ECM5605(5089) F'19 : Programming Project 01

!!! DUE: 2020/02/16 THURSDAY PM11:55 !!!

[Instruction]

Please compress your program files into one and upload it to e3 course website; Your zipped file should be named <StudentID>-P1-v<X>.zip. For Example, 0480777-P1-v1.zip where 0480777 is your student id, P1 denotes programming project 1 and v1 refers to the version of your submission for this assignment. You are allowed to submit multiple versions and only the latest version will be graded. Last, a Makefile is also required for compiling your program in the zip file. Note that this programming project is to test your C/C++ programming skills (including data structure) for this course and requires very limited professional background. If you cannot understand the problem description, please make appointment with me for explanation. Last but not lest, students who cannot complete this project in time will be forced to drop this course. Thanks for your understanding.

[Problem Description & Requirement]

Recently, the power issue on VLSI testing is important. As one knows, an automatic generated pattern (a.k.a. ATPG pattern) aims to exercise as much part of the design as possible. The power consumption during the test mode will be much higher than the operation mode. One of the most concerned issues is that the large peak power may cause the functional fail during testing, i.e., a good chip may not pass the ATPG pattern due to the excessive peak power consumption. To reduce the peak power consumption during testing, one of the most effective techniques is to re-order flip-flops (a.k.a. scan cells) in the scan chain to reduce the peak power consumption during the shift stage. For example, if the pattern to be shifted into an 8-bit scan chain is 01010101, all FFs in this scan chain will transit simultaneously at the last cycle of the shift procedure. However, if the scan-chain is reordered to make the pattern become 11110000, it is clear that only one FF will transit at each cycle during the whole shift procedure, as shown in Figure 1. In this figure, the arrow above a FF indicates the direction of the transition in the FF. That is, ↑ and ↓ denote the rising and falling transitions, respectively.

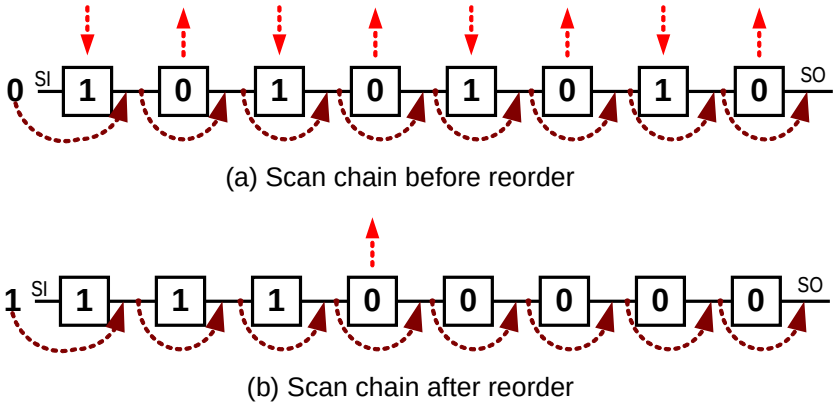

Figure 1: The last cycle of the scan chain shift procedure

The objective of this problem concerns about the re-order of scan cells in the scan chain so that both the peak power and the length of the scan chain are minimized. To judge whether the result of the re-ordered scan chain is good or not, three indexes will be taken into consideration, including maximum peak power consumption, scan-chain length and execution time. In addition, different settings of weights on these three indexes are specified in different configuration files. Furthermore, in order to simplify the problem, two assumptions are taken: (1) all scan cells have been reset to 0 before applying the first scan pattern. (2) A new pattern can be shifted into the scan chain only if the previous one has been shifted out. Under these two assumptions, only the bit transitions inside the scan patterns have to be considered in the peak power calculation.

Maximum Peak Power Calculation

To simply the problem, it is assumed that a FF will only consume power during logic transitions, and the peak power consumption is a given fixed value that is independent of the states of other FFs. Each FF will be given a specific value, which is determined by the capacitive load of the fan-out circuitry driven by such FF. It is also assumed that the peak power of all FFs will occurs at the same time in the same cycle, i.e., no timing issue needs to be taken into consideration. The peak power consumption of the scan chain in a specific cycle could be easily calculated by the summation of the peak power consumption of all the FFs that transited in this cycle. For example, if 4 FFs transited in one cycle and the peak power values for these 4 FFs were given as [3.5, 5.7, 3.8, 1.3], the peak power of this cycle can be calculated as 3.5+5.7+3.8+1.3=14.3.

The maximum peak power for a specific scan chain under a given pattern is calculated by the selection the maximum peak power during the whole shift cycles. For example, an 8-bit scan-chain will need 8 cycles to shift in a scan pattern and the peak power for these 8 cycles are [14.3, 9.4, 24.6, 11.9, 8.3, 25.6, 7.8, 9.5]. The maximum peak power for this scan chain pattern is 25.6. The maximum peak power for the whole scan procedure is calculated by the selection of the maximum peak power among all the scan chain patterns.

Scan-Chain Length Calculation

To represent the physical location of each FF in the layout, each FF will be assign a (x,y) coordinate. Assuming that two FFs with coordinates (x1, y1) and (x2, y2) are given, respectively, the distance between these two FFs is calculated by its Manhattan distance (|x1-x2|+|y1-y2|). The total length of the whole scan chain should be calculated by the accumulation of all the distances between successive FFs in the scan chain.

[Input Files]

Your program will take three input files, which are explained as follows.

1. Scan-chain declaration file (XXX.chn): this file is used the store the scan chain information. The corresponding format is shown as:

```
Cell1 <x1,x2> 3.75
Cell2 <x2,y2> 6.32
Cell3 <x3,y3> 2.41
CellA <x4,y4> 0.35
...
```

Figure 2: An Example of Scan-Chain Declaration File

Each line in this file refers to a scan cell and the line order represents the scan chain order. In each line, integer pair <x,y> denotes the coordinate of such scan cell and the last value represents the peak power consumption for the scan cell during logic transition. Particularly, the first line represents the scan cell connecting the scan input pin and the last line represents the cell connecting to the scan output pin. Both cells are not allowed to re-ordered.

2. Scan Pattern File (XXX.src): this file stores the pattern to be shift into the scan chain. The format is as the following:

```
010011001
110110110
001100101
...
```

Figure 3: An Example of Scan-Chain Pattern File

Each line in this file represents a pattern to be shifted into the scan chain. The least-significant bit (LSB) refers to the value to be shifted into the scan cell connected to the scan input, and the most-significant bit (MSB) refers to the value to be shifted into the scan cell connected to the scan output. Namely, the MSB is to be shifted first while the LSB is to be shifted last.

3. Configuration file (XXX.cfg): this file specifies the weights of the peak power, the total length of the scan chain and the execution time with the following format:

```
0.7
0.1
0.2
```

Figure 4: An Example of Configuration File

The first line represents the weight for the peak power after your reordering. The second line represents the weight for the total length of the scan chain. The third line represents the weight for the execution time for running your program. The details of the grading policy will be described later.

[Output Files]

Your program will generate three output files, which are explained as follows.

1. A scan-chain re-ordered file (XXX.ror): this file should be generated according to your re-ordering of this scan chain. The format must be the same as the input scan-chain declaration file.

2. A re-ordered scan-chain pattern (XXX.ptn) should be generated and the format must be as the same as the input scan-chain pattern.

3. A report file (XXX.rpt) should be generated with the following format:

```
Original:
Scan-Chain Length = 130
Max Peak Power = 37.20

Reordered:
Scan-Chain Length = 100
Max Peak Power = 24.00
```

Figure 5: An Example of Report File

[Grading Policy]

1. Five test cases (three public and two hidden) will be used to judge the efficiency of the scan re-order program, and each will be assigned 20% of the final score. The total score for a test case is 100.
2. If core dump occurs or your program runs more than 8-hours for a test case, the score for this test case will be 0. On the contrary, a valid and good reordering with no core dump and <8-hr runtime will score from 60.
3. As mentioned before, three indexes (i.e., maximum peak power, scan-chain length and execution time) will be used to grade your re-ordering program. The score calculation for each index follows a particular strategy below:
   All the execution results for a judge index for a test case generated from the submitted programs will be collected first. Then, the execution result will be normalized from 0 to 40 and the score on one test case that you will receive is based on this judge index after normalization.
   At the end, the total score for a test case is:

$$\text{Your score} = 60 + \text{weight}_{\text{PeakPower}} \times \text{score}_{\text{PeakPower}} + \text{weight}_{\text{ScanChainLength}} \times \text{score}_{\text{ScanChainLength}} + \text{weight}_{\text{ExecutionTime}} \times \text{score}_{\text{ExecutionTime}}$$

[Demonstration]

Your program will be compiled by the g++ compiler and executed on a Linux-based server with the operating system: openSUSE Leap 15.1. First, your program must be compiled as an executable code named pb01 by the Makefile. Second, three input files and three output files (for example: t1 in the testcase) should be specified in the command line during the execution. TA will execute your program as the following command.

> make

> ./pb01 t1.chn t1.src t1.cfg t1.ror t1.ptn t1.rpt

[FAQ]

1. Q: How do we calculate the transition among patterns?
   A: Assume that five FFs are used in the scan chain and initialized with the state 0. Given only two input patterns, P1: 01011 and P2: 11100, the transitions are show as below,

cycle	FF0	FF1	FF2	FF3	FF4	Note
0	0	0	0	0	0	initialization
1	**0**	0	0	0	0	scan in 0 from P1
2	**1**	**0**	0	0	0	scan in 1 from P1
3	**0**	**1**	**0**	0	0	scan in 0 from P1
4	**1**	**0**	**1**	**0**	0	scan in 1 from P1
5	**1**	**1**	**0**	**1**	**0**	scan in 1 from P1
6	**1**	**1**	**1**	**0**	**1**	scan in 1 from P2
7	**1**	**1**	**1**	**1**	**0**	scan in 1 from P2
8	**1**	**1**	**1**	**1**	**1**	scan in 1 from P2
9	**0**	**1**	**1**	**1**	**1**	scan in 0 from P2
10	**0**	**0**	**1**	**1**	**1**	scan in 0 from P2
11	?	**0**	**0**	**1**	**1**	
12	?	?	**0**	**0**	**1**	
13	?	?	?	**0**	**0**	
14	?	?	?	?	**0**	
15	?	?	?	?	?	

Since only two patterns are used, all ?'s are assigned with state 0 to prevent transition. As illustrated, cycle 4, 5 and 6 have three transitions on different FFs. Assume the peak-power array for these five FFs were given as [3.5, 5.7, 0.8, 1.3, 2.6]. Cycle 4 has 3.5+5.7+0.8=10.0 peak power, cycle 5 has 5.7+0.8+1.3=7.8 and cycle 6 has 0.8+1.3+2.6=3.9. Therefore, the max peak power over these three cycles are 10.0.