NCTU-EE IC LAB – Fall 2022

Lab02 Exercise

Design: Train Tour

Data Preparation

1. Extract test data from TA's directory: `% tar xvf train_ta/lab02.tar`

2. The extracted Lab02/ directory contains:
   a. Practice
   b. Exercise

Design Description

It has been 2 years since the outbreak of COVID-19. Now that more than half of the countries are gradually lifting lockdowns, you decide to plan a trip to Switzerland. You want to visit all the famous attractions while saving money. A train is the best way to explore Switzerland. But there are so many train lines to choose from: Bernina Express, Glacier Express, Gotthard Panorama Express…etc. You want to find the train line that costs the least.

To simplify the design, **the amount of train stations is limited to 16**. **A train station will be connected to at most 8 stations. Tracks are all bi-direction**, which means that if train station A is connected with train station B, trains can go either from A to B or from B to A. There may also be some abandoned train stations. It means that those train stations are unreachable. The train tickets from one station to another are all equal to **1**.

Inputs

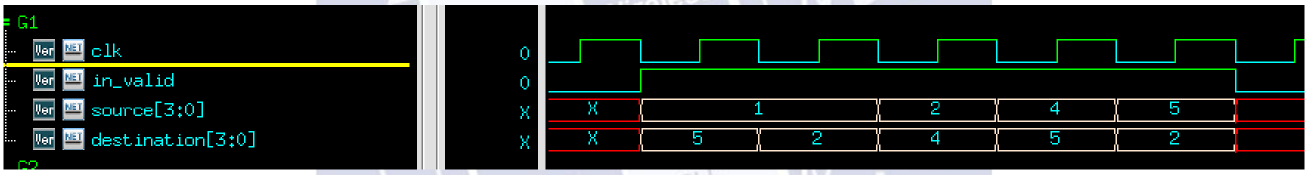
1. Input signals：

Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
source	4	**First cycle: your departure station**
		Following cycles: source station of a track
destination	4	**First cycle: your destination station**
		Following cycles: destination station of a track

2. The **source** [3:0] and **destination** [3:0] are valid only when **in_valid** is high.
3. All input signals will be synchronized at negative edge of the clock.
4. The first pair of **source [3:0] and destination [3:0]** indicates the **departure station and the destination station**. The following pairs indicate that there are **tracks connecting the station source and the station destination**.

- Example:

  Given input waveform as below:



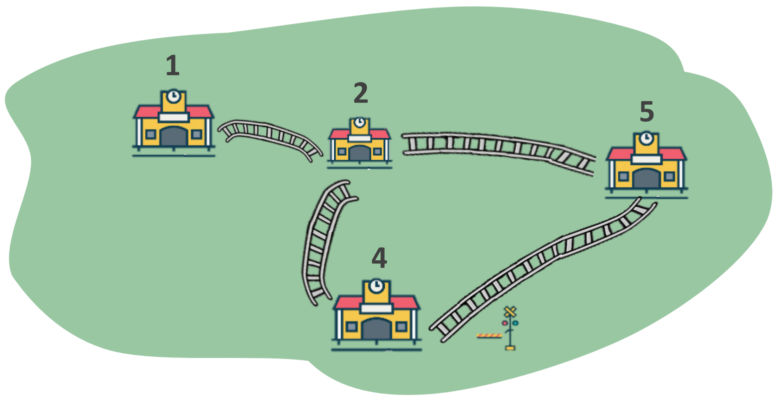
It means that you want to travel from station 1 to station 5.

And there are tracks between stations 1 and 2, 2 and 4, 4 and 5, 5 and 2.

To spend the least money, your travel route will be:

station 1 → station 2 → station 5

And the cost will be 2.



Outputs

1. Output signals:

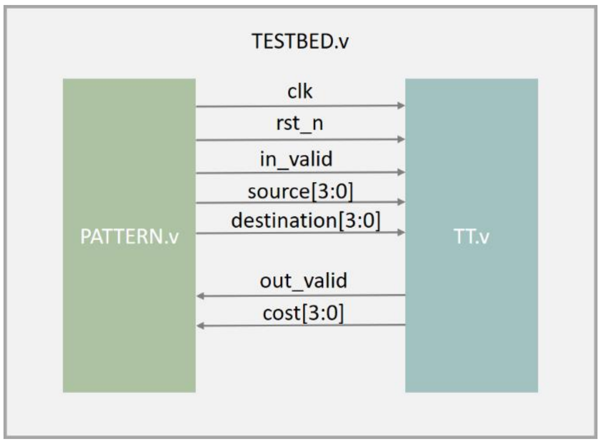
Output Signals	Bit Width	Definition
out_valid	1	High when output is valid
cost	4	Total cost from your departure station to destination station

2. All outputs should be low after initial reset.
3. Output signals should be synchronized at clock positive edge.
4. **out_valid** should not be raised when **in_valid** is high.
5. **out_valid** is set to high when **cost**[3:0] is valid, and will be high only for 1 cycle when be triggered.
6. **Return 0** if it is impossible to go from your departure station to your destination station.
7. TA's pattern will capture your output for checking at clock negative edge.

Specifications

1. Top module name：TT (Filename: TT.v)
2. It is an asynchronous reset and active-low architecture. If you use synchronous reset (reset after clock starting) in your design, you may fail to reset signals.
3. The clock period of the design is fixed to 7ns.
4. The next group of inputs will come in 2~5 cycles after your out_valid pull down.
5. The synthesis result of data type cannot include any LATCH.
6. After synthesis, you can check TT.area and TT.timing.
7. The slack in the timing report should be non-negative and the result should be MET.
8. The gate level simulation cannot include any timing violation.
9. The latency of your design in each pattern should not be larger than 30000 cycles.
10. Any words with "error", "latch" or "congratulation" can't be used as variable name.

Block Diagram



Note

1. **Grading policy:**

   RTL and gate-level simulation correctness: 70%

   Performance: 30%

   - (Area * (Execution Cycle+1)) 30%

2. **Please submit following files under 09_UPLOAD before 12:00 p.m. on Oct. 3:**

   - TT.v
   - If uploaded files violate the naming rule, you will get 5 deduct points.

3. **Template folders and reference commands:**

01_RTL/	(RTL simulation)	**./01_run**
02_SYN/	(Synthesis)	**./01_run_dc**

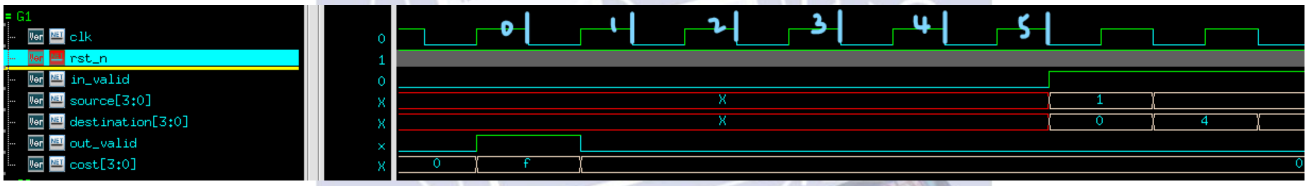
   (Check the design if there's latch or not in ***syn.log***)

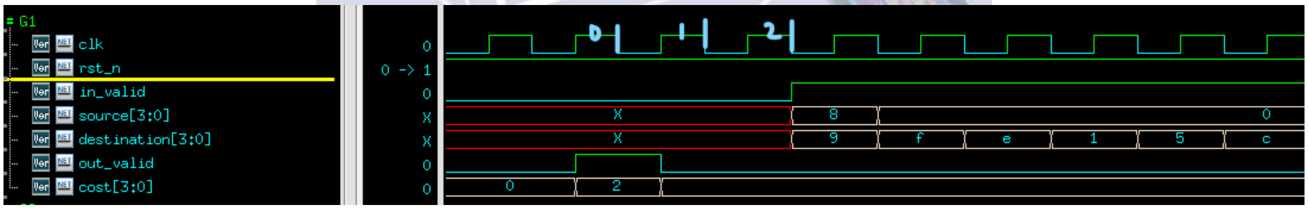
   (Check the design's timing in /Report/***TT.timing***)

03_GATE /	(Gate-level simulation)	**./01_run**
09_UPLOAD/(submit files)		**./01_submit**
09_UPLOAD/(check files)		**./02_check**

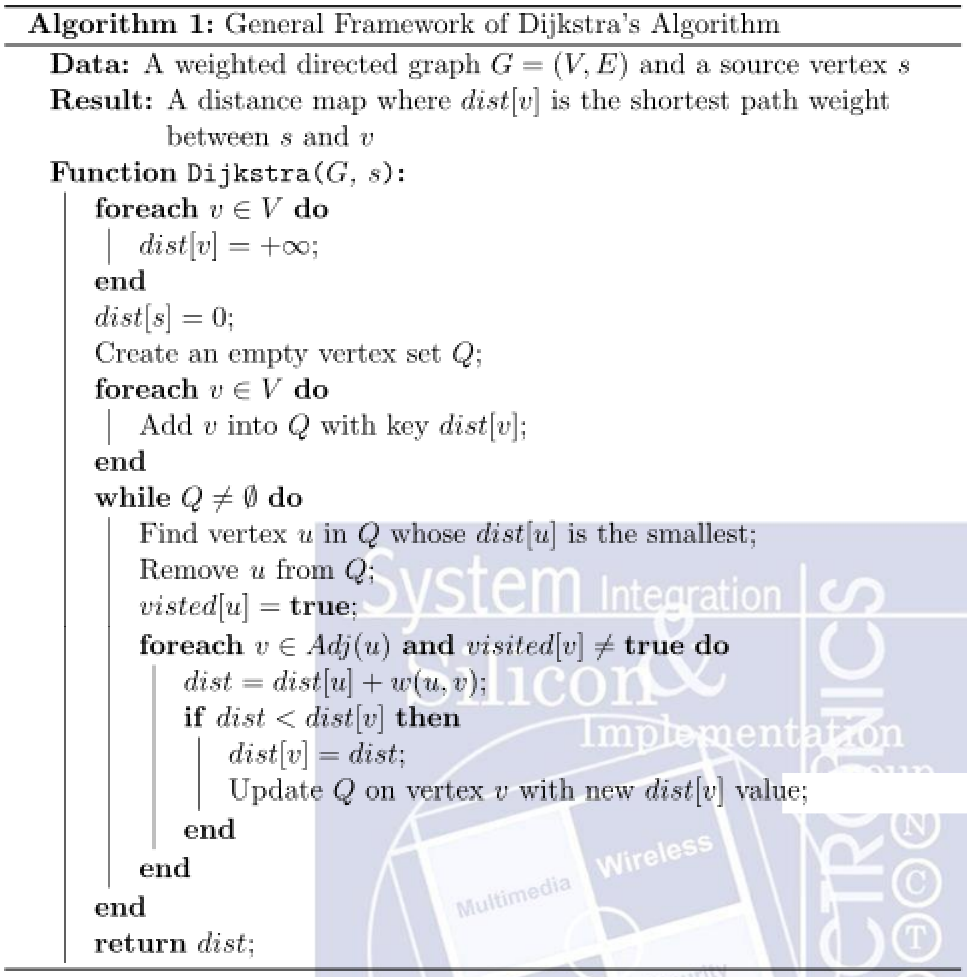
Sample Waveform

▪ The next in_valid will come in 2~5 cycles after out_valid pulls down.





Appendix



Algorithm 1: General Framework of Dijkstra's Algorithm

```
Data: A weighted directed graph $G = (V, E)$ and a source vertex $s$
Result: A distance map where $dist[v]$ is the shortest path weight
        between $s$ and $v$
Function Dijkstra($G$, $s$):
    foreach $v \in V$ do
        $dist[v] = +\infty$;
    end
    $dist[s] = 0$;
    Create an empty vertex set $Q$;
    foreach $v \in V$ do
        Add $v$ into $Q$ with key $dist[v]$;
    end
    while $Q \neq \emptyset$ do
        Find vertex $u$ in $Q$ whose $dist[u]$ is the smallest;
        Remove $u$ from $Q$;
        $visted[u] = $ true;
        foreach $v \in Adj(u)$ and $visited[v] \neq$ true do
            $dist = dist[u] + w(u, v)$;
            if $dist < dist[v]$ then
                $dist[v] = dist$;
                Update $Q$ on vertex $v$ with new $dist[v]$ value;
            end
        end
    end
    return $dist$;
```