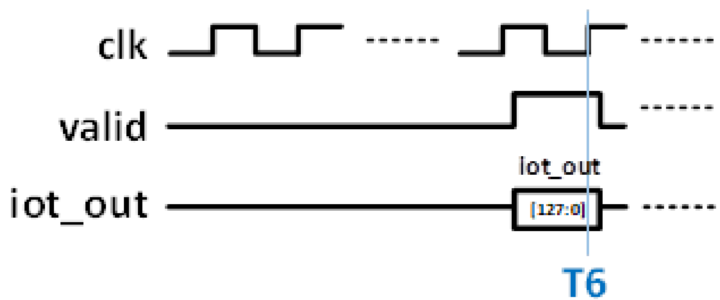




Specification (4)

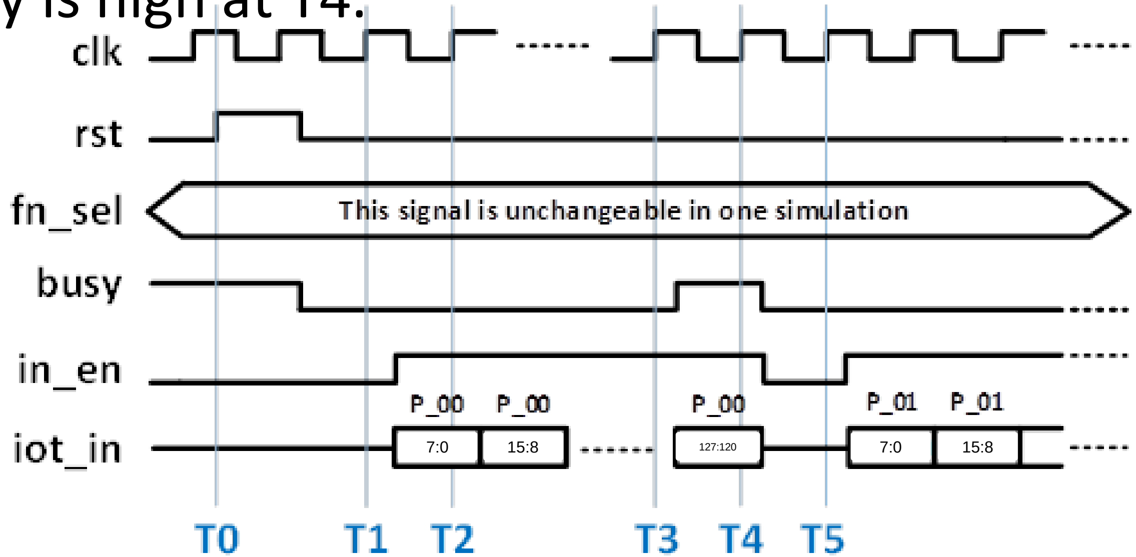
- in_en is kept to high and input IoT data P_00[127:120] if busy is low at T3.

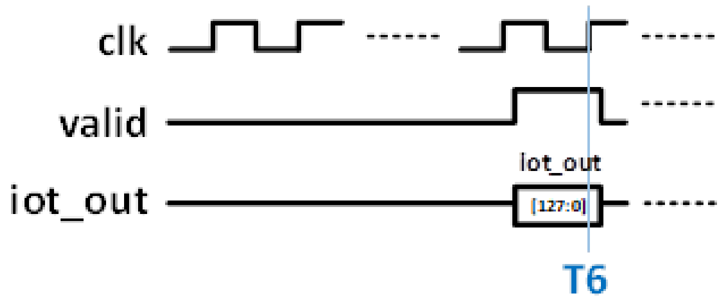




Specification (5)

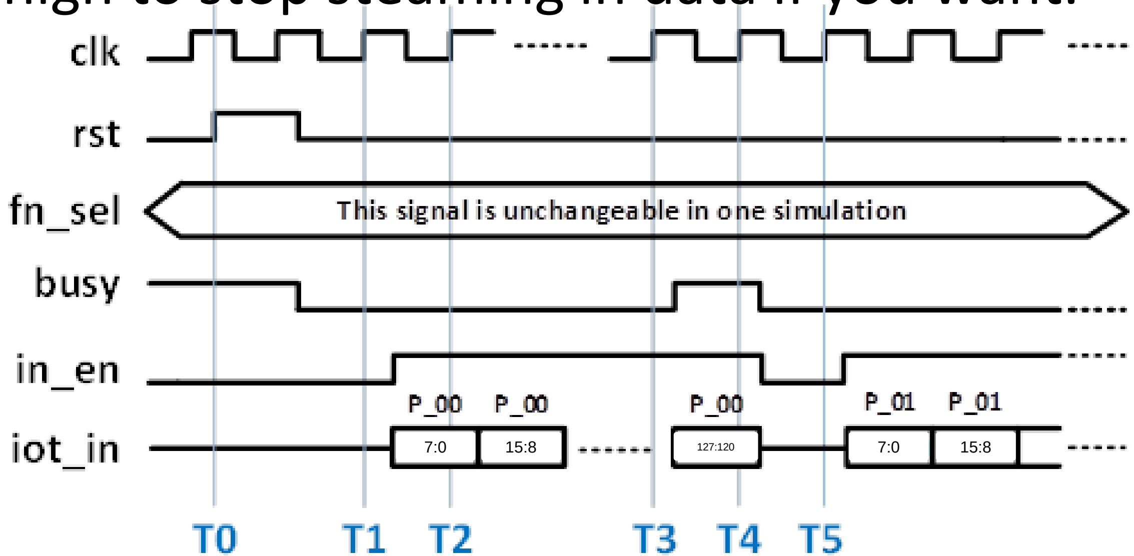
- in_en is set to low and IoT data is set to 0 (stop streaming in data) if busy is high at T4.

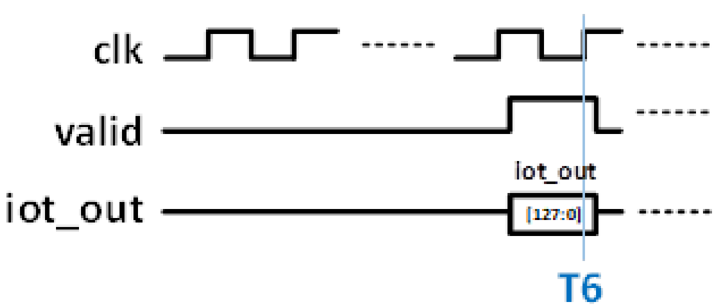




Specification (6)

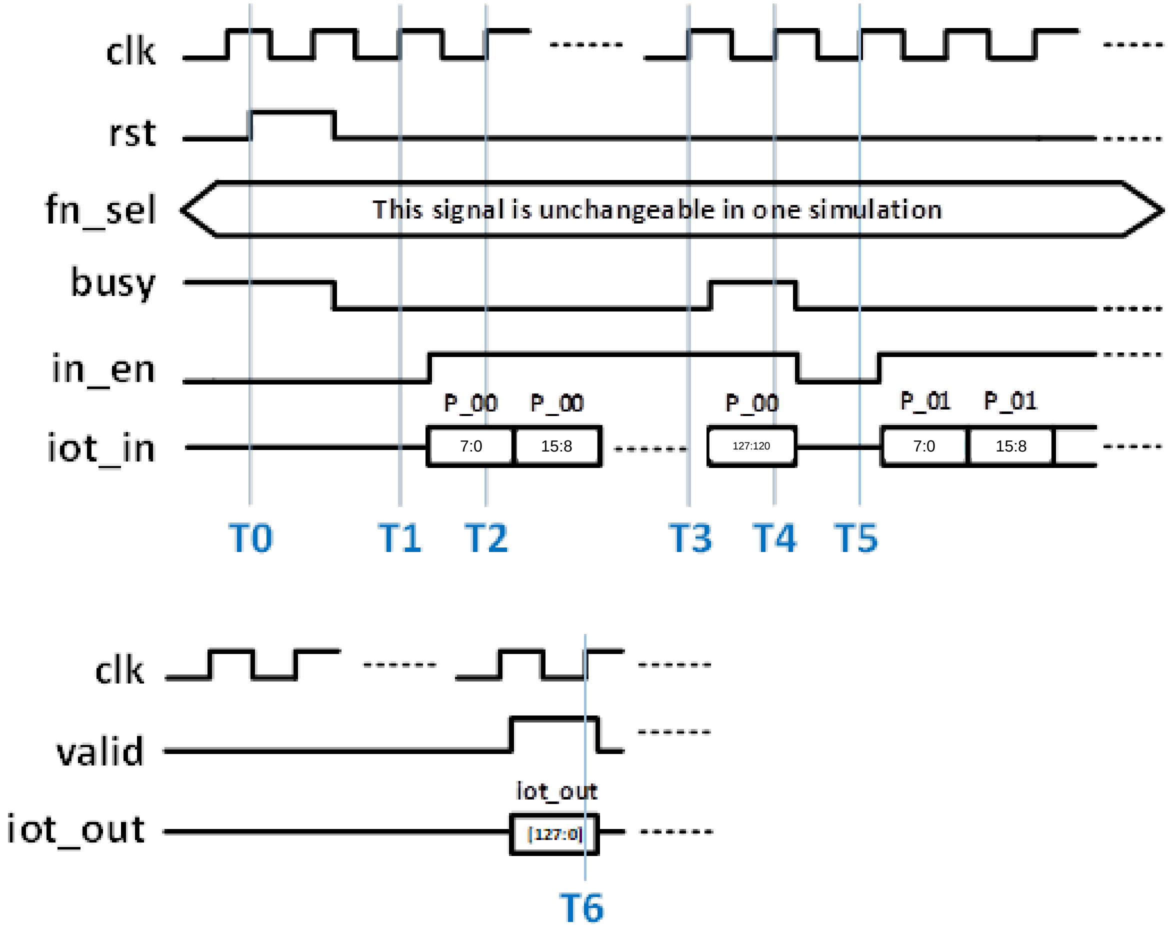
- There are 16 cycles between T1~T4 for one IoT data. You can set busy to high to stop steaming in data if you want.





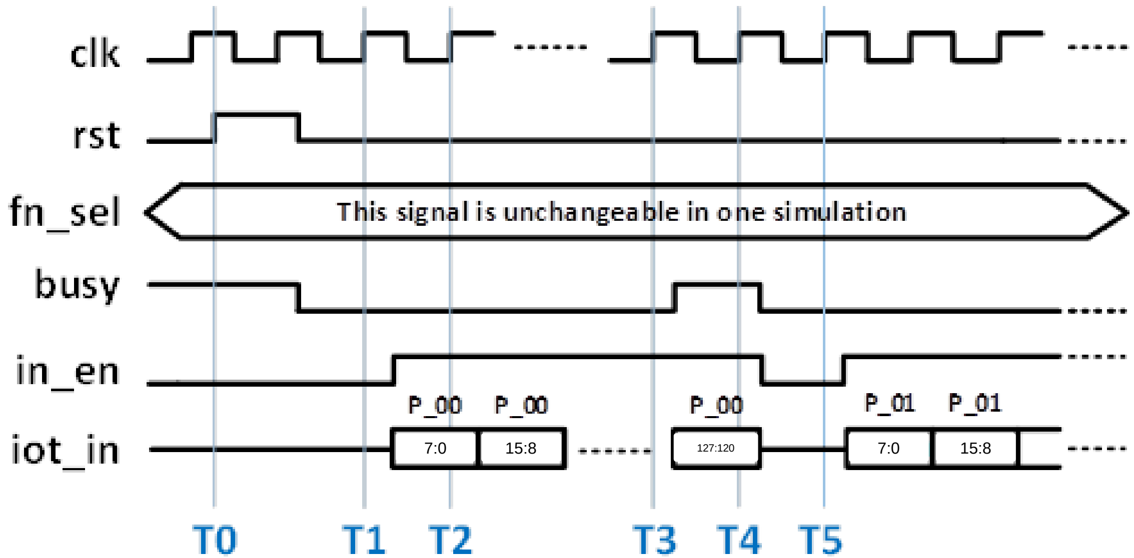
Specification (7)

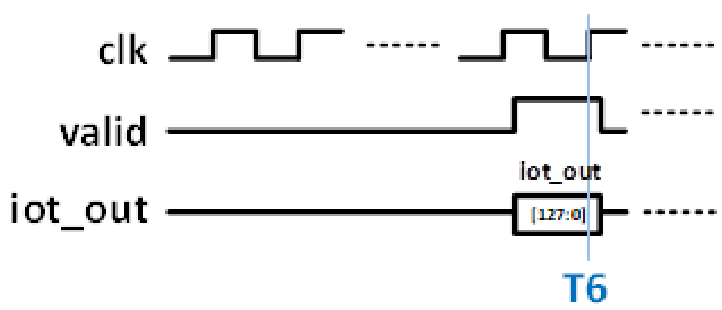
- You have to set valid to high if you want to output iot_out.



Specification (8)

- The whole processing time can't exceed 1000000 cycles.





Function

	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)

F1: Max(N)

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

Assume there are 16 IoT input data	34 F2 77 62 32 D9 15 CF 57 D2 DC 13 68 49 F0 A5
Round_0 comparison	34 F2 77 62 32 D9 15 CF
Round_0 output	F2
Round_1 comparison	57 D2 DC 13 68 49 F0 A5
Round_1 output	F0

Note:
8 bit for example

F2: Min(N)

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

Assume there are 16 IoT input data

```
34 F2 77 62 32 D9 15 CF
57 D2 DC 13 68 49 F0 A5
```

Round_0 comparison

```
34 F2 77 62 32 D9 15 CF
```

Round_0 output

```
15
```

Round_1 comparison

```
57 D2 DC 13 68 49 F0 A5
```

Round_1 output

```
13
```

Note:
8 bit for example

F3: Top2Max(N)

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

Assume there are 16 IoT input data	34 F2 77 62 32 D9 15 CF 57 D2 DC 13 68 49 F0 A5
Round_0 comparison	34 F2 77 62 32 D9 15 CF
Round_0 output	F2 D9
Round_1 comparison	57 D2 DC 13 68 49 F0 A5
Round_1 output	F0 DC

Note:
8 bit for example

F4: Last2Min(N)

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

Assume there are 16 IoT input data	34 F2 77 62 32 D9 15 CF 57 D2 DC 13 68 49 F0 A5
Round_0 comparison	34 F2 77 62 32 D9 15 CF
Round_0 output	15 32
Round_1 comparison	57 D2 DC 13 68 49 F0 A5
Round_1 output	13 49

Note:
8 bit for example

F5: Avg(N)

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

Assume there are 16 IoT input data	34 F2 77 62 32 D9 15 CF 57 D2 DC 13 68 49 F0 A5
Round_0 comparison	34 F2 77 62 32 D9 15 CF
Round_0 output	(34+F2+77+62+32+D9+15+CF) /8=7D
Round_1 comparison	57 D2 DC 13 68 49 F0 A5
Round_1 output	(57+D2+DC+13+68+49+F0+A5) /8=8B

Note:
8 bit for example

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

Assume there are 16 IoT input data	34 F2 77 62 32 D9 15 CF 57 D2 DC 13 68 49 F0 A5	low:30 High:70
Round_0 comparison	34 F2 77 62 32 D9 15 CF	
Round_0 output	34, 62, 32	
Round_1 comparison	57 D2 DC 13 68 49 F0 A5	
Round_1 output	57, 68, 49	

Note:
8 bit for example

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

Assume there are 16 IoT input data

```
34 F2 77 62 32 D9 15 CF
57 D2 DC 13 68 49 F0 A5
```

```
low:30
High:70
```

Round_0 comparison

```
34 F2 77 62 32 D9 15 CF
```

Round_0 output

```
F2, 77, D9, 15, CF
```

Round_1 comparison

```
57 D2 DC 13 68 49 F0 A5
```

Round_1 output

```
D2, DC, 13, F0, A5
```

Note:
8 bit for example

F8: PeakMax

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

Assume there are 16 IoT input data	`34 F2 77 62 32 D9 15 CF` `57 D2 DC 13 68 49 F0 A5`
Round_0 comparison	`34 F2 77 62 32 D9 15 CF`
Round_0 output	`F2`
Round_1 comparison	`57 D2 DC 13 68 49 F0 A5`
Round_1 output	`No output (because F0<F2)`

Note:
8 bit for example

F9: PeakMin

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

Assume there are 16 IoT input data	34 F2 77 62 32 D9 15 CF 57 D2 DC 13 68 49 F0 A5
Round_0 comparison	34 F2 77 62 32 D9 15 CF
Round_0 output	15
Round_1 comparison	57 D2 DC 13 68 49 F0 A5
Round_1 output	13 (because 13<previous output 15)

Note:
8 bit for example

IOTDF.v

```
`timescale 1ns/10ps
module IOTDF( clk, rst, in_en, iot_in, fn_sel, busy, valid, iot_out);
input          clk;
input          rst;
input          in_en;
input  [7:0]   iot_in;
input  [3:0]   fn_sel;
output         busy;
output         valid;
output [127:0] iot_out;

endmodule
```

rtl_01.f

- Filelist

```
// ------------------------------------------------------------------------------
// Simulation: HW4_IOT
// ------------------------------------------------------------------------------

// testbench
// ------------------------------------------------------------------------------
../00_TESTBED/testfixture.v


// design files
// ------------------------------------------------------------------------------
./IOTDF.v
```

02_SYN

- IOTDF_DC.sdc

```
# operating conditions and boundary conditions #


create_clock -name clk  -period 6.5   [get_ports  clk]      ;#Modify period by yourself
```

- Run the command to do synthesis
  - syn.tcl needs to be written by yourself (can refer to hw3)

```
dc_shell-t -f syn.tcl | tee syn.log
```

rtl_03.f

- Filelist

```
// ------------------------------------------------------------
// Simulation: HW4_IOT
// ------------------------------------------------------------

// testbench
// ------------------------------------------------------------
../00_TESTBED/testfixture.v
/home/raid7_2/course/cvsd/CBDK_IC_Contest_v2.5/Verilog/tsmc13_neg.v


// design files
// ------------------------------------------------------------
./IOTDF_syn.v
```

runall_rtl & runall_syn

- runall_rtl

```
vcs -f rtl_01.f -full64 -R +v2k -debug_access+all \
+define+p1+F1 | tee rtl_F1.log
```

- runall_syn

```
vcs -f rtl_03.f -full64 -R +v2k -debug_access+all \
+define+SDF+p1+F1 +neg_tchk | tee rtl_syn_F1.log
```

testfixture.v

- P2 is for hidden pattern

```
`timescale 1ns/10ps
`define SDFFILE     "../02_SYN/Netlist/IOTDF_syn.sdf"     //Modify your sdf file name
`define CYCLE       6.5                     //Modify your CYCLE
`define DEL         1.0
`define PAT_NUM     96
`define End_CYCLE  10000000
```

```
`elsif p2
    localparam PAT_NUM = 96;
    localparam F1_NUM = 12;
    localparam F2_NUM = 12;
    localparam F3_NUM = 24;
    localparam F4_NUM = 24;
    localparam F5_NUM = 12;
    localparam F6_NUM = 20;
    localparam F7_NUM = 73;
    localparam F8_NUM = 3;
    localparam F9_NUM = 1;
```

Submission

- Create a folder named studentID_hw4 and follow the hierarchy below

```
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
```

- Compress the folder **studentID_hw4** in a tar file named **studentID_hw4_v*k*.tar** (*k* is the number of version, *k* =1,2,…)
- Submit to NTU Cool

Report

- TAs will run your design with your clock period
- 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)
```

Grading Policy

- Simulation:

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

- Performance: (Use pattern1)
  - Performance = (Power1 × Time1 + … + Power9 × Time9)*Area
    Unit: power(mW), Time(ns), Area(um^2)
  - Baseline = $3.5*10^9$
  - Need to pass hidden pattern to get the score of this part

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

Area

- Area: Cell area from synthesis report (ex. 28254.92um^2 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

- Time: processing time from simulation (ex. 10016.50ns 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

- 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
```

Grading Policy

- TA will use **runall_rtl** and **runall_syn** to run your code at RTL and gate-level simulation.
- Do not memorize the answers directly in any way, otherwise you will not get the ranking part of the grade
- **No delay submission is allowed**
- Lose **3 point** for any wrong naming rule or format for submission
- No plagiarism

Hint

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

References

- [1] IC Design Contest, 2019.