Examen 12 Marzo 2025

1. What is verilog?
   - Verilog es un lenguaje de descripción de hardware

2. How many values can take an output port? 1/2
   - Un puerto de salida puede tomar valores ~~reales, integers,~~ de 0 o 1. → (x, z)

3. What is the purpose of a testbench in verilog?
   - Es probar y verificar el funcionamiento del programa mediante pulsaciones simuladas, que permiten errores y/o salidas esperadas.

4. How do you instantiate a module in verilog?
   - Se manda llamar el modulo mediante su nombre y se colocan ya sean variables del módulo donde se quiere instanciar o variables auxiliares.

```
module nombre (.a1(A_1), .b2(B_2), c3(C_3));
       ^nombre instancia   ^variables dentro del modulo "module"
```

5. What is the purpose of a wire in verilog?
   - Debido a que wire es un cable físico que une componentes mediante conexiones, su propósito es la interconexión dentro de módulos.

6. Difference between blocking and non-blocking assigments.
   - Los blocking assigments se representan mediante un (=) y los non-blocking mediante (=>). Su principal diferencia es que los non-blocking no interfieren con el flujo del programa. De igual manera, dentro de bloques "always", su uso se puede observar en la asignación de valores a "diferentes" flancos de subida de reloj.

Elizabeth Jauregui Zarate (A01253381)

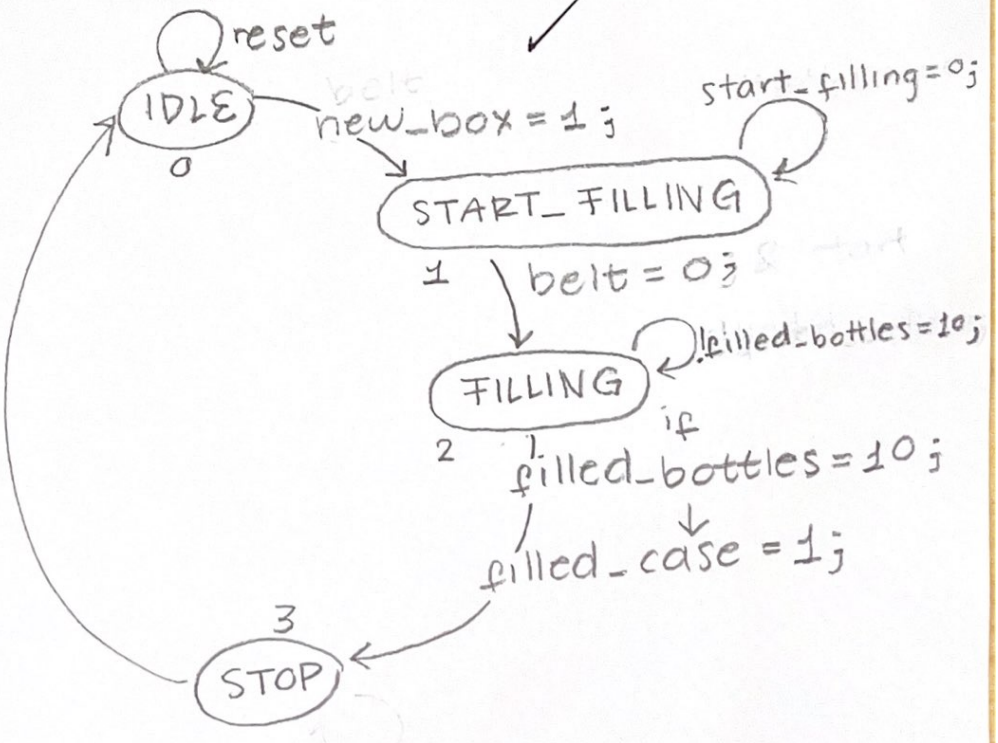
7. Define an always block. (siempre)
   - Un bloque "always" se puede entender como un bucle que se realiza si la condición dentro de su sensitivity list se cumple. Esto se puede ver como el siguiente ejemplo.

```
always @(posedge clk)
  begin
    :::::
  end
```

(posedge clk) = sensitivity list ✓

8. I want to create a module that counts between 0 and 100, what length of bits need in the internal register used to count? ✓
   - $2^7 = 128$ → debido a que se necesitan (hasta) 100 valores, se necesita un registro de [0:6] bits.

9. Define a FSM diagram for a bottle packer that performs the following steps: ✓



10. Define a structure of Finite State Machine Module

- Se declara como modulo, se introducen parametros locales que representan los estados de la FSM.
  Se divide en bloques:
  1. Bloque que describe el cambio del estado actual al siguiente estado.
  2. Bloque que describe la salida de cada estado.

```
module nombre (variables);
  localparameters  IDLE=0, START_FILLING=1, ... ;

  always @(posedge clk)
    begin
    .....
    end
```

} 1er bloque always que cambia el estado de current_state => next_state;

```
  always @(posedge clk)
    begin
    .....
    end
```

} 2do bloque always que describe el cambio de estado

```
  always @(posedge clk)
    begin
    ....
    end
```

} 3er bloque always que describe las salidas de cada estado.

```
endmodule
```

Sum Parcial

Cronometro Parcial