Computer-Aided VLSI System Design

Homework 4: IoT Data Filtering

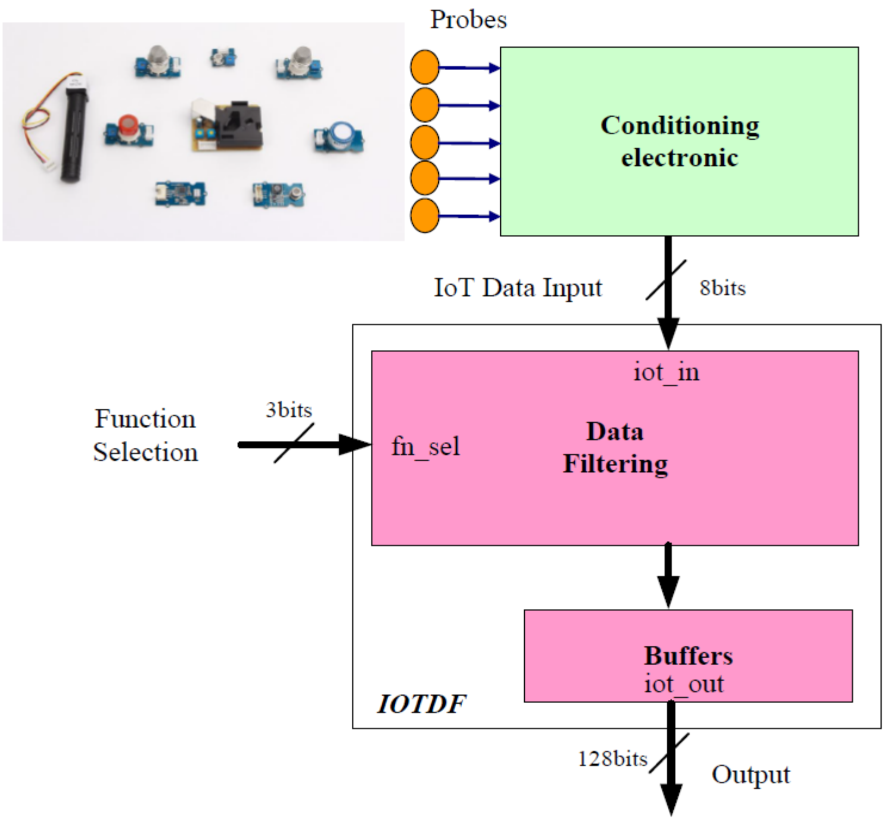
Graduate Institute of Electronics Engineering, National Taiwan University

Goals

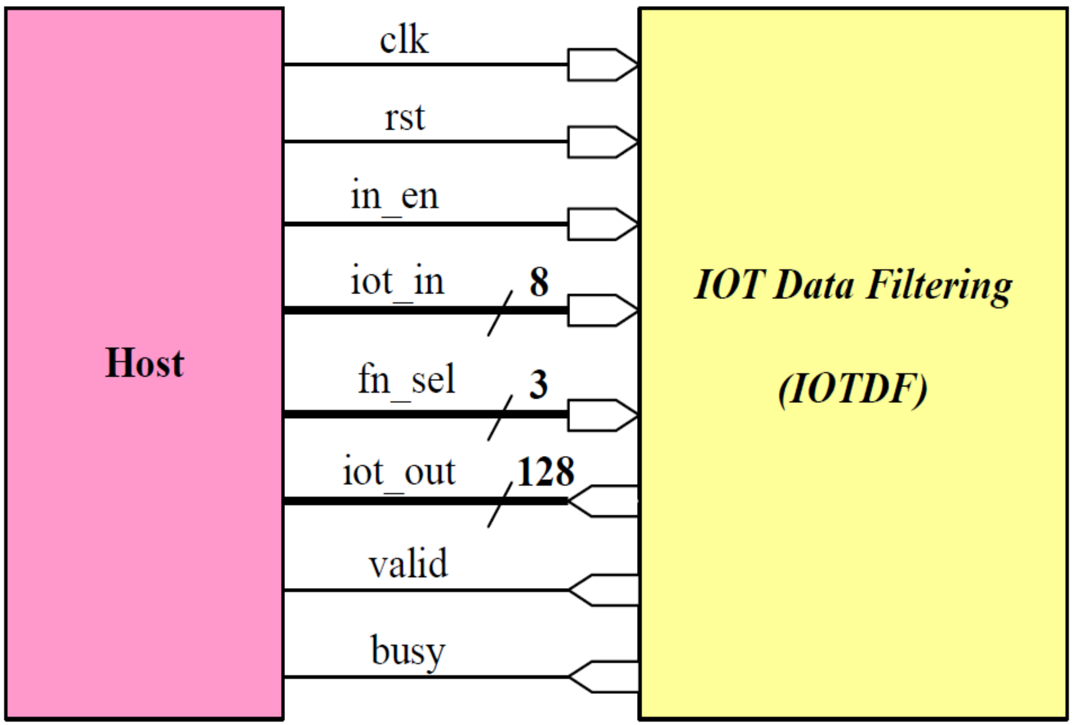
- In this homework, you will learn
  - Generate patterns for testing
  - Optimizing the trade-off between power consumption, operating frequency, and area
  - Use primetime to estimate power
  - Learn to design an architecture for processing data with long bit lengths
  - Learn to efficiently access the look-up table and accelerate its throughput

Introduction

- 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 [1]

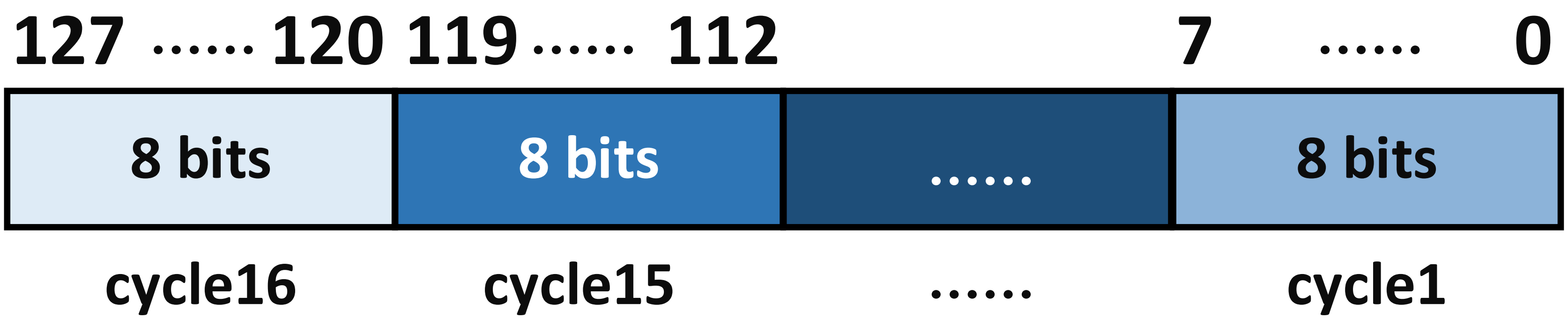


Block Diagram



Design Description

- The sensor data is a 128-bit unsigned data, which is divided in 16 8-bit partial data for IOTDF fetching.
- Only 64 data are required to fetch for each function simulation.

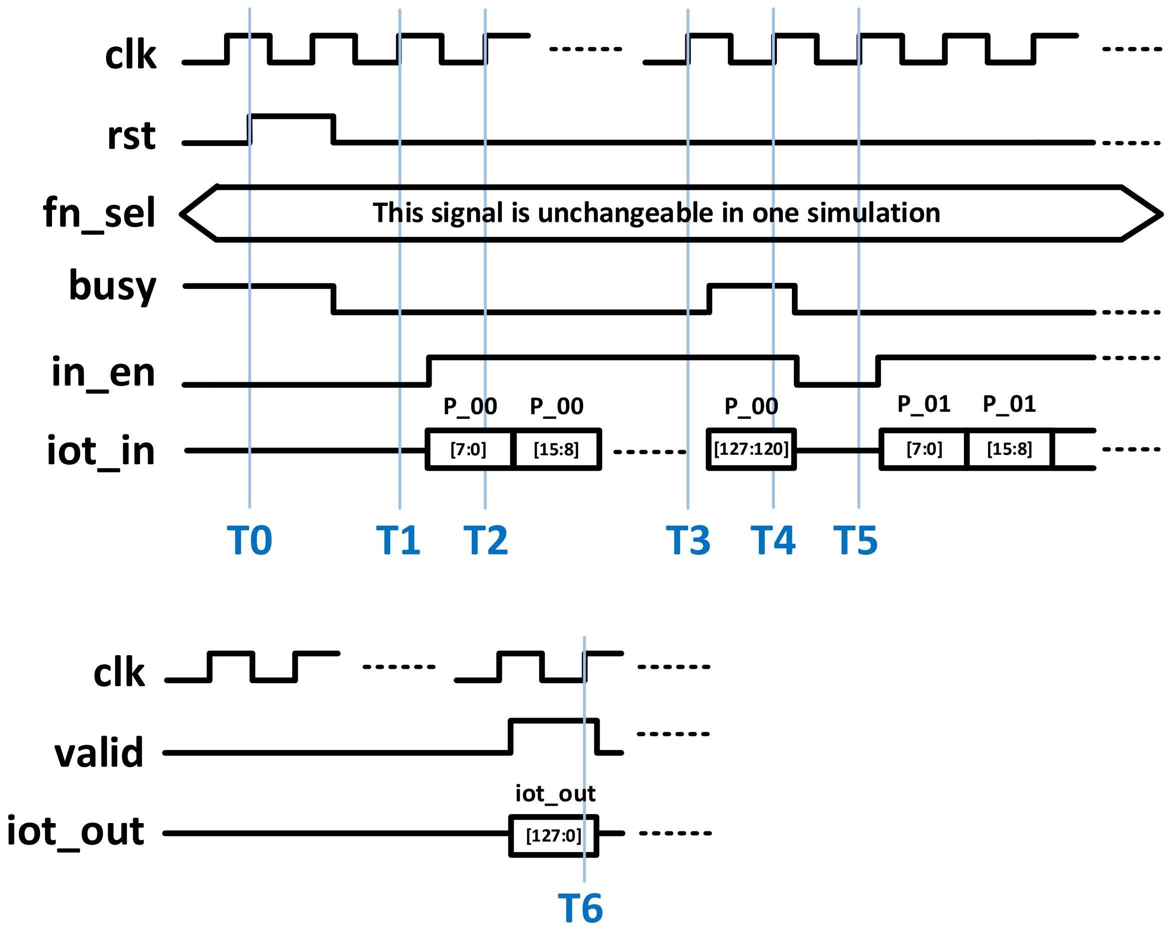


Input/Output

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 64.
fn_sel	I	3	Function Select Signal. There are 4 functions supported in IOTDF. For each simulation, only 1 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.

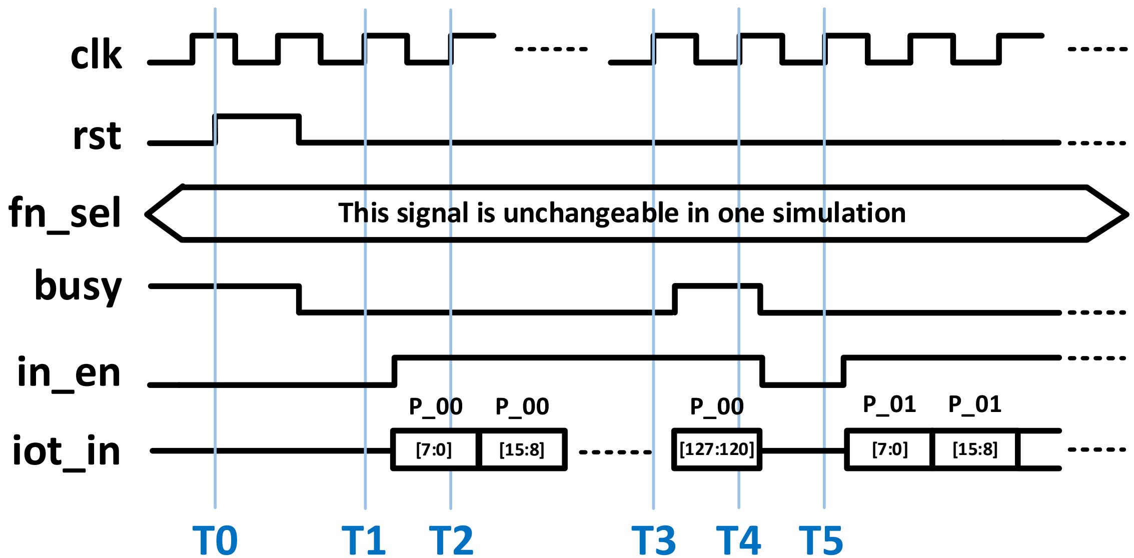
Specification (1)

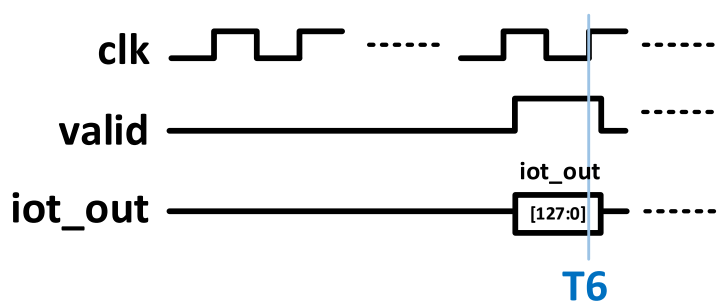
- IOTDF is initialized between T0~T1..



Specification (2)

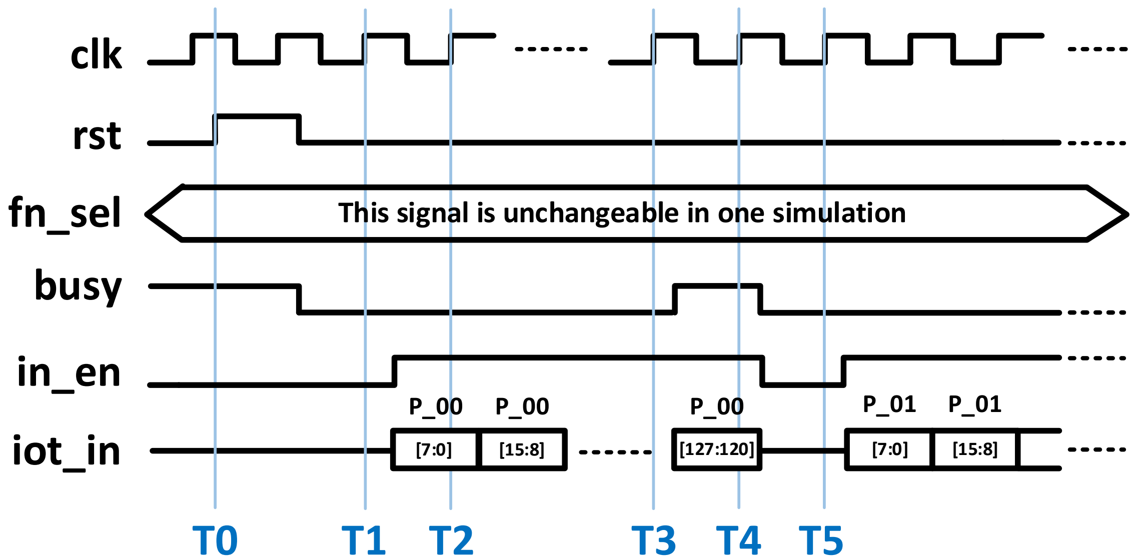
- in_en is set to high and start to input IoT data P_00[7:0] if busy is low at T1.

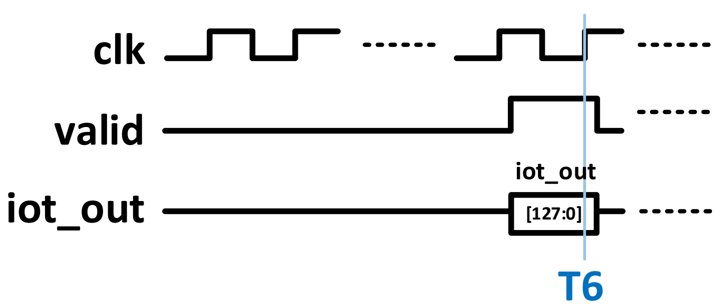




Specification (3)

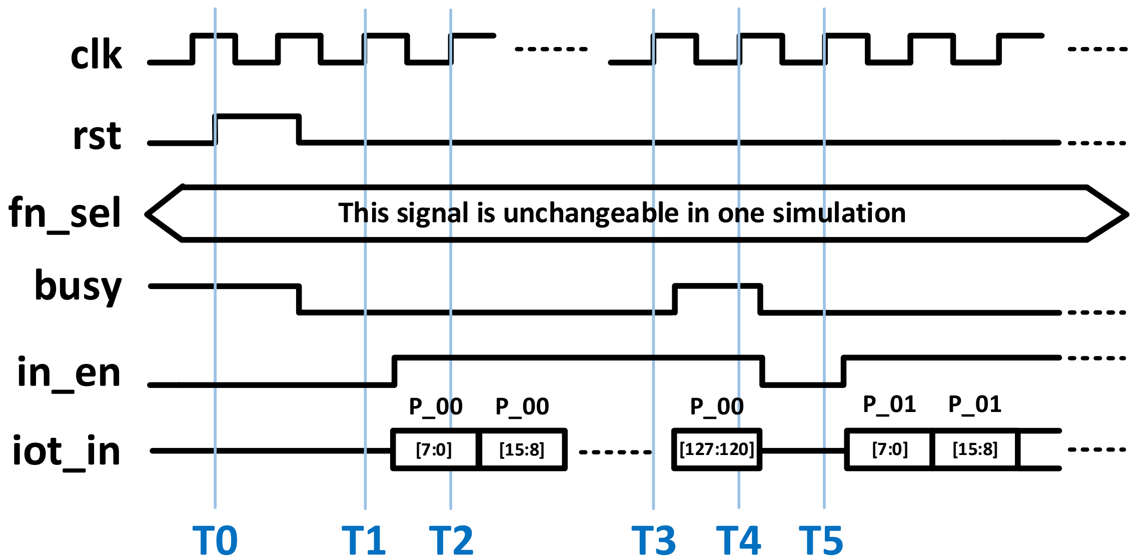
- in_en is kept to high and input IoT data P_00[15:8] if busy is low at T2.

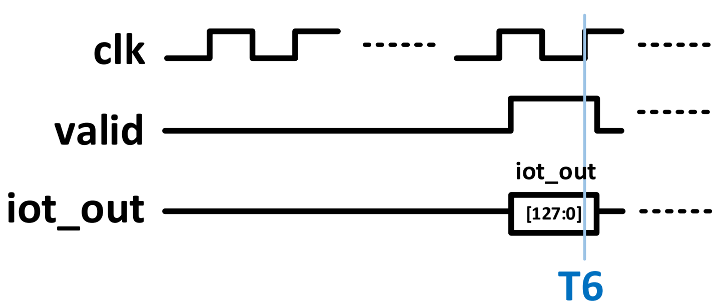




Specification (4)

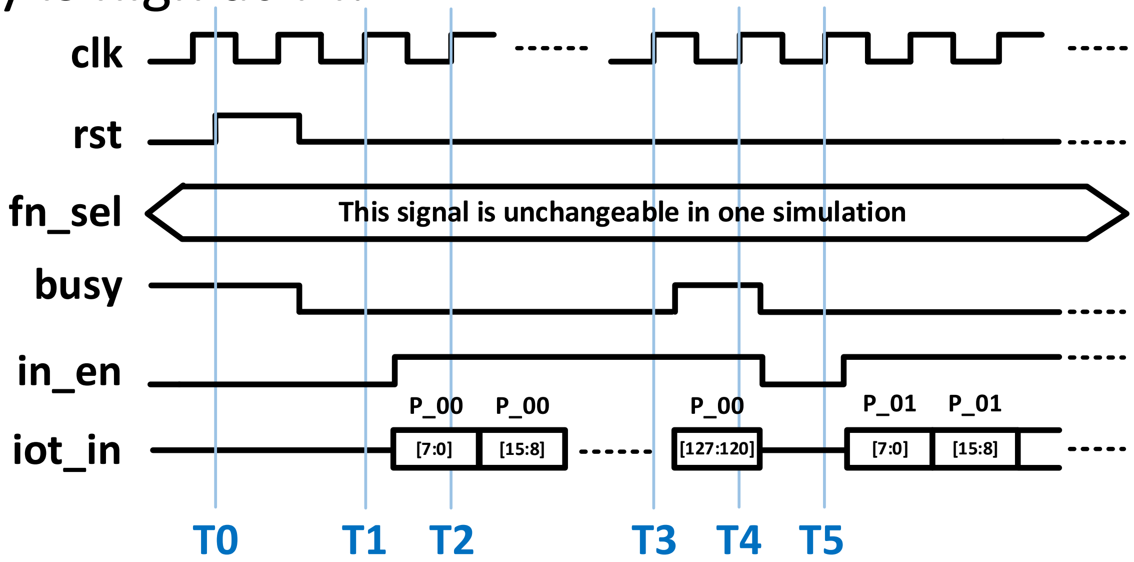
- in_en is kept to high and input IoT data P_00[127:120] if busy is low at T3.

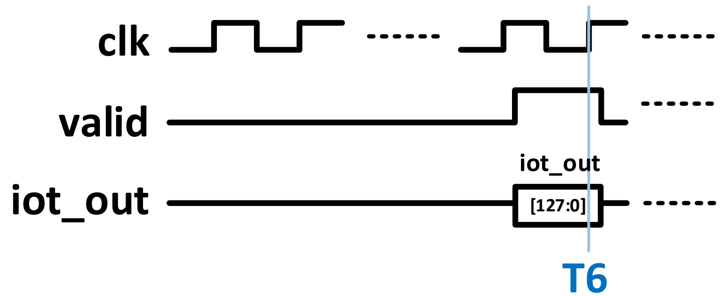




Specification (5)

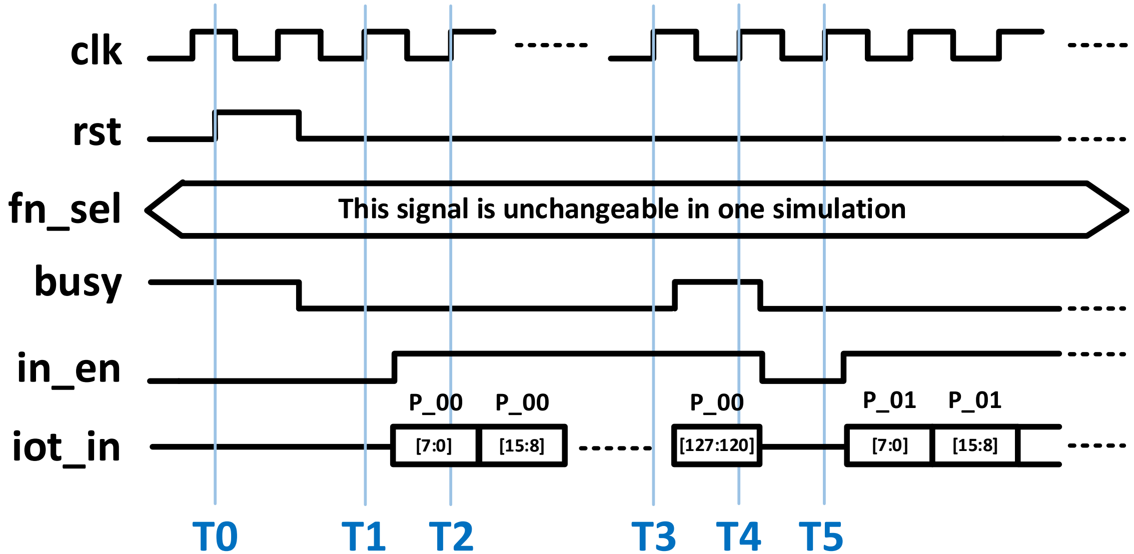
- in_en is set to low and IoT data is set to 0 (stop streaming in data) if busy is high at T4.

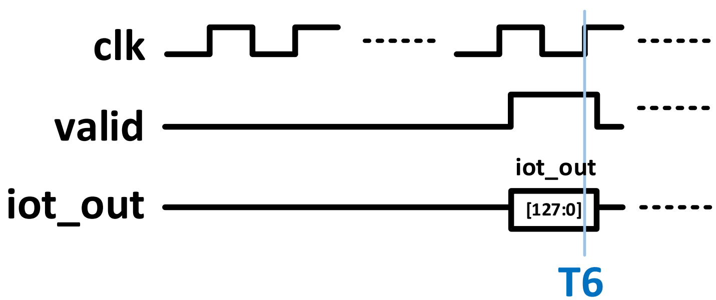




Specification (6)

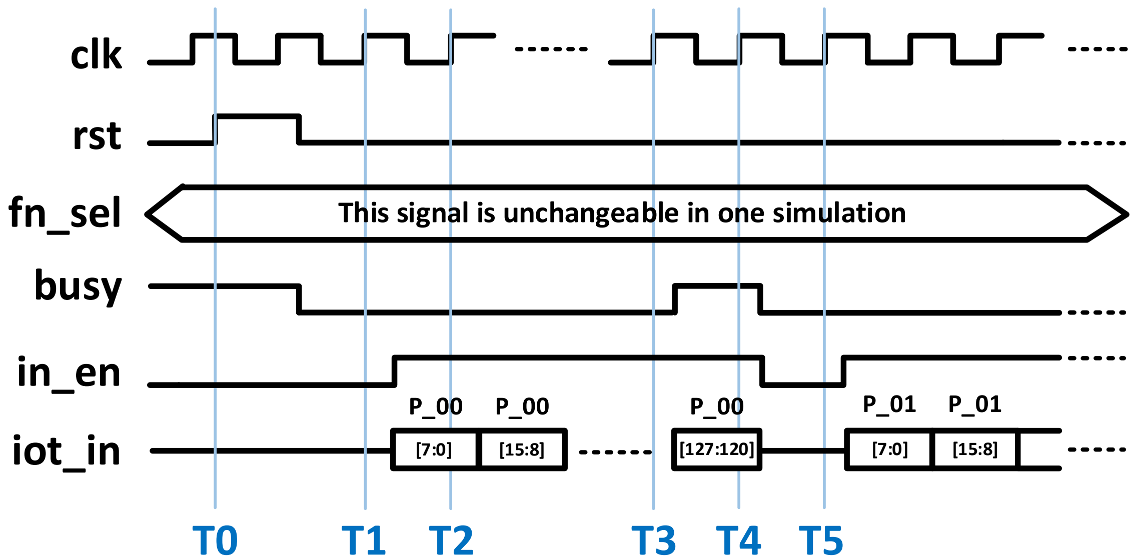
- There are 16 cycles between T1~T4 for one IoT data. You can set busy to high to stop steaming in data if you want.

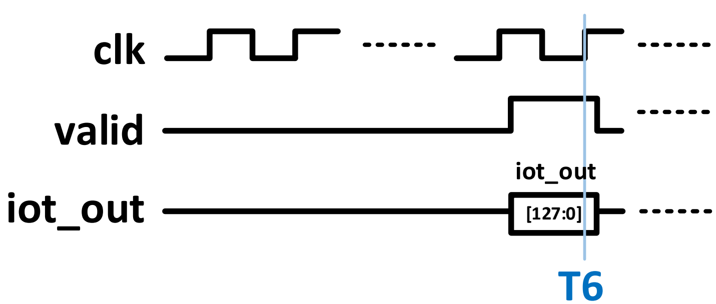




Specification (7)

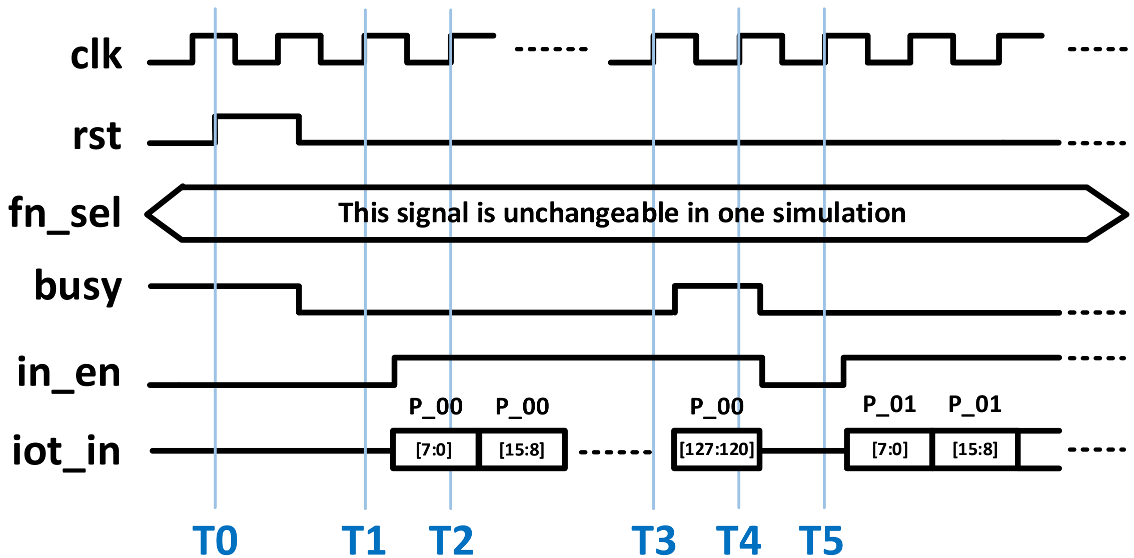
- You have to set valid to high if you want to output iot_out.

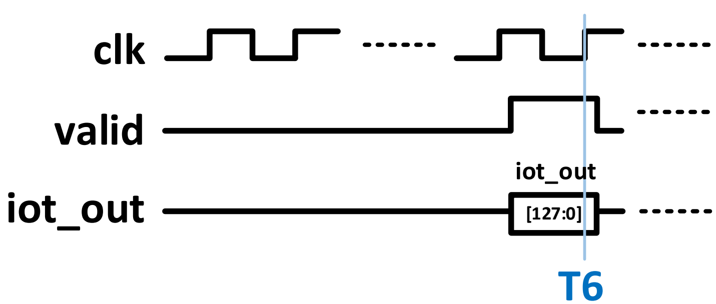




Specification (8)

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





Functions

	Fn_sel	Functions
F1	3’b001	Encrypt(N)
F2	3’b010	Decrypt(N)
F3	3’b011	CRC_gen(N)
F4	3’b100	Sorting(N)

F1: Encrypt(N)

- Use the DES algorithm to encrypt 64-bit data [2]

128-bit input data

[127:64]	[63:0]
64-bit main key	64-bit plaintext

128-bit output data (iot_out)

[127:64]	[63:0]
64-bit main key	64-bit cipher text

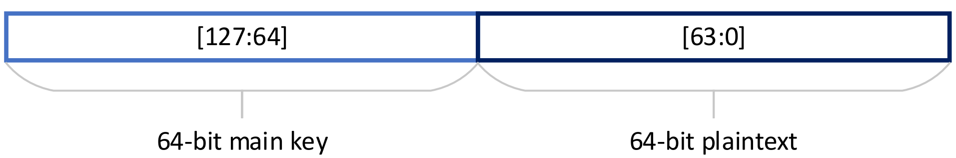
Data Encryption Standard (DES)

- Development
  - IBM's creation, 1970s
  - Adopted by NIST in 1977
- Application
  - Prevailing encryption for years
  - Basis for modern ciphers
- Security
  - Susceptible to brute-force
  - Superseded by Advanced Encryption Standard (AES)

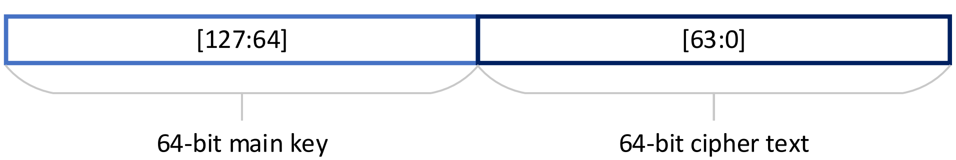
DES Workflow

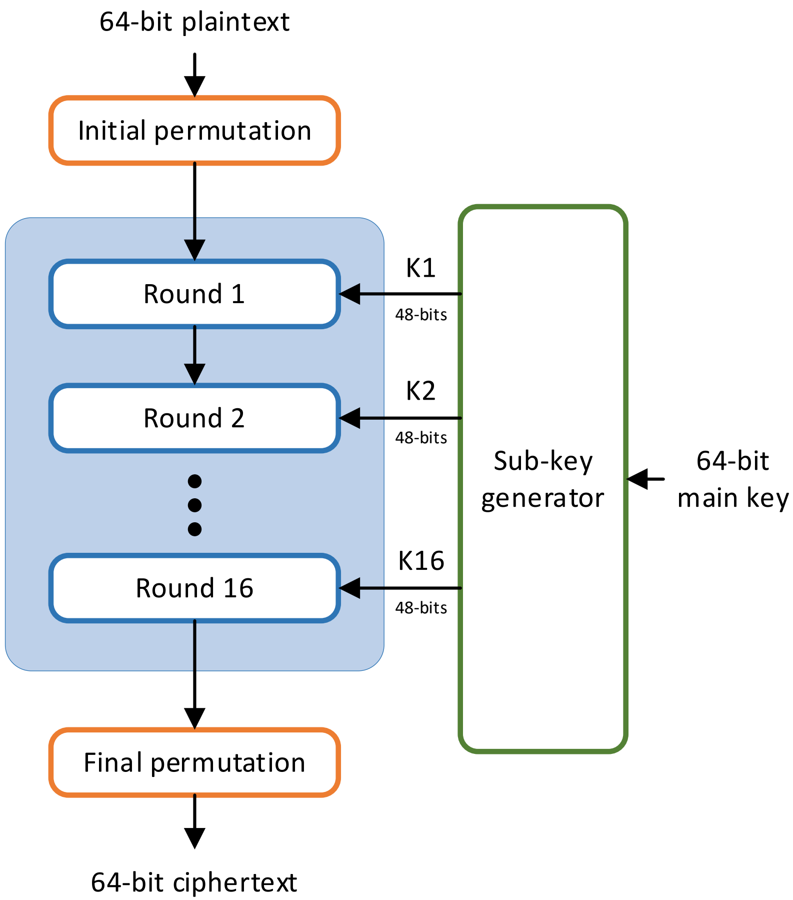
- Require 16 rounds of encryption
- Each round needs a different sub-key
- The orange box represents a LUT
- Final permutation is the inverse of the initial permutation

128-bit input data



128-bit output data (iot_out)





Permutation Table

- Excel file for the Permutation Table is located in the "permutations" folder
- Name of the Excel file matches the table name
  - Ex: Initial permutation corresponds to Initial_permutation.xlsx

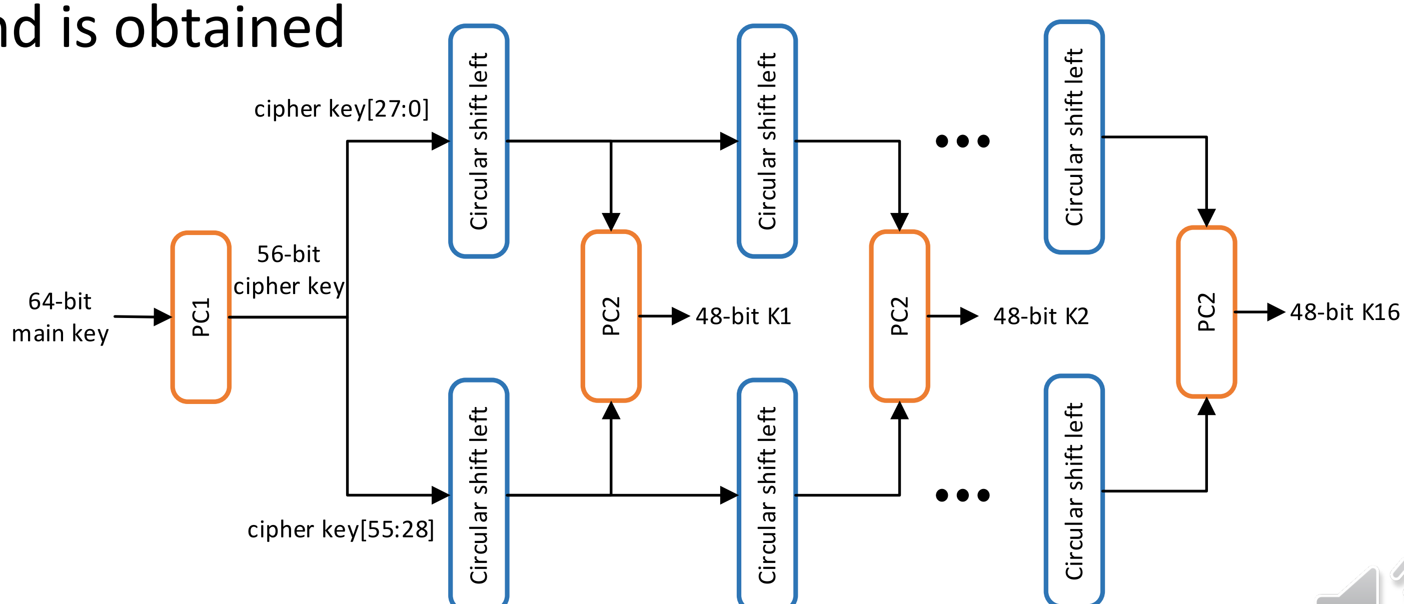
	A	B
1	Output index	Input index
2	55	7
3	54	15
4	53	23
5	52	31
6	51	39
7	50	47
8	49	55
9	48	63
10	47	6
11	46	14
12	45	22
13	44	30

Details Of Key Generator

- Main key is processed through the PC1 LUT to form the cipher key, then splited into left and right halves for circular shift left
- Each round has different shift amount, following the shifting rule
- After passing through the PC2 LUT, the sub-key required for each round is obtained

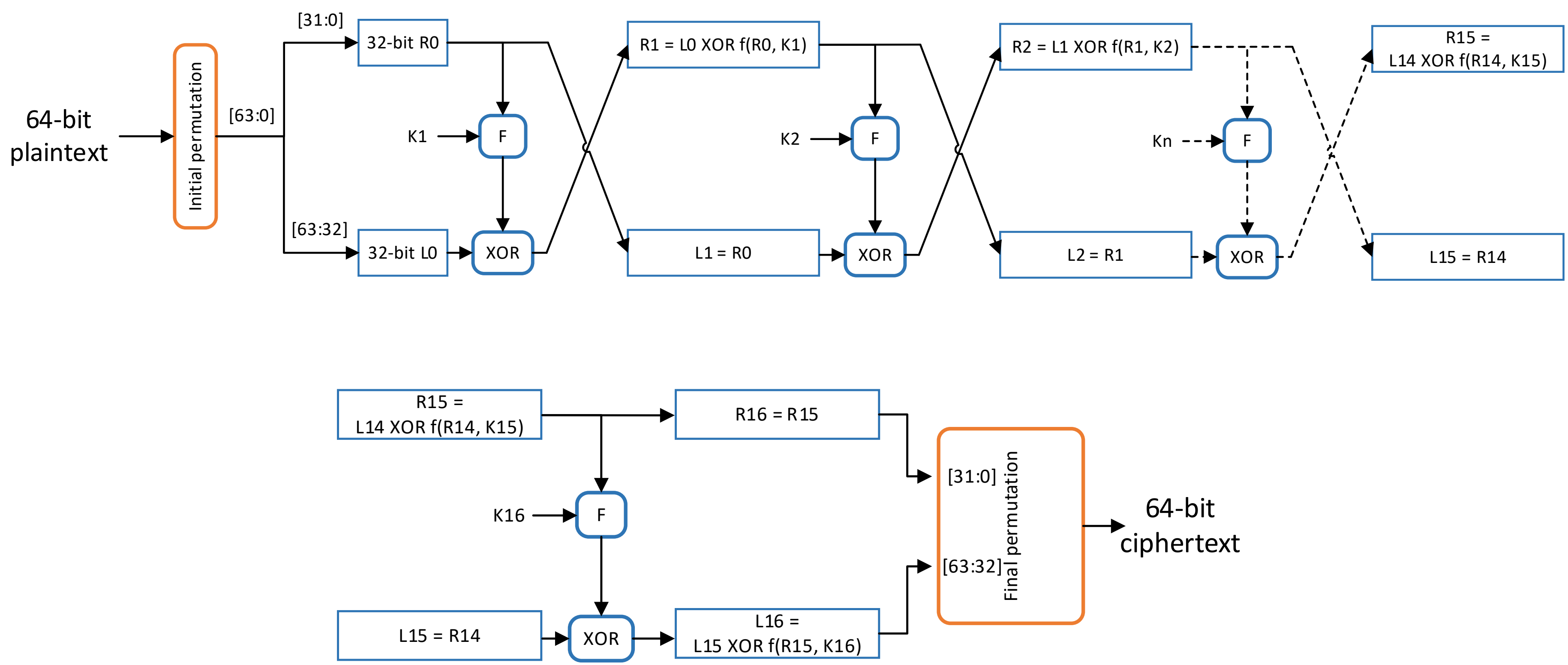
Shifting Rule

Rounds	Shift
1, 2, 9, 16	One bit
Others	Two bit



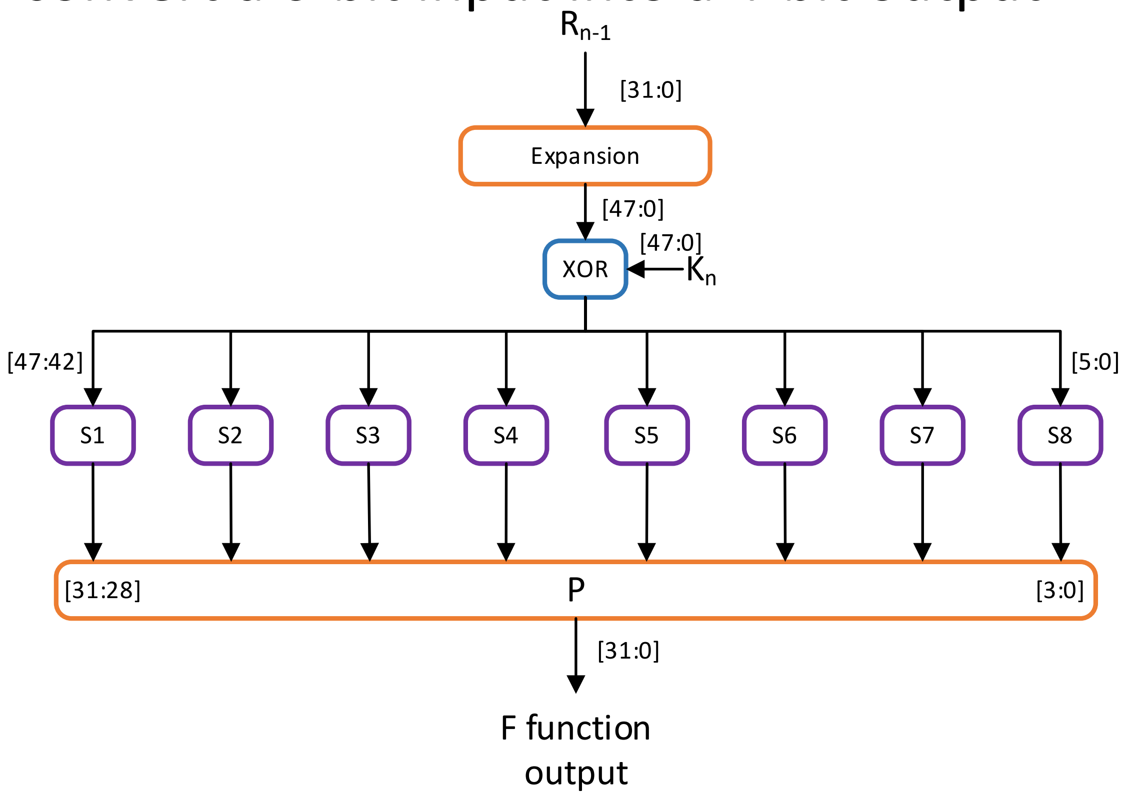
Details Of Each Round

- Details of F function is on the next slide



F Function

- The Expansion LUT transforms a 32-bit input into a 48-bit output
- S-boxes convert a 6-bit input into a 4-bit output



S-box

- Excel files for S1 to S8 are located in the 'S_boxes' folder
- The method of S-box reading is as follows

6-bit input data

1 1 0 0 1 0

Row number

1 y y y y 0

Column number

x 1 0 0 1 x

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	S1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
2	0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
3	0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1yyyy0	4	1	14	8	13	6	2	11	15	$\boxed{12}$	9	7	3	10	5	0
5	1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

4-bit output data

1 1 0 0

F2: Decrypt(N)

- Use the DES algorithm to decrypt 64-bit data
- Operation process is similar to Encrypt, with the only difference being the usage order of the sub-keys, changing from 1~16 to 16~1

128-bit input data

[127:64]	[63:0]
64-bit main key	64-bit cipher text

128-bit output data (iot_out)

[127:64]	[63:0]
64-bit main key	64-bit plaintext

F3: CRC_gen(N)

- Generate a CRC checksum [2]
- Generator polynomial = $x^3 + x^2 + 1$
  - This assignment focuses on this Generator polynomial
- Place 3-bit calculation result in iot_out[2:0], and fill the rest with zeros

Assume input data: 1110

CRC Outcome: iot_out[2:0] = 010

```
         1010
1101 ) 1110000
       1101
       ----
        1100
        1101
        ----
         010
```

Note: 4 bit for example

F4: Sorting(N)

- Sort 16 8-bit partial data in descending order [3]

34	F2	...	A5

⇩

16 IoT input data

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

⇩

Sorted data

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

⇩

F2	F0	...	13

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   [2:0]  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 -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 -sverilog –v2005 –debug_access+all
+notimingcheck +define+p1+F1 | tee rtl_F1.log
```

- runall_syn

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

testfixture.v

- P2 is for hidden pattern

```
`timescale 1ns/10ps
`define SDFFILE     "./IOTDF_syn.sdf"     //Modify your sdf file name
`define CYCLE       6.5                   //Modify your CYCLE
`define DEL         1.0
`define PAT_NUM     60
`define End_CYCLE  1000000
```

```
`elsif p2 // modify the following number according to your pattern
   localparam PAT_NUM = 64;
   localparam F1_NUM = 64;
   localparam F2_NUM = 64;
   localparam F3_NUM = 64;
   localparam F4_NUM = 64;
```

Submission

- Create a folder named studentID_hw4 and follow the hierarchy below

```
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
```

- 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 the reported clock periods
- report.txt (record the power and processing time of gate-level simulation)

```
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)
```

Grading Policy

- Simulation:

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

- Performance: (Use pattern1)
  - Performance = (Power1 × Time1 + … + Power4 × Time4) × Area
    Unit: Power(mW), Time(ns), Area(um^2)
  - Need to pass hidden pattern to get the score of this part

	Score
Ranking (Need to pass hidden)	30%

Area

- Area: Cell area from synthesis report (ex. 93677.81um^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

- 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

- 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
```

Grading Policy

- Deadline: 2025/11/18 13:59:59 (UTC+8)
- 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
- **No delay submission is allowed**
- Lose **5 point** for any wrong naming rule or format
  - Pack all files into a single folder and compress the folder
  - Ensure that the files submitted can be decompressed and executed without issues
- **No plagiarism**

Discussion

- **NTU Cool Discussion Forum**
  - For any questions not related to assignment answers or privacy concerns, please use the NTU Cool discussion forum.
  - **TAs will prioritize answering questions on the NTU Cool discussion forum.**

- **Email: r13943008@ntu.edu.tw**
  - **Title should start with [CVSD 2025 Fall HW4]**
  - **Email with wrong title will be moved to trash automatically**

Hints

- Clock gating
- Register sharing
- Pipelining
- Reasonably use LUT

References

- [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