Digital System Design and Implementation

Homework #3 (Due on 04/29 PM 8:00)

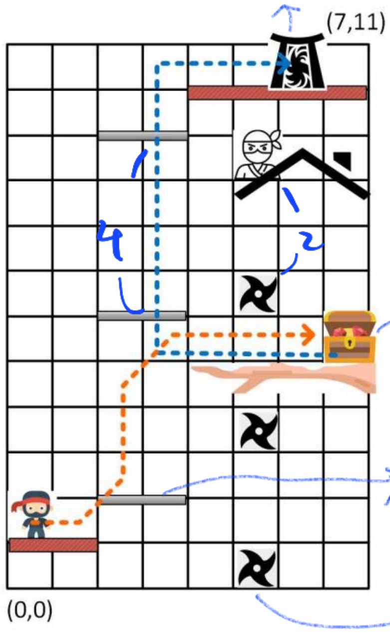
Note: Please upload your codes and hand in the **hardcopy** of this experiment including

a. Explanations about your design
b. Verilog codes (please use two-process or three-process in lecture notes to write the codes)
c. Test bench
d. Input/output waveforms (behavior and post-sim)

一定要用 2-process or 3-process

In this homework, we will implement a finite state machine for the game "**Ninja**". Assume that we have a rectangular area of size 12×8. The lower left corner is indicated as (0,0) and the upper right corner is indicated as (7,11). You need to control the **Ninja** to get the **treasure** together with the **key**, avoid the **shuriken** (手裡劍), and arrive at the exist **gate**. There is an **elevator** in this game. The **elevator** moves upward continuously. The spacing of each step is 4. Hence, there are always three **steps** on the screen. The **gate** is in (6,11). The **treasure** is in (7,5). One opponent throws **shuriken** every three clock cycles. All are shown in Fig. 1, which is used for even-numbered students.

(6.11) exit



→(7.5) treasure & key.

每個clk動一次. screen上 永遠有3個電梯

每3個clk丟一次

Fig. 1 The panel of game "**Ninja**"

Initially the player's **Ninja** starts from (0,1) (start). The player controls the left and right movement of the **Ninja**. The **Ninja** enters the "Elevation" state when he successfully take the elevator. In order to successfully take the elevator, the y position of the **Ninja** must be equal to or larger than the y position of **the closest elevator step** by 1. Otherwise, the **Ninja** will die for the drop. The **Ninja** is in the "Movement" state after the game starts. To get the **treasure** and the **key**, the **Ninja** must avoid the **shuriken** thrown by the opponent. If the **shuriken** touch the **Ninja**, he will die, too. The y position of each **elevator step** will be increased by 1 every clock cycles. The next y position of the **elevator step** is 0 if the current y position of the **elevator step** is 11. The y position of each **shuriken** will be decreased by 1 every clock cycle. The player has two chances for this "**Ninja**" game. When the **Ninja** takes the **treasure** and the **key**, the play gets 50 points. If the **Ninja** enters into the **gate** with the **key**, the player gets 50 points, too. If no **key**, the **Ninja** can not exist from the **gate**

input ×3

Input signals:

1. One **input** signal, **"Start"**, is used to indicate starting the game.
2. One **input** signal, "**GoRight**", is used to control the **Ninja** to move to the right, one step for one clock cycle.
   a. If "**GoRight**"=1'b0, **NinjaX**(i+1)= **NinjaX**(i).
   b. If "**GoRight**"=1'b1, **NinjaX**(i+1)= **NinjaX**(i)+4'd1.
3. One **input** signal, **"GoLeft"**, is used to control **the rocket** to move to the left, one step for one clock cycle.
   a. If "**GoLeft**"=1'b0, **NinjaX**(i+1)= **NinjaX**(i).
   b. If "**GoLeft**"=1'b1, **NinjaX**(i+1)= **NinjaX**(i)-4'd1.

output×14

Output signals:

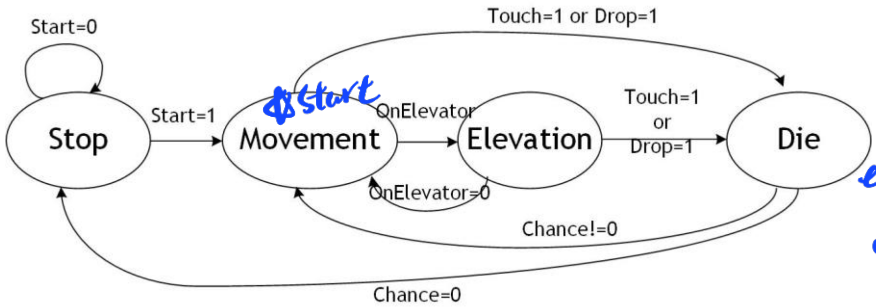
1. Two 4-bit **output** signals "**NinjaX**" and "**NinjaY**" indicate the current position of **the Ninja**. For example, after reset or the start of a new chance, the value of (**NinjaX**, **NinjaY**) in Fig. 1 is assigned to (0,1). The position can be changed in the "Movement" state and in the "Elevation" state.
2. Three 4-bit **output** signals "**Elevator1Y**", "**Elevator2Y**", and "**Elevator3Y**" indicate the positions of the respective **elevator steps**. The signals are updated in the "Movement" state and in the "Elevation" state. Note that **Elevator1Y** starts from 2; **Elevator2Y** starts from 6 and **Elevator3Y** starts from 10. When they are equal to 11, they will become 0 in the next clock cycle.
3. Three 4-bit **output** signals "**Shuriken1Y**", "**Shuriken2Y**", and "**Shuriken3Y**" indicate the positions of the respective **shuriken**. The signals are updated in the "Movement" state and in the "Elevation" state. Note that **Shuriken1Y** starts from 8; **Shuriken2Y** starts from 5 and **Shuriken3Y** starts from 2. When they are equal to 0, they will become 8 in the next clock cycle.

少一個 chance

4. One 1-bit **output** signal "**Touch**" denotes that **the Ninja** is touched by the **Shuriken.**
5. One 1-bit **output** signal "**Drop**" denotes that **the Ninja** does not take the elevator successfully. 7bits (更正)
6. One ~~6~~-bit **output** signal "**Score**" denotes the score of the player. 100 = 1100100 50 = 110010
7. One 2-bit **output** signal "**Chance**" indicates the chances that the player has. Initially, the player has two chances. If **Chance** =2'd0, game is over.
8. One 1-bit **output** signal "**Key**" to show that **the Ninja** takes the key.
9. One 1-bit output signal **"OnElevator"** denotes **the Ninja** takes the elevator successfully.

Now, the state diagram of this game is indicated as in Fig. 2. If key "**Start**" is pressed, then the game enters into state "Movement". If signal "**OnElevator**"=1'b1, the game enters into the state of "Elevation". If "OnElevator"=1'b0, the game goes back to the state "Movement". If "**Touch**"=1'b1 or "**Drop**"=1, the game enters into the state of "Die". If "**Chance**"!=2'd0, the game will enter into state "Movement". Otherwise, if "**Chance**"=2'd0, it will enter into state "Stop". start才会清掉

or 100分.



Stop 2'b00
movement 2'b01
elevation 2'b10
die 2'b11

Fig. 2 State diagram of this game.

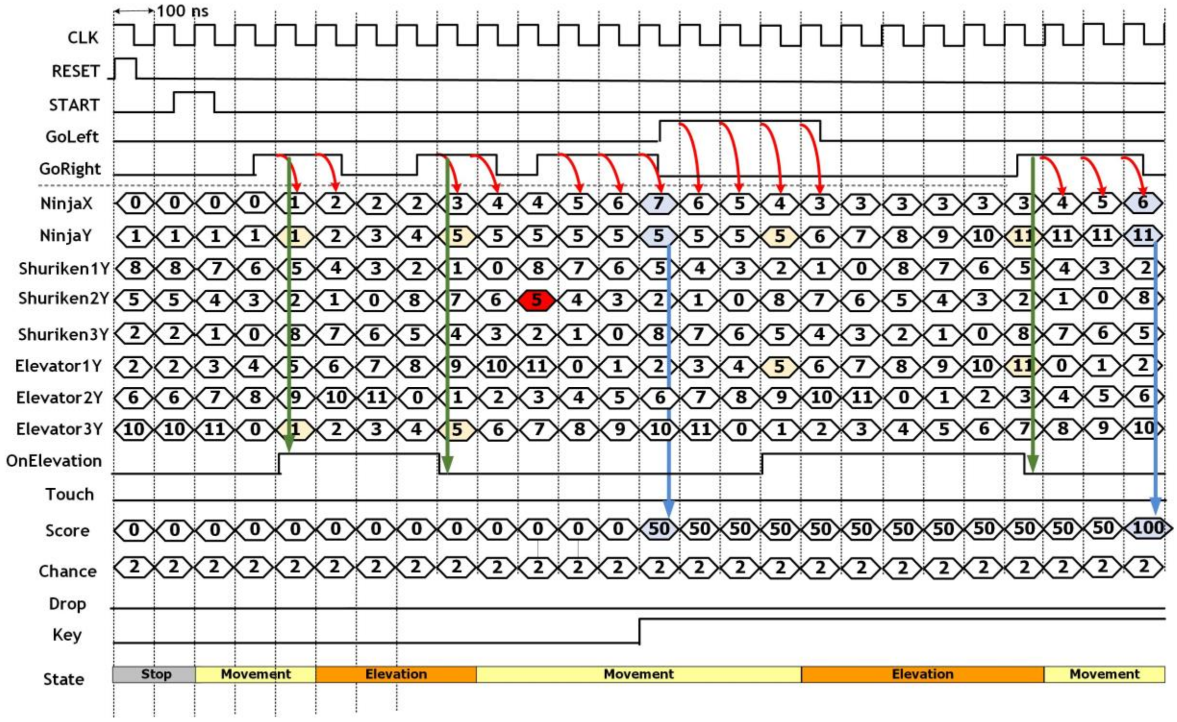
start pressed ⇒ points=0

Note that the points that the player gets are cleared only when key "**Start**" is pressed. When the state becomes "Movement" due to nonzero **"Chance**" after "Die" state**, the Ninja** is back to the same initial position.

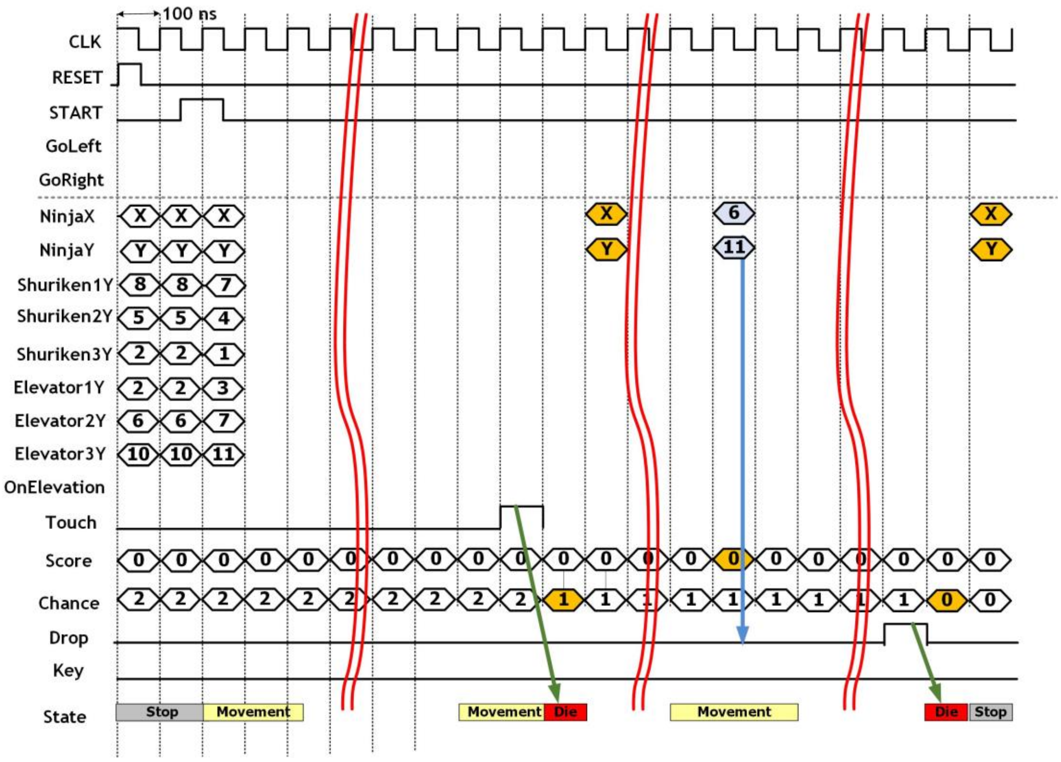
Die ⇒ points 不變

For the even-numbered students

1. **The Ninja** starts from position (0,1). First, **the Ninja** will take the elevator, avoids the **shuriken** and take the treasure and key to get 50 points as indicated by the dashed line Fig. 1. Then, the Ninja will avoid the shuriken and take the elevator to arrive at the exit gate to obtain a total of 100 points in this game. Send the correct input pattern and make sure that if your output patterns are exactly the same as the correct one in Fig. 3(a). The score should become 100 points.
2. Now please design your own test for verifying the chance and "Die" state. You can move the initial position of the **Ninja**, but the y position must be less than 5. This time, **the Ninja** will die due to touch the shuriken for the first chance. In the second chance, the Ninja will go the exist gate without the key to test for the function of key and then die due to drop. You will use up your two chances and check if the state goes back to "Stop" in one simulation. Please design your own travelling path and issue "**GoLeft**" and "**GoRight**" at correct time slots to complete the test. Please draw the figure to explain your test as well as your path and evaluate if your result is correct.
3. So you need to hand in the following items.
   a. Explain your design. (10%)
   b. Write Verilog codes for this finite state machine. (60%)
   c. Write a test bench following the input signals in Fig. 3(a) (15%) and Fig. 3(b) (15%)
   d. Show your output waveforms of the simulation results for Q1 and compare it to Fig. 3(a). (25%)
   e. Show the output waveform of Q2 and compare it to Fig. 3(b). (25%)



(a)



(b)

Fig. 3(a) Timing diagram example 1 of the game "**Ninja**". (b) Timing diagram example 2.

For the odd-numbered students.

4. **The Ninja** starts from position (6,0). First, **the Ninja** will take the elevator, avoids the

shuriken and take the treasure and key to get 50 points as indicated by the dashed line Fig. 1. Then, the Ninja will avoid the shuriken and take the elevator to arrive at the exit gate to obtain a total of 100 points in this game. Send the correct input pattern and make sure that if your output patterns are exactly the same as the correct one in Fig. 5(a). The score should become 100 points.

5. Now please design your own test for verifying the chance and "Die" state. You can move the initial position of the **Ninja**, but the y position must be less than 5. This time, **the Ninja** will die due to touch the shuriken for the first chance. In the second chance, the Ninja will go the exist gate without the key to test for the function of key and then die due to drop. You will use up your two chances and check if the state goes back to "Stop" in one simulation. Please design your own travelling path and issue "**GoLeft**" and "**GoRight**" at correct time slots to complete the test. Please draw the figure to explain your test as well as your path and evaluate if your result is correct.
6. So you need to hand in the following items.
   a. Explain your design. (10%)
   b. Write Verilog codes for this finite state machine. (60%)
   c. Write a test bench following the input signals in Fig. 5(a) (15%) and Fig. 5(b) (15%)
   d. Show your output waveforms of the simulation results for Q4 and compare it to Fig. 5(a). (25%)
   e. Show the output waveform of Q5 and compare it to Fig. 5(b). (25%)

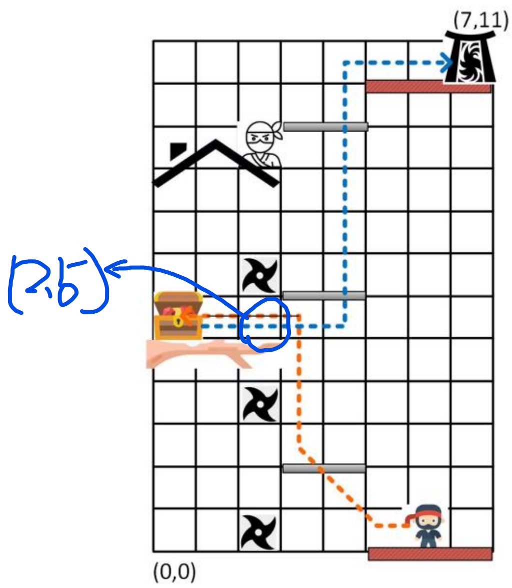
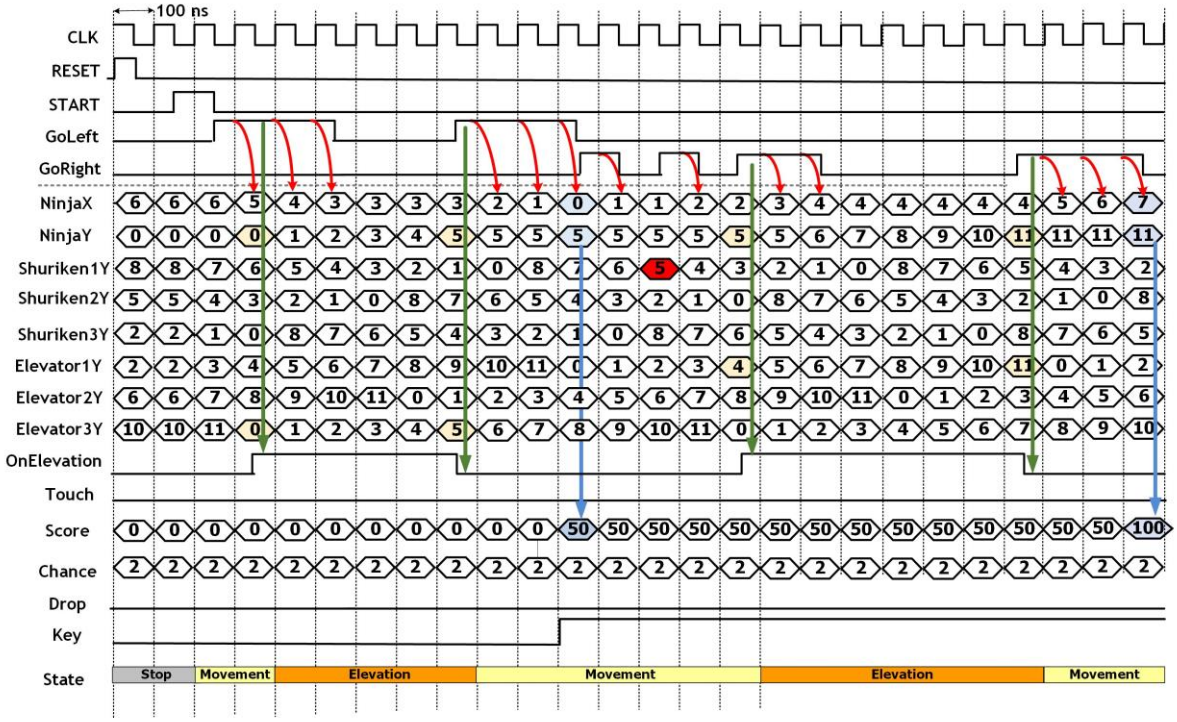
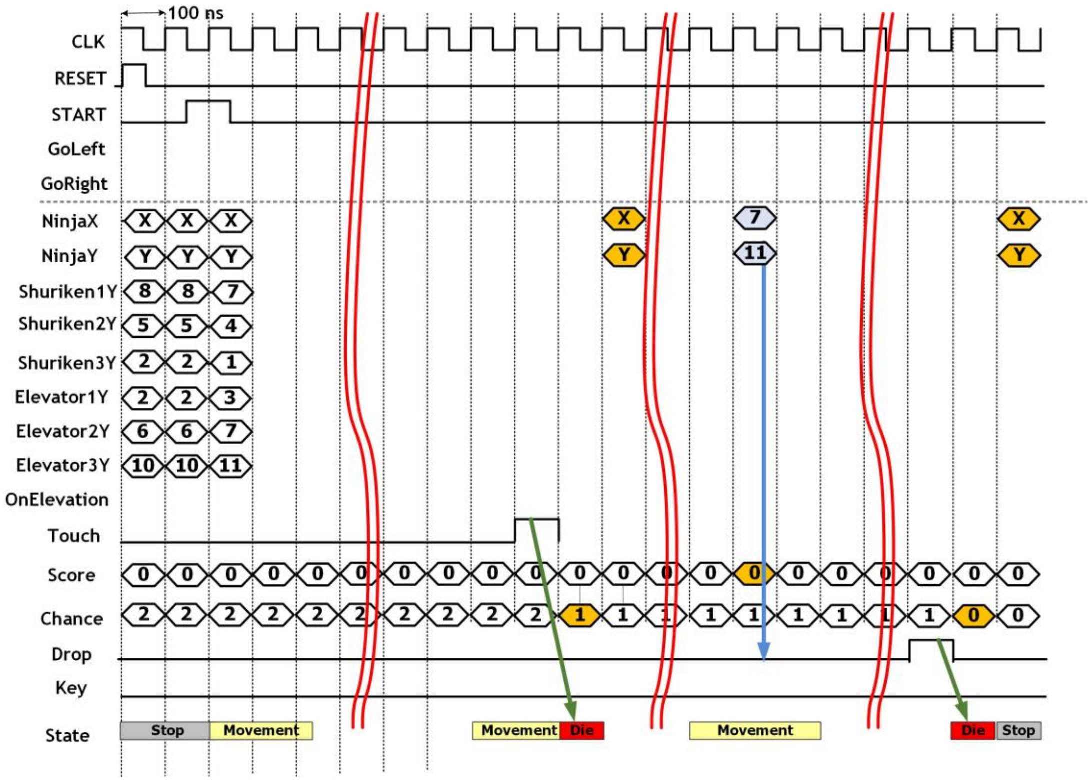


Fig. 4 Example of another settings for odd-numbered students.



(a)



(b)

Fig. 5 (a) Timing diagram example 3 of the game "**Ninja**". (b) Timing diagram example 4.