Review on Synthesizable VHDL

Assignment Project Exam Help
https://tutorcs.com

WeChat: cstutorcs

Steve Wilton

University of British Columbia

Introduction

For the most part, in this course, you are writing a fairly small VHDL Description

For larger designs, if you aren't *really* careful, it is likely that when you try to compile your code to hardware**, it won't work!**

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Why?

I'll tell you in this slide set, and also talk about how to make sure it does work.

Why is this so important?

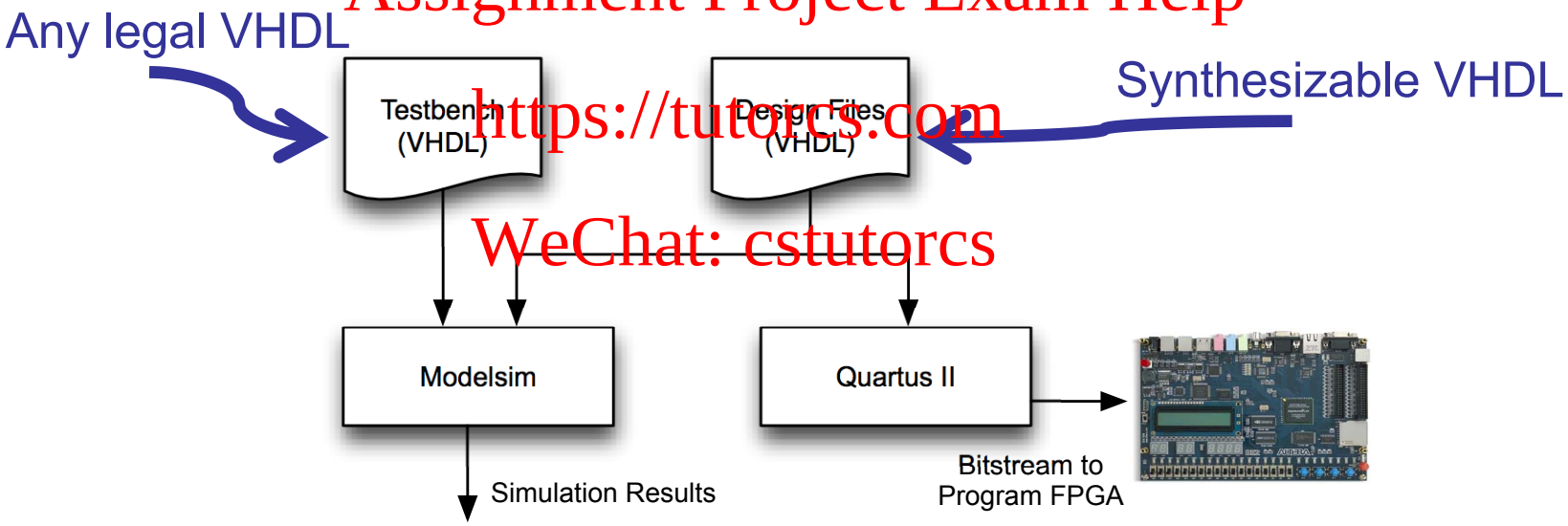
Because, it will save you tons of debugging time in the lab if you understand it.

Recall: What is VHDL?

VHDLs serves two roles:

- Synthesis – Describe hardware that you ultimately want to create
- Simulation – Describe hardware for simulation, and describe tests

Assignment Project Exam Help



https://tutorcs.com
WeChat: cstutorcs

Subset of VHDL that can be synthesized is called synthesizable VHDL

- Many legal VHDL constructs can not be synthesized

"Synthesizable" VHDL

Not all VHDL Code can be synthesized by current tools.

- This isn't limited to our tools

Synthesizable VHDL is a subset of VHDL that can be synthesized by current tools.

Assignment Project Exam Help

If you write VHDL that is not synthesizable:

https://tutorcs.com

- Tools will not be able to create hardware
  - Sometimes it will try, but end up with something that is "not quite right"
  - Sometimes it gives an error message, sometimes not!

WeChat: cstutorcs

Moral: if you are going to synthesize, always write Synthesizable VHDL!

Synthesizable Language Constructs

- Entities/Architectures
- Signals
- Concurrent Signal Assignments
- Component Instantiations
- Processes
  - If/else Conditional Statement
  - Case Statement

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

These are generally synthesizable …

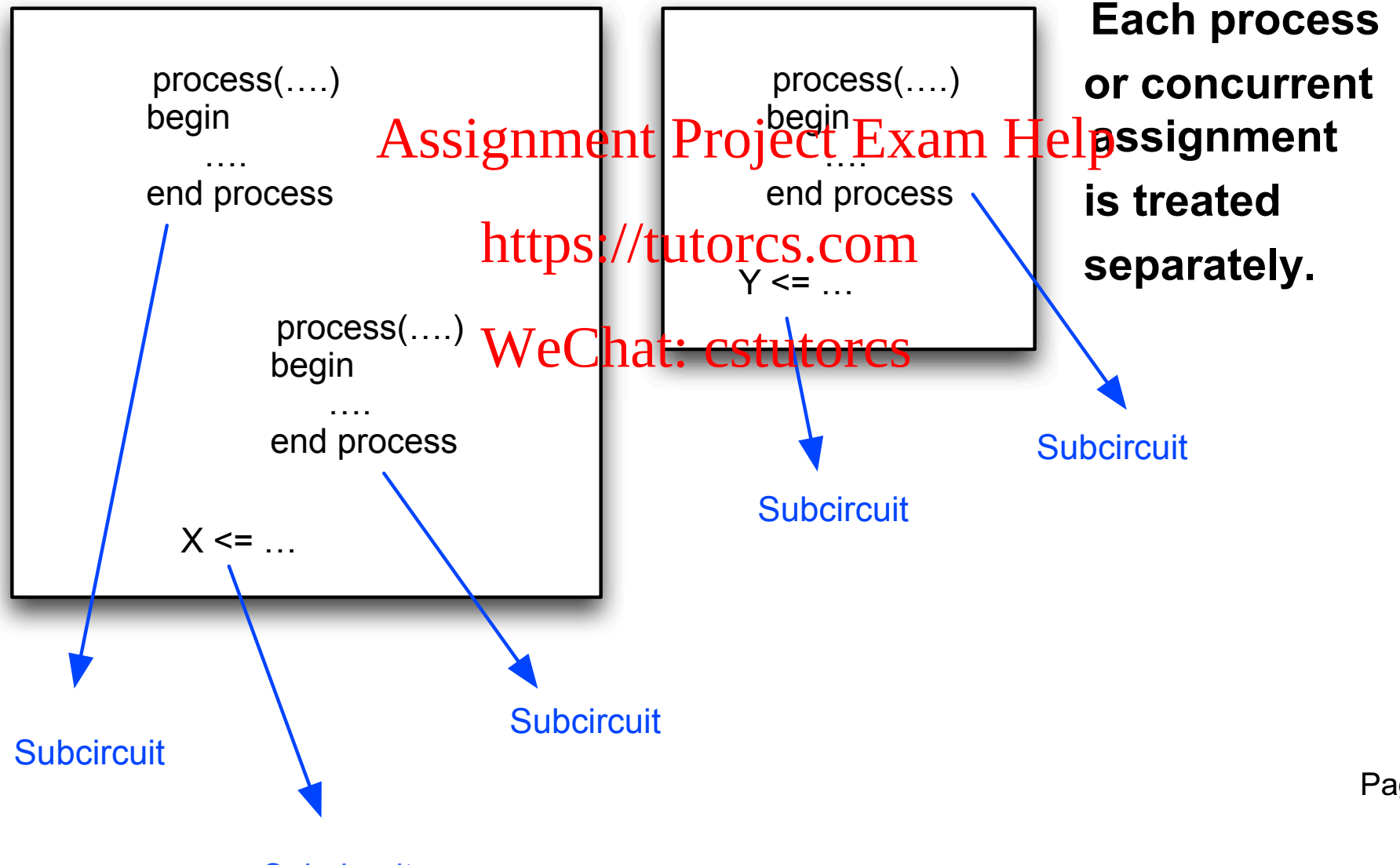
To be Synthesizable, there needs to be a way to determine an equivalent gate-level implementation of the construct

For all of the constructs above, there is a straight-forward "recipe" to determine the gate-level implementation …

…except for the PROCESS

A modern synthesis tool:

- Extracts processes and concurrent assignments from the code
- Converts ***each*** *process and concurrent assignment* to ***a piece of hardware***



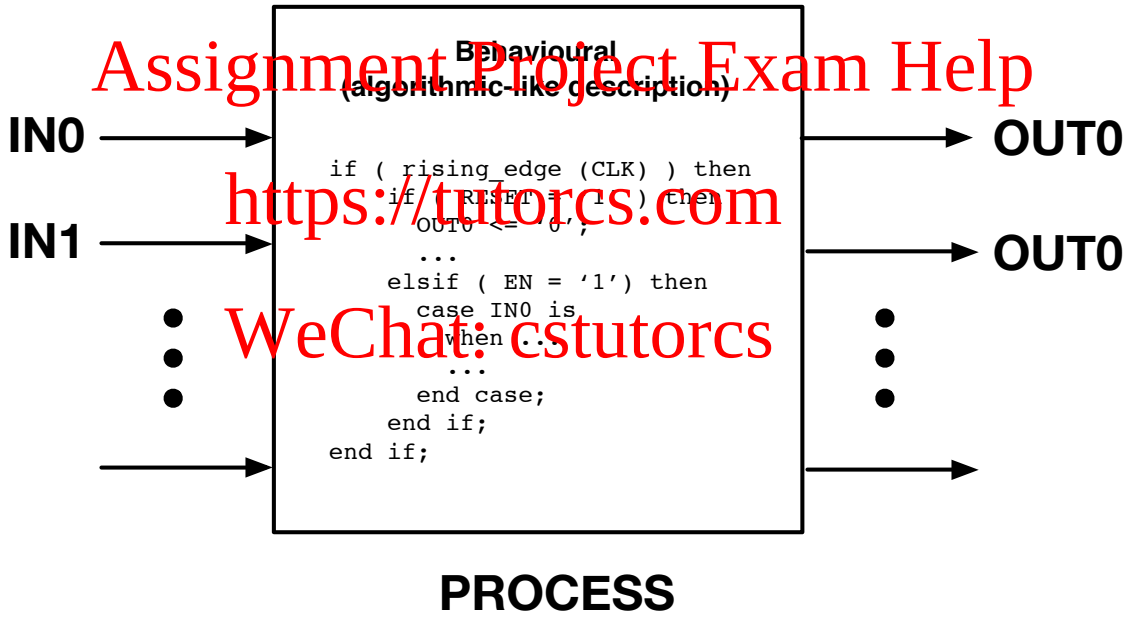
Each process or concurrent assignment is treated separately.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Synthesis of VHDL Processes

We use the VHDL Process to **model** the behaviour of a block of hardware.

The synthesis tool then **tries its best** to find a hardware implementation that matches this behaviour.



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

It's possible to write a process in such a way, that it becomes non-synthesizable

Three Coding Patterns

Synthesis tools use **PATTERN MATCHING** on each process to determine its hardware implementation

Three patterns that ALL synthesis tools can understand

1. Purely Combinational
2. Sequential
3. Sequential with asynchronous reset

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

ANY PROCESS THAT YOU WRITE MUST USE ONE OF THESE THREE PATTERNS

Pattern 1. Purely Combinational

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

1. Purely Combinational

Process outputs are a function of the current input values

Rule 1A:

- Every input to the process must be in the sensitivity list (or in VHDL 2008, the sensitivity list must only contain the keyword **all**)

Assignment Project Exam Help

Rule 1B:

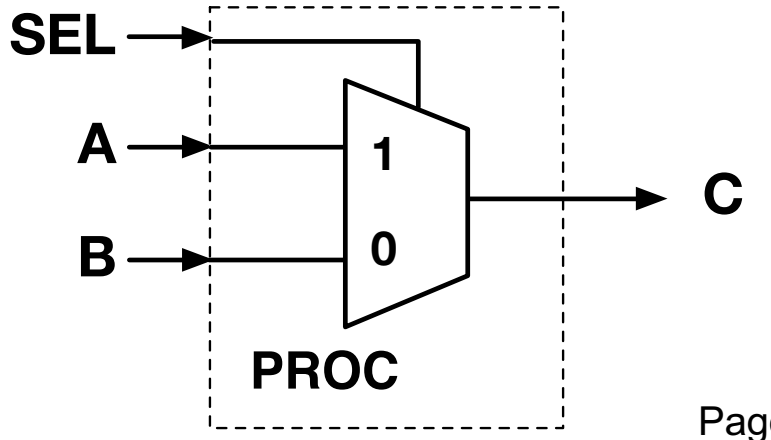
- Every output must be assigned a value for every possible combination of inputs.

https://tutorcs.com

- In other words, every output must be assigned a value for every possible path through the process' description

WeChat: cstutorcs

```
PROC: process (A, B, SEL)
begin
    if (SEL = '1') then
      C <= A;
    else
      C <= B;
    end if;
end process;
```



1A. Sensitivity List Rule

Rule 1A:

- Every input to the process must be in the sensitivity list

If you break this rule, you are saying…

… "If an event occurs on the input signal, do not immediately update the output"

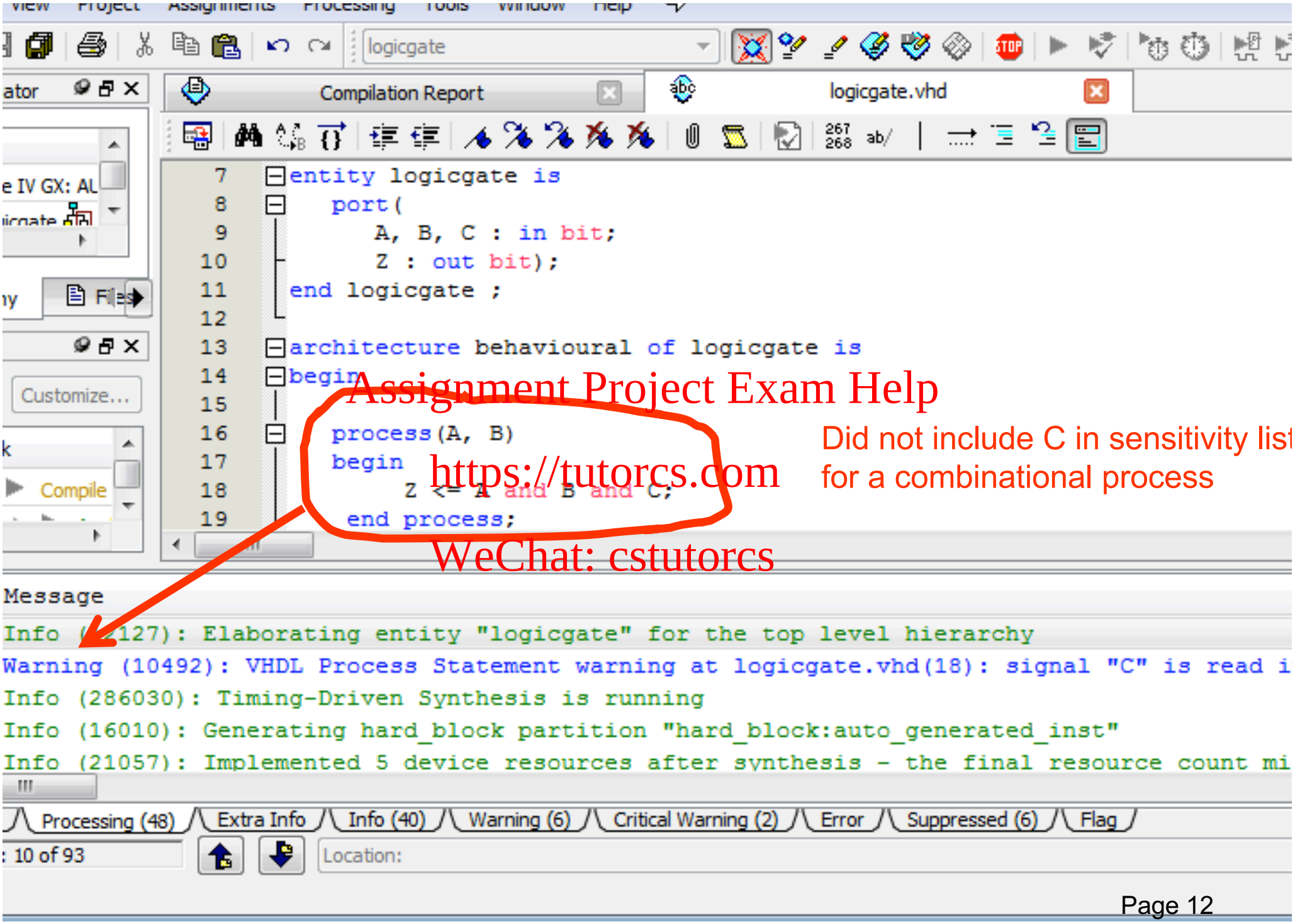
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

```
-- Note: SEL is missing
-- from sensitivity list
PROC: process (A, B)
begin
    if (SEL = '1') then
      C <= A;
    else
      C <= B;
    end if;
end process;
```

From Quartus:

Warning (10492): VHDL Process Statement warning at testtest.vhd(15): signal "SEL" is read inside the Process Statement but isn't in the Process Statement's sensitivity list

Quartus is actually smart enough in this case, but **don't rely on this in general**. And your **simulations will definitely be wrong.**


logicgate
Compilation Report
logicgate.vhd
7 entity logicgate is
8 port(
9 A, B, C : in bit;
10 Z : out bit);
11 end logicgate ;
12
13 architecture behavioural of logicgate is
14 begin
15
16 process(A, B)
17 begin
18 Z <= A and B and C;
19 end process;
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs
Did not include C in sensitivity list
for a combinational process
Message
Info (12127): Elaborating entity "logicgate" for the top level hierarchy
Warning (10492): VHDL Process Statement warning at logicgate.vhd(18): signal "C" is read i
Info (286030): Timing-Driven Synthesis is running
Info (16010): Generating hard_block partition "hard_block:auto_generated_inst"
Info (21057): Implemented 5 device resources after synthesis - the final resource count mi
Processing (48)
Extra Info
Info (40)
Warning (6)
Critical Warning (2)
Error
Suppressed (6)
Flag
10 of 93
Location:

1B. Output Assignment

Rule 1B:

- Every output must be assigned a value for every possible combination of input values.
- In other words, every output must be assigned a value for every possible path through the process' description

Assignment Project Exam Help

If you break this rule, you are saying…

https://tutorcs.com

… "For the combination of input values where we don't assign a value to the output, the output should remember its old value."

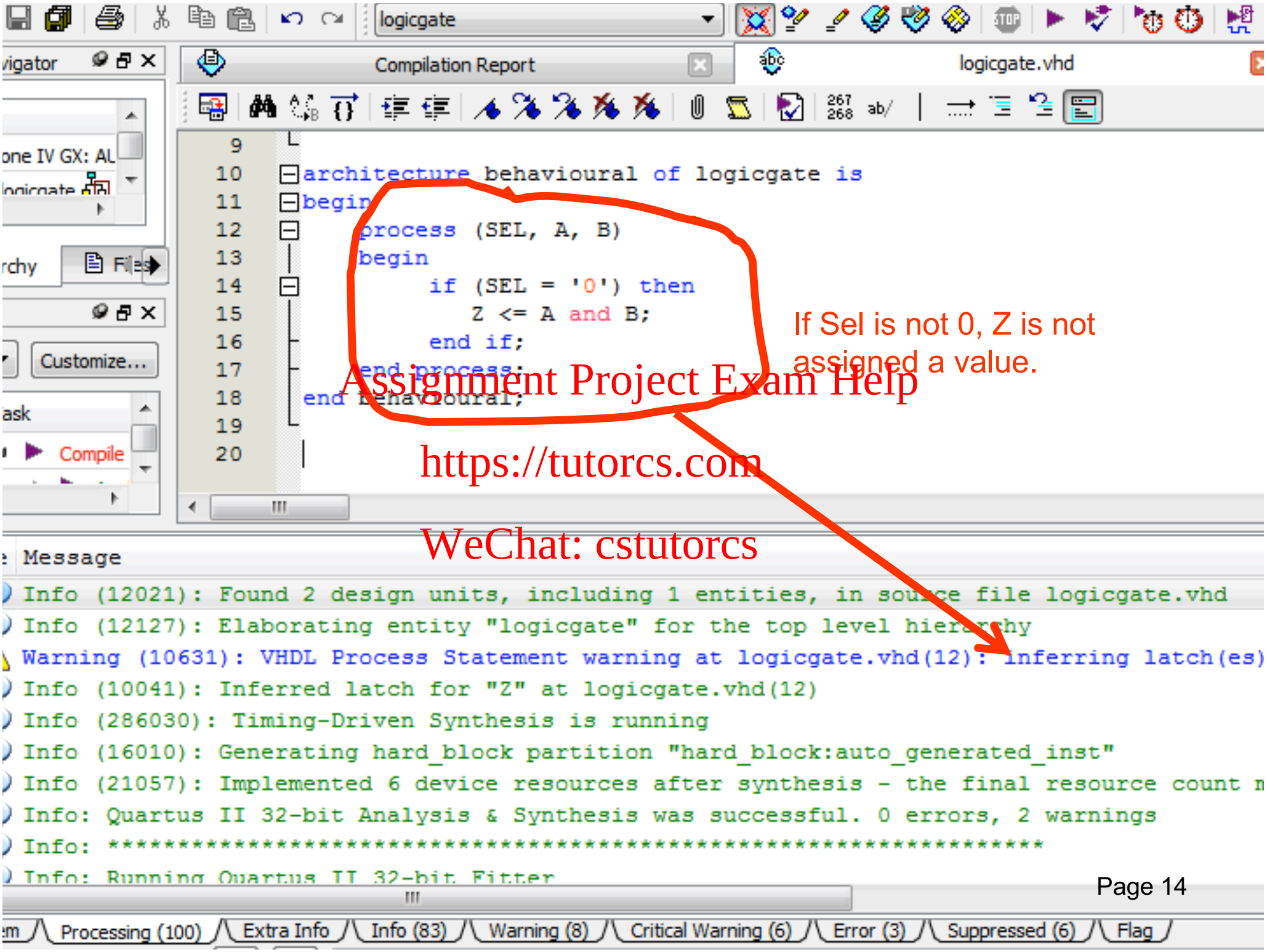
WeChat: cstutorcs

→ **Memory is implied → Sequential**

```
PROC: process (A, B, SEL)
begin
    if (SEL = '1') then
      C <= A;
    end if;
end process;
```

From Quartus:

Warning (10631): VHDL Process Statement warning at testtest.vhd(13): inferring latch(es) for signal or variable "C", which holds its previous value in one or more paths through the process


logicgate
Compilation Report
logicgate.vhd
9
10 architecture behavioural of logicgate is
11 begin
12 process (SEL, A, B)
13 begin
14 if (SEL = '0') then
15 Z <= A and B;
16 end if;
17 end process;
18 end behavioural;
19
20
If Sel is not 0, Z is not assigned a value.
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs
Message
Info (12021): Found 2 design units, including 1 entities, in source file logicgate.vhd
Info (12127): Elaborating entity "logicgate" for the top level hierarchy
Warning (10631): VHDL Process Statement warning at logicgate.vhd(12): inferring latch(es)
Info (10041): Inferred latch for "Z" at logicgate.vhd(12)
Info (286030): Timing-Driven Synthesis is running
Info (16010): Generating hard_block partition "hard_block:auto_generated_inst"
Info (21057): Implemented 6 device resources after synthesis - the final resource count
Info: Quartus II 32-bit Analysis & Synthesis was successful. 0 errors, 2 warnings
Info: **********
Info: Running Quartus II 32-bit Fitter
Processing (100)
Extra Info
Info (83)
Warning (8)
Critical Warning (6)
Error (3)
Suppressed (6)
Flag

1B. Output Assignment

Rule 1B:

- Every output must be assigned a value for every possible combination of input values.
- In other words, every output must be assigned a value for every possible path through the process' description

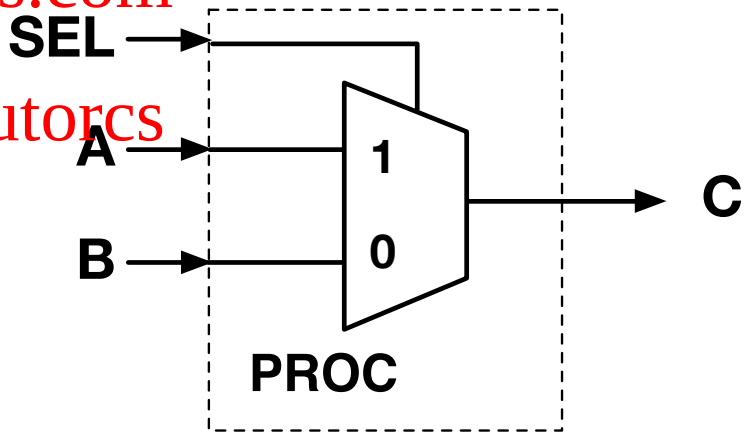
Assignment Project Exam Help

The following works though:

https://tutorcs.com

```
PROC: process (A, B, SEL)
begin
    C <= B;
    if (SEL = '1') then
      C <= A;
    end if;
end process;
```

WeChat: cstutorcs



Make sure you understand why!

CASE Statement

If you don't handle all choices in the **CASE** statement it actually gives you an error. So you won't accidentally get the wrong circuit.

```
-- Assume SEL is 2-bits
PROC: process (A, B, C, SEL)
begin
  -- Note: Missing "11"
  case SEL is
    when "00" => F <= A;
    when "01" => F <= B;
    when "10" => F <= C;
end process;
```

From Quartus:

Error (10313): VHDL Case Statement error at testtest.vhd(16): Case Statement choices must cover all possible values of expression

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

BEST PRACTICE: Use **when others** to catch all other cases

Summary: 1. Purely Combinational

Process outputs are a function of the current input values

Rule 1A:

- Every input to the process must be in the sensitivity list or use keyword **all** in VHDL 2008

Assignment Project Exam Help

Rule 1B:

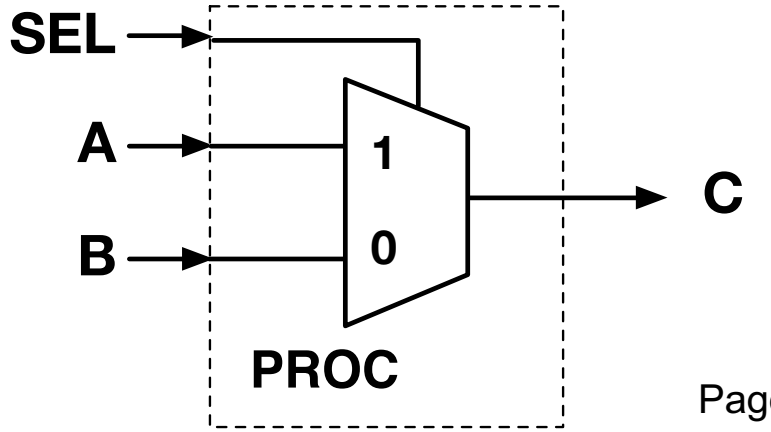
- Every output must be assigned a value for every possible combination of inputs.

https://tutorcs.com

- In other words, every output must be assigned a value for every possible path through the process' description

WeChat: cstutorcs

```
PROC: process (A, B, SEL)
begin
    if (SEL = '1') then
      C <= A;
    else
      C <= B;
    end if;
end process;
```



Pattern 2: Sequential

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

2. Sequential

Each output changes ONLY on the rising or falling edge of a single clock

Rule 2A:

- Only the clock should be in the sensitivity list

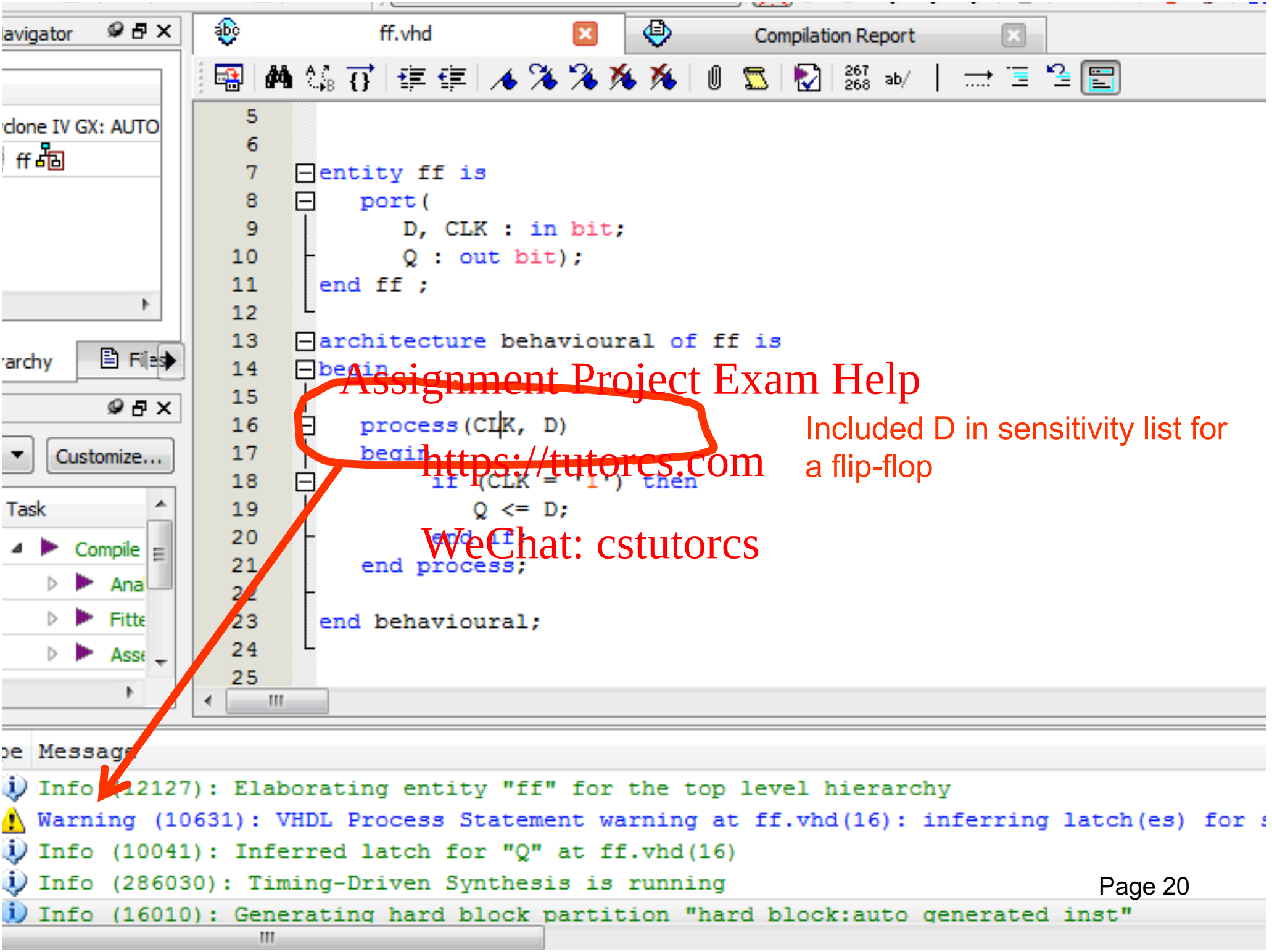
Rule 2B:

- Only signals that change on the **same edge** of the **same clock** should be part of the **same process**

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

```
process (CLK)
begin
  if (rising_edge(CLK)) then
    <logic>
  end if;
end process;
```

Sequential Circuit with Synchronous Reset Falls Under This Category


ff.vhd
Compilation Report
5
6
7 entity ff is
8 port(
9 D, CLK : in bit;
10 Q : out bit);
11 end ff ;
12
13 architecture behavioural of ff is
14 begin
15
16 process(CLK, D)
17 begin
18 if (CLK = '1') then
19 Q <= D;
20 end if;
21 end process;
22
23 end behavioural;
24
25
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs
Included D in sensitivity list for a flip-flop
Message
Info (12127): Elaborating entity "ff" for the top level hierarchy
Warning (10631): VHDL Process Statement warning at ff.vhd(16): inferring latch(es) for s
Info (10041): Inferred latch for "Q" at ff.vhd(16)
Info (286030): Timing-Driven Synthesis is running
Info (16010): Generating hard block partition "hard block:auto generated inst"

Flip Flop Reset Signals

A flip-flop can have either a synchronous or asynchronous reset

Asynchronous Reset: When the reset signal is high, the flip-flop is reset (forced to '0') immediately, regardless of the clock.

Assignment Project Exam Help

Synchronous Reset: On a rising clock edge, if the reset signal is high, The flip-flop is reset (forced to '0').

https://tutorcs.com

WeChat: cstutorcs

The difference

Synchronous reset → flip-flop only resets on a rising clock edge

Asynchronous reset → flip-flop resets immediately

Recall rule for Pattern 2: Each output changes ONLY on the rising or falling edge of a single clock

Do either of these match this pattern?

Assignment Project Exam Help

Asynchronous Reset: When the reset signal is high, the flip-flop is reset (forced to '0') immediately, regardless of the clock.

https://tutorcs.com

WeChat: cstutorcs

Synchronous Reset: On a rising clock edge, if the reset signal is high, The flip-flop is reset (forced to '0').

Describing DFF with Synchronous Reset

Synchronous Reset: On a rising clock edge, if the reset signal is high, The flip-flop is reset (forced to '0').

```
architecture ARCH of DFF is
begin
  process (CLK)
  begin
    if (rising_edge(CLK)) then
      if (RESET = '1') then
        Q <= '0';
      else
        Q <= D;
      end if;
    end if;
  end process;
end architecture;
```

Reset Case

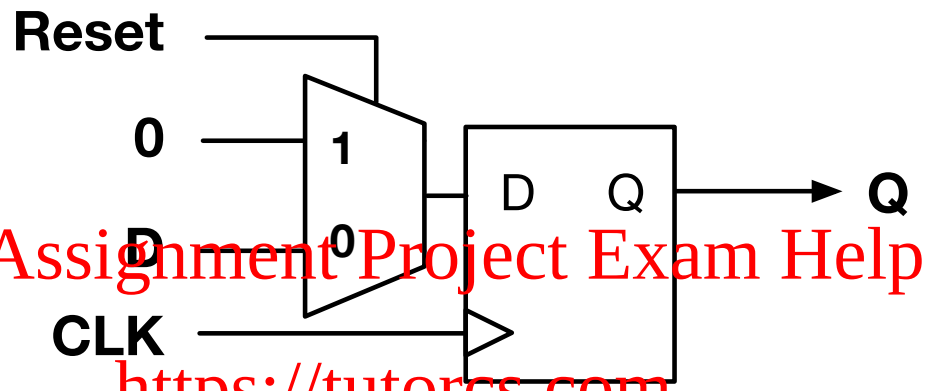
Normal Operation

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Reset behaviour described **inside** rising clock edge case

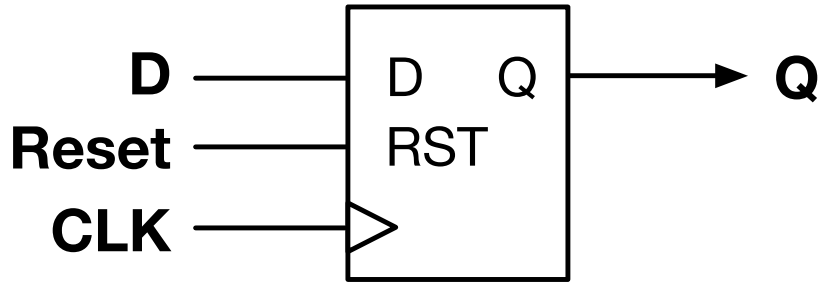
DFF With Synchronous Reset

Behaviour is equivalent to:



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

But there are better transistor level implementations of such functionality and we just assume that it's possible to create such a flip flop with synchronous reset input



Asynchronous Reset: When the reset signal is high, the flip-flop is reset (forced to '0') immediately, regardless of the clock.

?

Assignment Project Exam Help

https://tutorcs.com

Synchronous Reset: On a rising clock edge, if the reset signal is high, The flip-flop is reset (forced to '0').

WeChat: cstutorcs

✔ **Pattern 2**

Pattern 3, Sequential with Asynchronous Reset

Assignment Project Exam Help
https://tutorcs.com

WeChat: cstutorcs

3. Sequential with Asynchronous Reset

Reset occurs immediately

Rule 3A:

- Sensitivity list includes clock and reset

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

```
process (CLK, RESET)
  begin
    if (RESET = '1') then
      <reset assignments>
    elsif (rising_edge(CLK)) then
      <normal operation/logic>
    end if;
end process;
```

3. Sequential with Asynchronous Reset

Rule 3B:

- Assignments in the reset part can only reference constants or literals

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

```
process (CLK, RESET)
  begin
    if ( RESET = '1' ) then
      A <= '0';              -- YES
      B <= ANOTHER_SIG;      -- NO
    elsif ( rising_edge(CLK) ) then
...
```

Describing DFF with Asynchronous Reset

Asynchronous Reset: When the reset signal is high, the flip-flop is reset (forced to '0') immediately, regardless of the clock.

```
architecture ARCH of DFF is
begin
  process (CLK, RESET)
  begin
    if (RESET = '1') then
      Q <= '0';
    elsif (rising_edge(CLK)) then
      Q <= D;
    end if;
  end process;
end architecture;
```

RESET now in sensitivity list

Reset Case

Normal Operation

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Reset behaviour described **before** rising clock edge case

Final Rule (the most important rule of all):

If you want to synthesize your circuit, *every* process must fall *exactly* into one of these categories. Every process. Every single one. No exceptions.

Assignment Project Exam Help

If one of your processes doesn't, you need to break it up into blocks, where each block does fit into one of these categories.

https://tutorcs.com

WeChat: cstutorcs

(note: I am being a little bit conservative here… some synthesizers will handle a few patterns not described here. But don't count on it).

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Summary: Synthesizable Processes

Make sure all of your processes fall under one of these three patterns

If one of them doesn't then break it into multiple processes that do!

Assignment Project Exam Help

I'm being conservative. Some synthesis tools can handle more complex patterns. But don't count on it

https://tutorcs.com

WeChat: cstutorcs

TIP: Anytime you write a process, ask your self "Does this process fall under one of these three patterns?"