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

TA: 江承恩 r13943008@ntu.edu.tw

TA: 陳柏任 d13943013@ntu.edu.tw

Due Tuesday, Nov. 18, 13:59

Data Preparation

1. Decompress 1141_hw4.tar with following command

```
tar -xvf 1141_hw4.tar
```

Folder	File	Description
DES_additional_material/permutations	All the tables	Excel files for the Permutation Tables
DES_additional_material/S_boxes	All the S boxes	Excel files for the S boxes
00_TESTBED	testfixture.v	Testbench for IOTDF.
00_TESTBED/pattern1_data	pattern1.dat	Input IoT data for f1~f5
00_TESTBED/pattern1_data	f1.dat ~ f5.dat	Output golden for function 1 ~ function 5
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

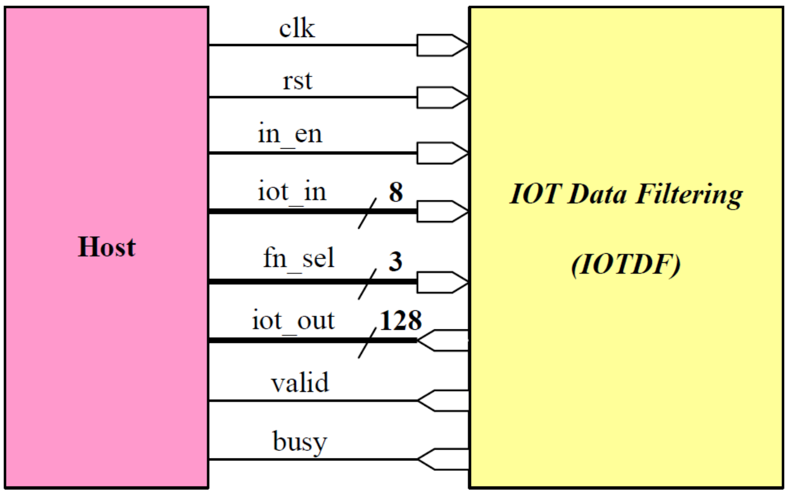
All libraries needed for synthesis, simulation can be found in previous homework.

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

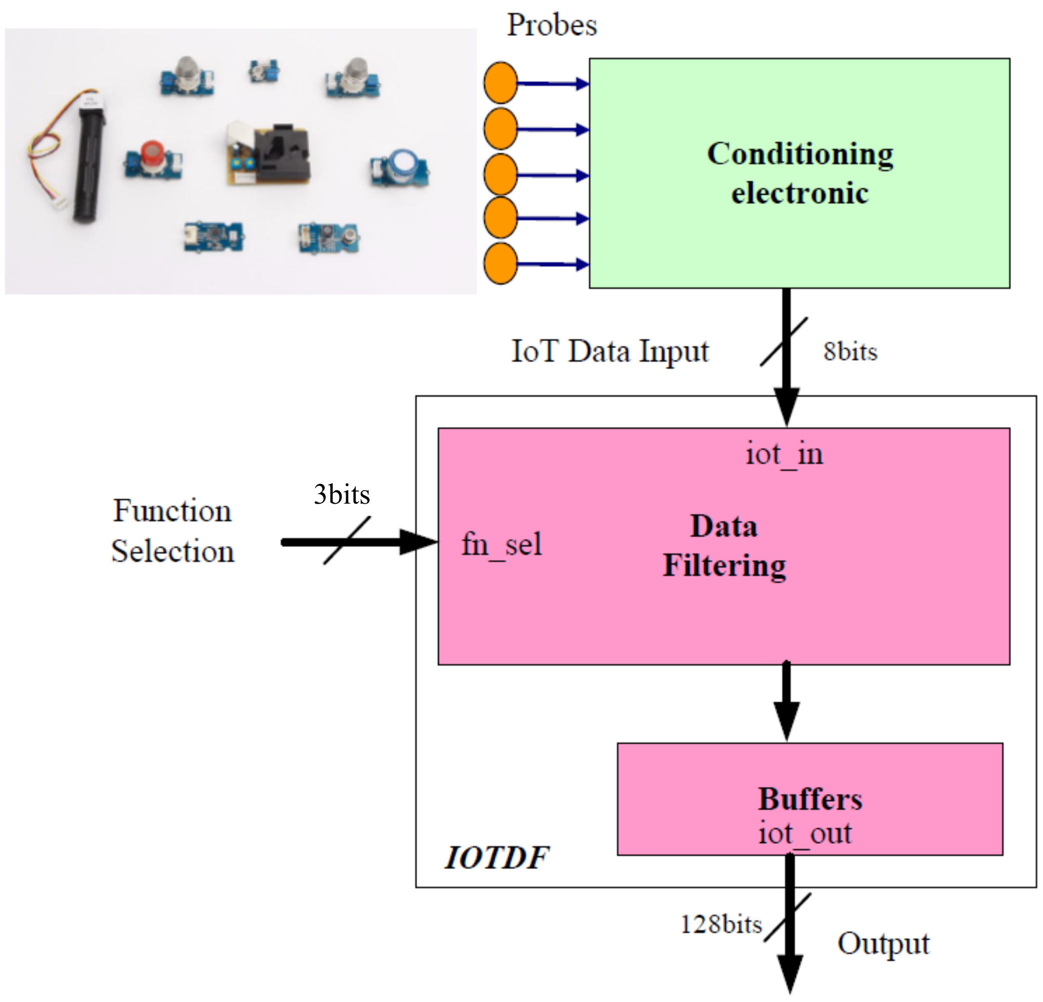


Specifications

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

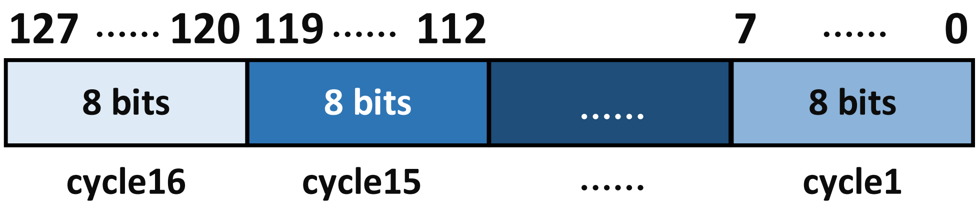
Signal Name	I/O	Width	Simple Description
clk	I	1	Clock signal in the system (positive edge trigger). All inputs are synchronized with the **positive** edge clock. All outputs should be synchronized at clock **rising** edge
rst	I	1	Active **high** asynchronous reset.
in_en	I	1	Input enable signal. When **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 **64** in this homework.
fn_sel	I	3	Function Select Signal. There are 5 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



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 64 data are required to fetch for each function simulation.

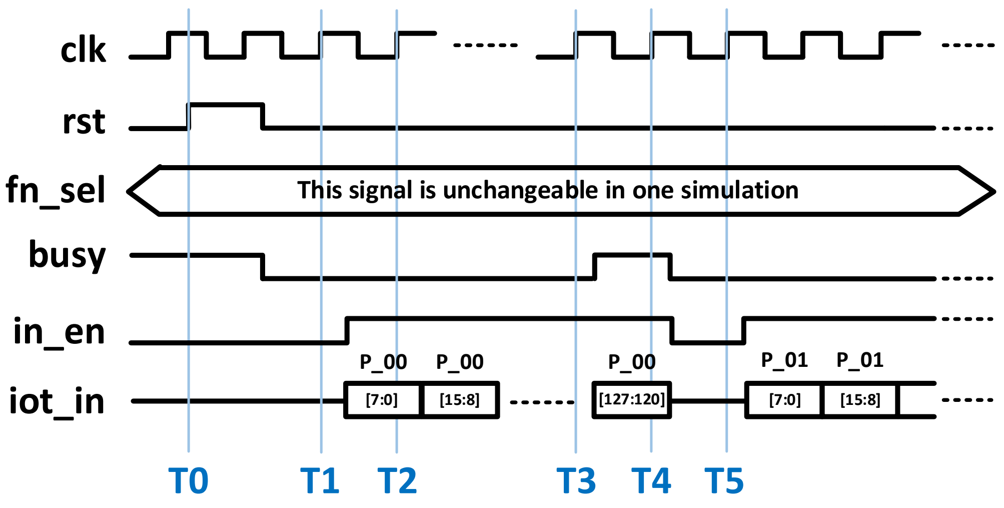


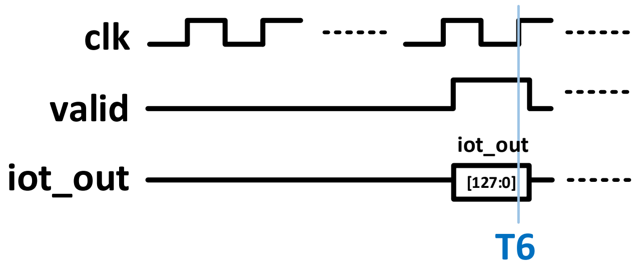
2. Nine functions are asked to design in this homework.

	Fn_sel	Functions	Description
F1	3’b001	Encrypt(N)	Use the DES algorithm to encrypt 64-bit data
F2	3’b010	Decrypt(N)	Use the DES algorithm to decrypt 64-bit data
F3	3’b011	CRC_gen(N)	Generate a CRC checksum, with the generator polynomial = $x^3 + x^2 + 1$
F4	3’b100	Sorting(N)	Sort 16 8-bit partial data in descending order

The details of each function can be found in 1141_hw4_note.pdf

Timing Diagram





1. IOTDF is initialized between T0~T1.
2. in_en is set to high and start to input IoT data P_00[127:120] if busy is low at T1.
3. in_en is kept to high and input IoT data P_00[119:112] if busy is low at T2.
4. in_en is kept to high and input IoT data P_00[7:0] 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.

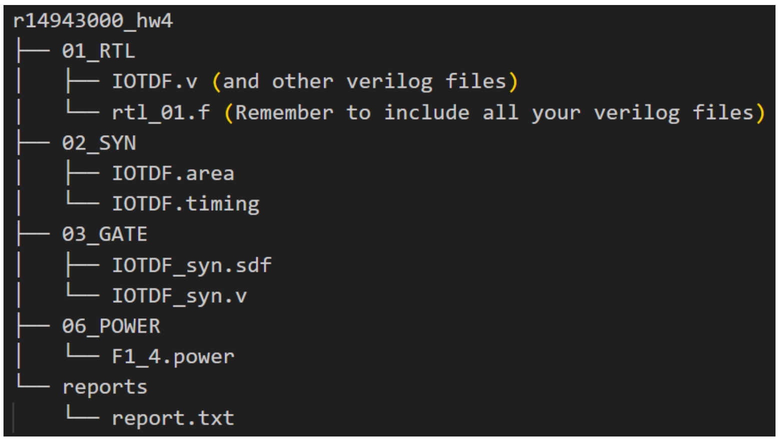
7. You have to set valid to high if you want to output iot_out.
8. The whole processing time can't exceed 1000000 cycles.

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.
4. Reasonably allocate the number of tables that need to be read in each cycle.

Submission

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



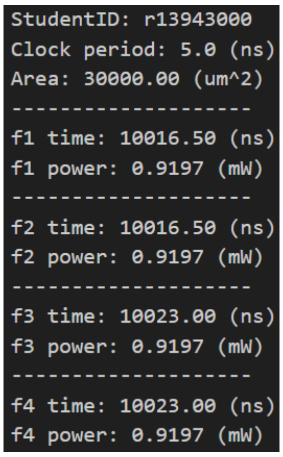
```
r14943000_hw4
├── 01_RTL
│   ├── IOTDF.v (and other verilog files)
│   └── rtl_01.f (Remember to include all your verilog files)
├── 02_SYN
│   ├── IOTDF.area
│   └── IOTDF.timing
├── 03_GATE
│   ├── IOTDF_syn.sdf
│   └── IOTDF_syn.v
├── 06_POWER
│   └── F1_4.power
└── reports
    └── report.txt
```

Note: Use **lower case** for the letter in your student ID. (Ex. r13943000_hw4)

2. Content of the report.txt

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

   (I will read the clock period from the report to run the simulation, and the rest of the data will be referenced when the results I produce differ from those obtained by the students themselves. I will use this data to infer why they are different. I will not directly use this data to calculate grades.)



```
StudentID: r13943000
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)
```

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.
   (Ex. r13943008_hw4_v1.tar)

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%)

Function	RTL Simulation	Gate-level Simulation
F1	10%	10%
F2	5%	5%
F3	5%	5%
F4	10%	10%
Hidden	X	10%

3. Performance (30%)
   Score for performance to be considered: (Use pattern1)

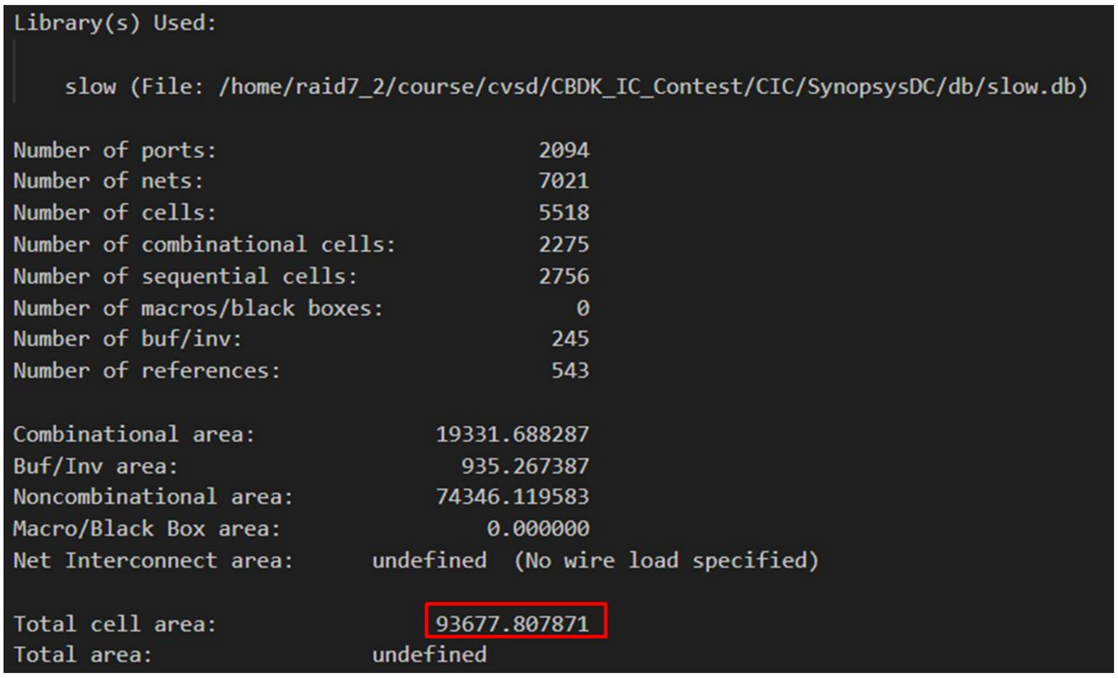
   Score = (Power1 × Time1 + … + Power4 × Time4)*Area

   Unit: Power(mW), Time(ns), Area(um^2)

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

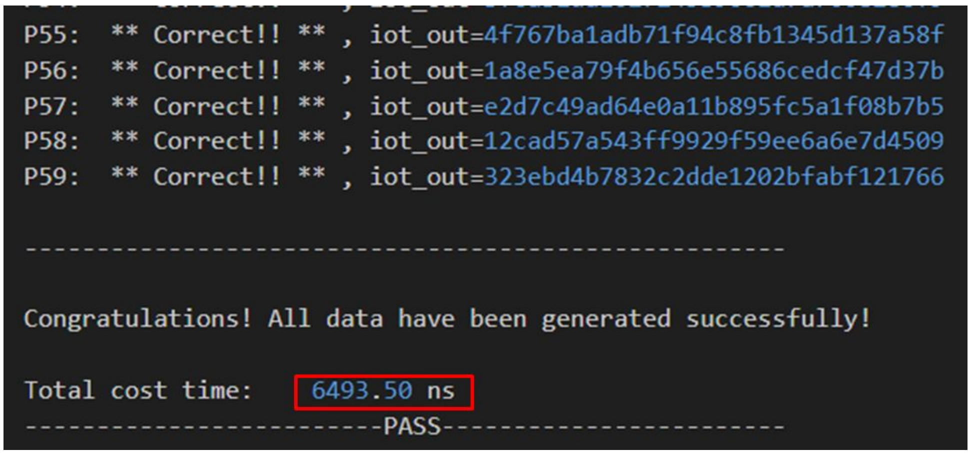
	Score
Ranking	30%

4. Precise explanation of the terms **Power**, **Time**, and **Area**
   - Area: Cell area from synthesis report (ex. 93677.81 um^2 below)



```
Library(s) Used:

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

Number of ports:                         2094
Number of nets:                          7021
Number of cells:                         5518
Number of combinational cells:           2275
Number of sequential cells:              2756
Number of macros/black boxes:               0
Number of buf/inv:                        245
Number of references:                     543

Combinational area:              19331.688287
Buf/Inv area:                      935.267387
Noncombinational area:           74346.119583
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 93677.807871
Total area:                 undefined
```

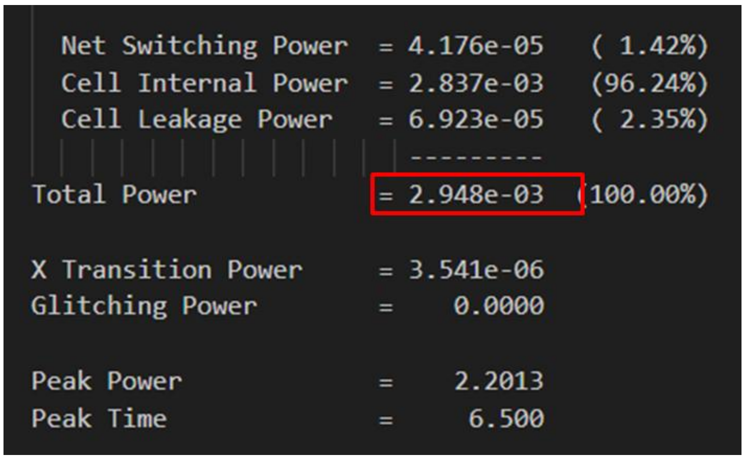
- Time: processing time from simulation (ex. 6493.50ns below)



```
P55:  ** Correct!! ** , iot_out=4f767ba1adb71f94c8fb1345d137a58f
P56:  ** Correct!! ** , iot_out=1a8e5ea79f4b656e55686cedcf47d37b
P57:  ** Correct!! ** , iot_out=e2d7c49ad64e0a11b895fc5a1f08b7b5
P58:  ** Correct!! ** , iot_out=12cad57a543ff9929f59ee6a6e7d4509
P59:  ** Correct!! ** , iot_out=323ebd4b7832c2dde1202bfabf121766


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

Congratulations! All data have been generated successfully!

Total cost time:   6493.50 ns
-----------------------PASS----------------------
```

- Power: Use below command to analyze the power. (Need to source the following .cshrc file first!) (ex. 2.948 mW below)

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



```
    Net Switching Power  = 4.176e-05   ( 1.42%)
    Cell Internal Power  = 2.837e-03   (96.24%)
    Cell Leakage Power   = 6.923e-05   ( 2.35%)
                           ---------
  Total Power            = 2.948e-03   (100.00%)

  X Transition Power     = 3.541e-06
  Glitching Power        =    0.0000

  Peak Power             =    2.2013
  Peak Time              =     6.500
```

Reference

[1] Reference for DES algorithm DES Algorithm - HackMD

[2] Reference for CRC calculation On-line CRC calculation and free library - Lammert Bies

[3] Reference for sorting algorithm implementation Implementation of sorting algorithms in reconfigurable hardware