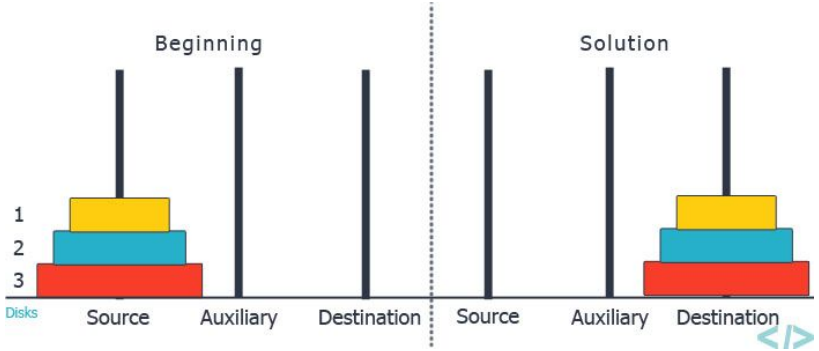
Formal Verification of Tower of Hanoi Game

CSEE 6863 Formal Verification of HW SW Systems

Project by:
Saher Iqbal si2443, MSEE

Goal and Understanding RTL

- **Given a system verilog goal for Tower of Hanoi, the goal was to first understand the RTL and verify it using Formal Verification methodology.**



- **In the RTL given, the game of Tower of Hanoi was implemented where the number of disks and number of rods both were 3, in such a way that the disks are placed in ascending order of their weights.**
- **In the code, two of the important variables were : top_of_rod & rod_data**
  - **top_of_rod indicates the height of each rod**
  - **rod_data indicates the weight of each disk on any rod**

Step I : Assertions

- **After understanding the RTL given, I worked on writing the assertions first. The assertions written were as follows:**
  - **The following assertion was written to ensure that upon reset, the RTL design functions correctly. In this assertion we check that the number of disks on the initial rod is set to total number of disks.**

```
// To ensure that on reset, all disks are on one rod
assert_no_movement_on_reset: assert property (@(posedge clk)
    if (rst) (top_of_rod[from_rod] == NUMBER_OF_DISKS));
```

  - **The way this assertion works, is that the variable *top_of_rod* indicates the height of the stack, so the height of the stack of the initial rod should be equal to the total number of disks on reset.**

Assertions contd.

- **The following assertion is meant to assure that the net amount of disks in the game's runtime remains always constant, this is achieved by checking the top_of_rod array stack for each of the 3 rods i.e (0,1,2). The net sum of the number of disks belonging to each of these stack should equate to the number of disks originally defined.**

```
// To ensure that the total number of disks across rods is same
assert_disk_count_consistency: assert property (
    @(posedge clk) (top_of_rod[0] + top_of_rod[1] + top_of_rod[2] == NUMBER_OF_DISKS)
);
```

Assertions contd.

- **As during the game, The design should not pick one disk from a particular rod and place it again to the same rod as this is a violation of the rules of the game. Thus, this assertion ensures that the disk being moved goes to any different rod other than the the current rod which it is picked from.**

```
// To ensure that the disk being moved from the from_rod to the to_rod is not the same during any point in the game
  assert_different_disk_move: assert property (
        @(posedge clk)
        if ((from_rod < NUMBER_OF_RODS) && (to_rod < NUMBER_OF_RODS) &&
            (from_rod != to_rod) && (top_of_rod[from_rod] > 0) &&
            (top_of_rod[from_rod] <= NUMBER_OF_DISKS) &&
            ((top_of_rod[to_rod] == 0) || (top_of_rod[to_rod] <= NUMBER_OF_DISKS)))
            ((top_of_rod[to_rod] == 0) ||
             (rod_data[from_rod][top_of_rod[from_rod] - 1] != rod_data[to_rod][top_of_rod[to_rod] - 1]))
    );
```

Assertions contd.

- **This assertion checks if the move is valid by checking if the to_rod is empty or the number of disks in to_rod are less than the total number of disks; the = sign was used as the property was failing during the transition phase, the reason is that even when all the disks have transferred to to_rod, the control loop of program goes once again, before ending, in this phase the assertion is executed again, and thus to satisfy the assertion with the latest end values, we use = sign, as the final rod has all the disks now.**
  (Towards the end of the program, the assertion is checked again)

```
//To ensure a valid move to to_rod
  assert_valid_move: assert property (
        @(posedge clk)
        ((top_of_rod[to_rod] == 0) || (top_of_rod[to_rod] <= NUMBER_OF_DISKS))
    );
```

Step II : Assumptions

- **The following assumption is to ensure that the game starts with all the disks placed at the first rod during the reset state.**

```
//On reset, all disks are on the first rod
assume_all_disks_on_first_rod: assume property (
@(posedge clk)
if (rst) (top_of_rod[0] == NUMBER_OF_DISKS) &&
     (top_of_rod[1] == 0) &&
     (top_of_rod[2] == 0)
);
```

Assumptions contd.

- **In the design, the from_rod and to_rod are represented using 2 bits to represent the combination 00, 01, 10, thus we do not want the vale 11 to be used anywhere, the following assumption is written to ensure this.**

```
//  Ignore the combinations - 11 in binary for number of disks and rods
assume_ignore_combination_3: assume property (
  @(posedge clk)
  if (!rst) (
    (from_rod != 2'b11) && (to_rod != 2'b11) && (DISKS_LOG2 != 2'b11)
  )
);
```

Assumptions contd.

- **The following assumption is to ensure that there is at least one disk in the rod from which acts as the ‘from_rod’.**

```
// Source rod always has at least one disk
assume_source_rod_not_empty: assume property (
  @(posedge clk)
  if (!rst) (
    (from_rod < NUMBER_OF_RODS) &&
    (top_of_rod[from_rod] > 0)
  )
);
```

Assumptions contd.

- **This assumption is written to assume the constraints on the number of disks on each rod, this was written as a few of the assertions were giving an out of bound error.**

```
//To assume constrains on the number of disks on a rod
assume_top_lessthan3: assume property(
    @(posedge clk)
    if(!rst) (
    top_of_rod[from_rod] <= NUMBER_OF_DISKS && top_of_rod[to_rod] <= NUMBER_OF_DISKS && top_of_rod[from_rod] >= 0 && top_of_rod[to_rod] >= 0)
    );
```

Step III : Cover Properties

- **Following cover properties check that all of the rods are used during the game and that the game is finally completed at any stage.**

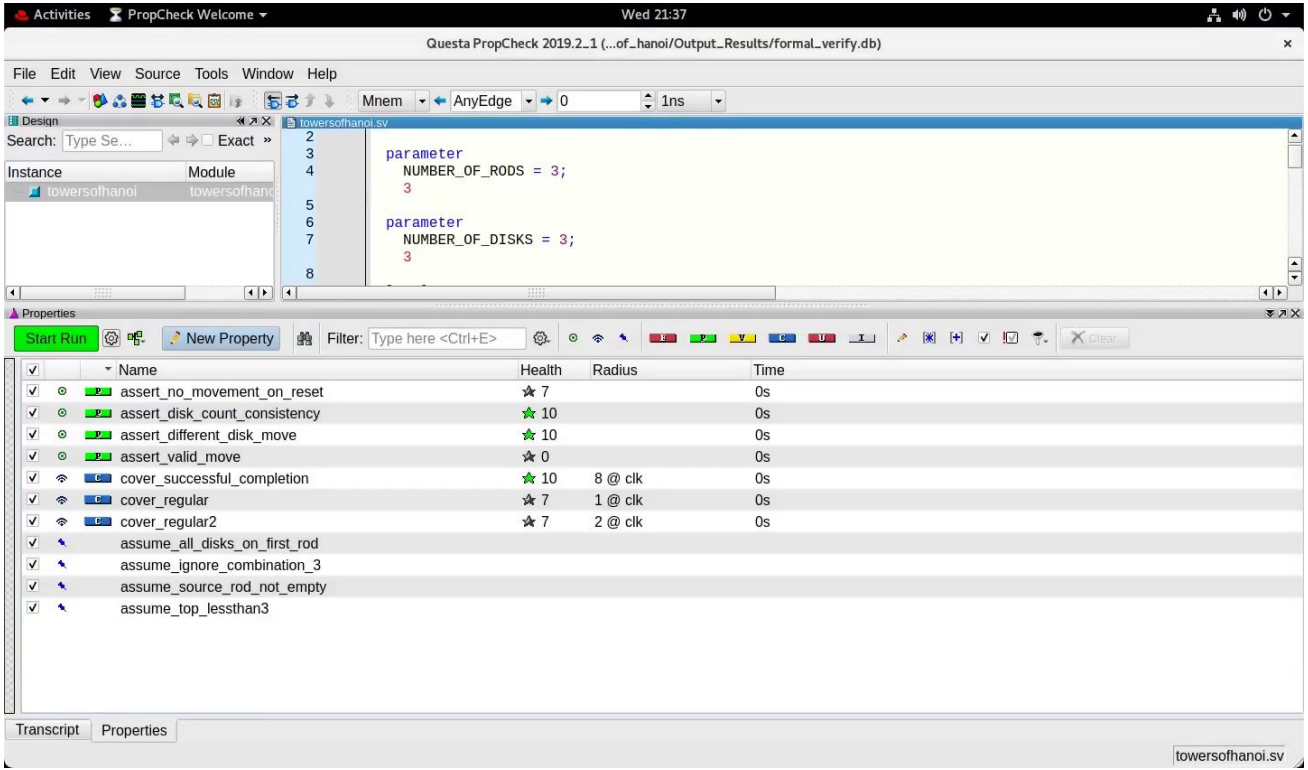
```
// Cover property for successful completion of the game
cover_successful_completion: cover property (
  @(posedge clk)
  if (!rst) (
    ((top_of_rod[1] == NUMBER_OF_DISKS || top_of_rod[2] == NUMBER_OF_DISKS))
  )
);
```

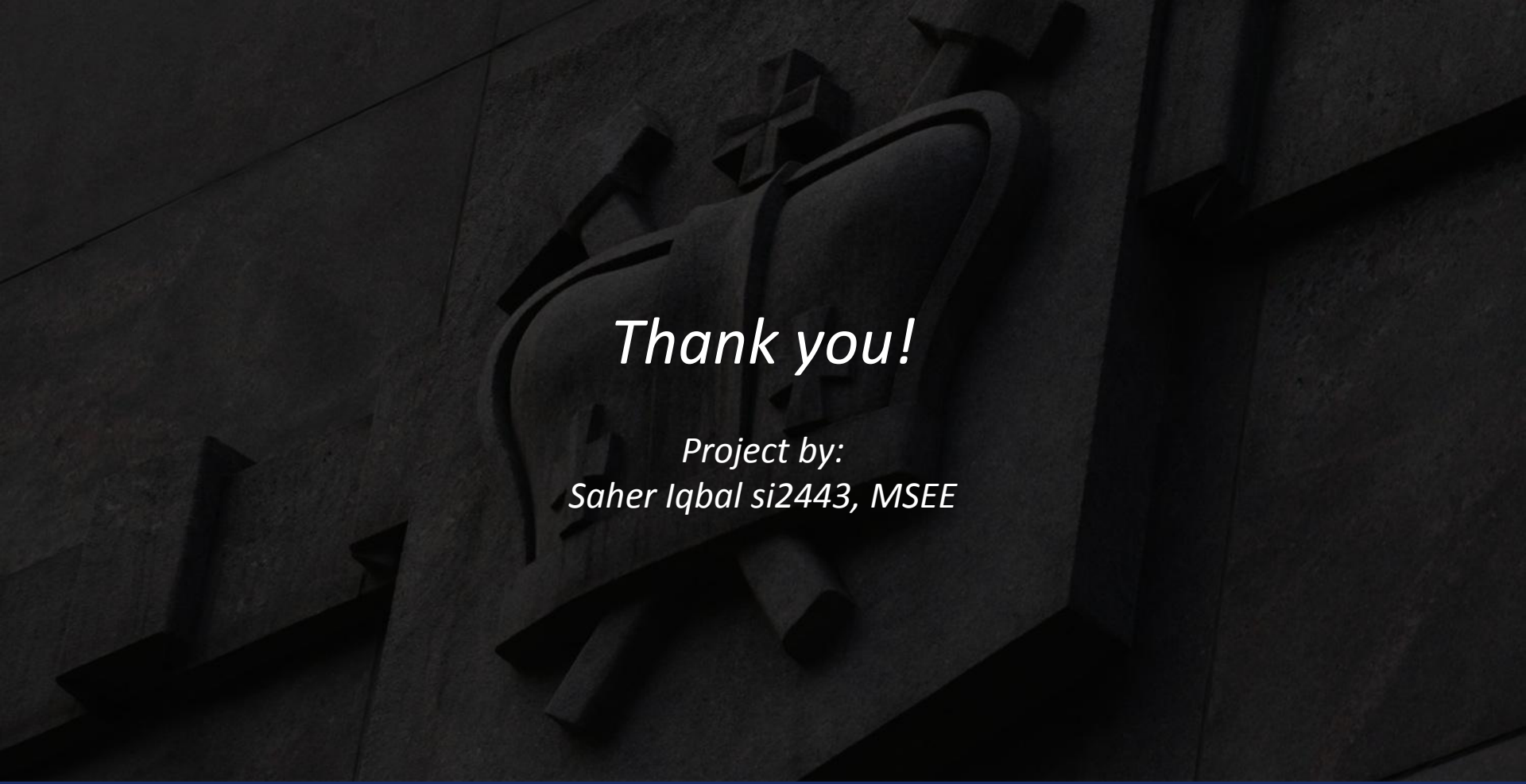
```
//Cover properties to check if all the rods were used
cover_regular: cover property (@(posedge clk) from_rod == 0 && to_rod == 1);

cover_regular2: cover property(@(posedge clk) from_rod == 1 && to_rod == 2);
```

Results

- **Following are the results using the above mentioned assertions, assumptions and cover properties:**




Thank you!
Project by:
Saher Iqbal si2443, MSEE