**Task 1:**

1. Latch:

* Code:

|  |
| --- |
| library IEEE;  use IEEE.std\_logic\_1164.all;  entity latch is  port (  D : in std\_ulogic;  clk : in std\_ulogic;  Q : out std\_ulogic);  end entity latch;  architecture rtl of latch is  begin  process(clk) is  begin  if clk='1' then  Q <= D;  end if;  end process;  end architecture rtl; |

* Schematic

**A diagram of a graph

Description automatically generated with medium confidence**

2. Examples:

* If statement without else (or default value before it).
* Case statement that does not specify all cases and without default case.

3. For the mentioned examples:

* Specify an else (or default value before the if)
* Use default case.

**Task 2:**

1. Design 1 will make “t” as a register because the right hand side of the “y<=t;” statement will need to be evaluated before the “t := a xor b;”.

* Design 1 utilization: 2 FF are created

A close-up of a sign

Description automatically generated

A diagram of a circuit

Description automatically generated

* Design 2 utilization: 1 FF is created

A diagram of a computer

Description automatically generated

3. It is different for signals and variables

* Signals: a FF is generated is when the signal is assigned in a clocked process
* Variables: a FF is generated when the variable is read before it is assigned.

4. For reading

* Signals: yes
* Variables: normally, not visible outside a process

For writing

* Signals: no, illegal (contention)
* Variables: same answer

**Task 3**

1. It is synthesizable.
2. The synthesized circuit will be a MUX (shown below). However, the simulation will not show the changes on “b” if “a” is stable at value “1”, which is not how a the a MUX works.

A diagram of a circuit

Description automatically generated

1. Put “b” in the sensitivity list

**Task 4**

1. In the table below, the design to the left (with complete sensitivity list) does not proceed in simulation time, which hints to the problem. However, the design to the right works and does not show hints for an error.

|  |  |
| --- | --- |
| architecture behave of comb\_loop is  begin  process (a,b) is  begin  b <= a and not b;  end process;  y <= b;  end behave; | architecture behave of comb\_loop is  begin  process (a) is  begin  b <= a and not b;  end process;  y <= b;  end behave; |
|  |  |

1. When the signal/variable generates a FF.