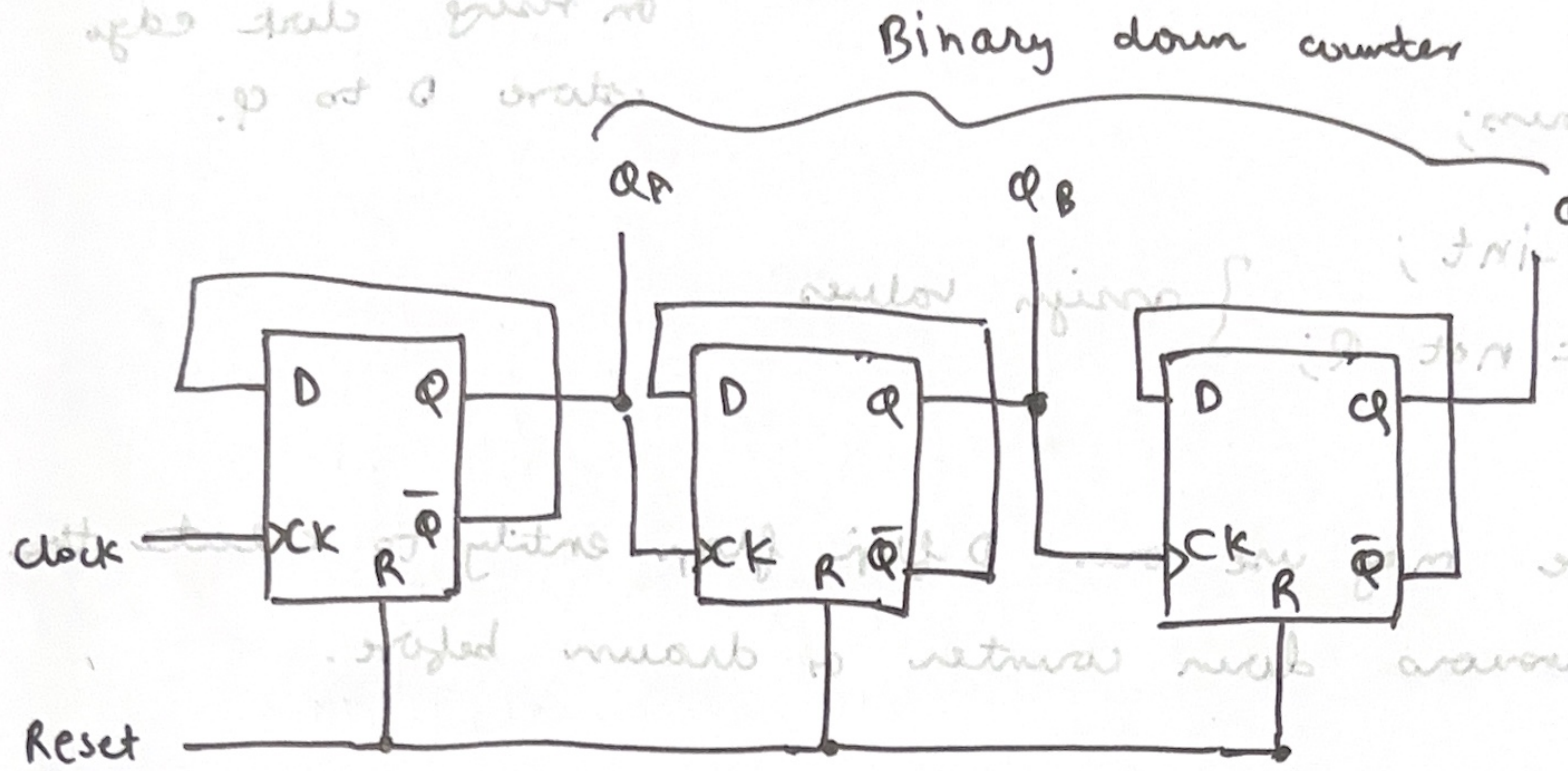
lab 6: 25 sep 2025

## Binary Asynchronous down counter:



i) on RESET counter start from '111'.

ii) RESET is asynchronous in nature

Implementation of D flip flop:
Done through behavioral modelling:

ports:

| | | |
|---|---|---|
| clk → clock signal | } | in std_logic |
| set → set q as 1 | | |
| reset → set q as 0 | | |
| D → input of D ~~Kat~~ flip flop for data | | |
| q → storage | } | out std_logic |
| qbar; $\bar{q}$ → storage inverted | | |

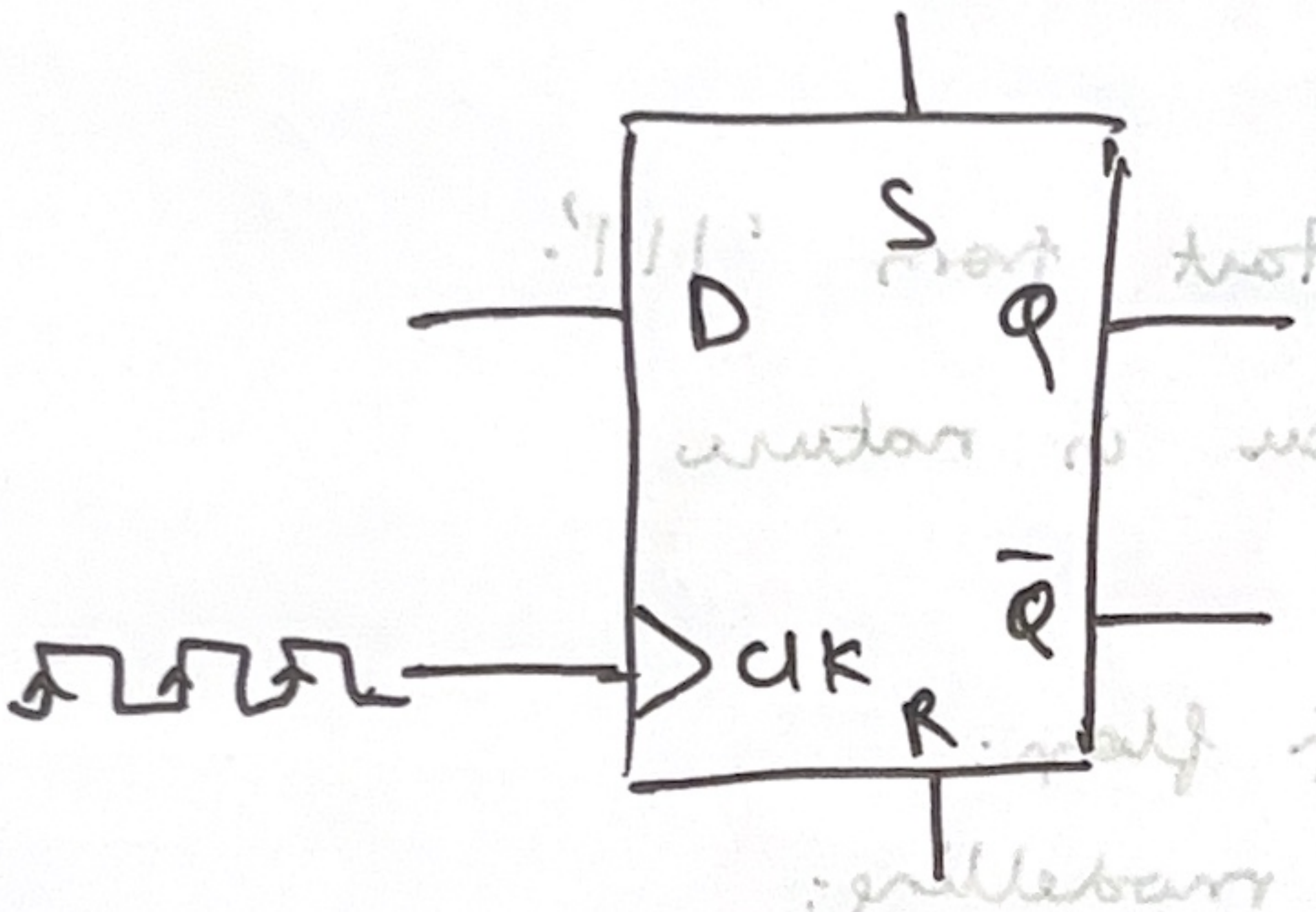
Logic of D flip flop:

```
process (clk, reset, set)  → whenever one of these signals
begin                           changes, block is triggered
  if (reset = '1') then
      q_int <= '1'          } Basic reset & set
  elsif (set = '1') then      active high functions
      q_int <= '1'
```

```
elsif rising_edge(clk) then
        ~~[illegible]~~ Q_int <= D
```
} if set & reset are not triggered then, on rising clock edge store D to Q.

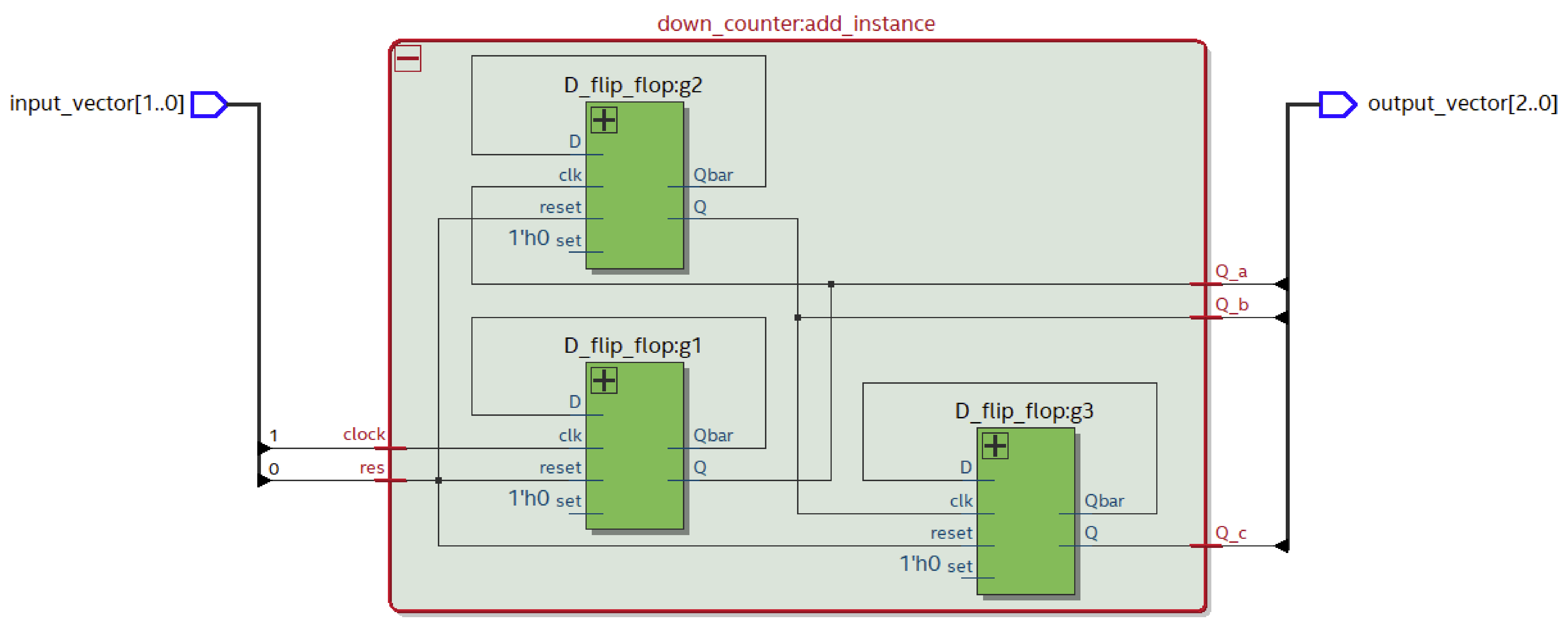
```
end if;
end process;

Q <= Q_int;
Qbar <= not Q;
```
} assign values

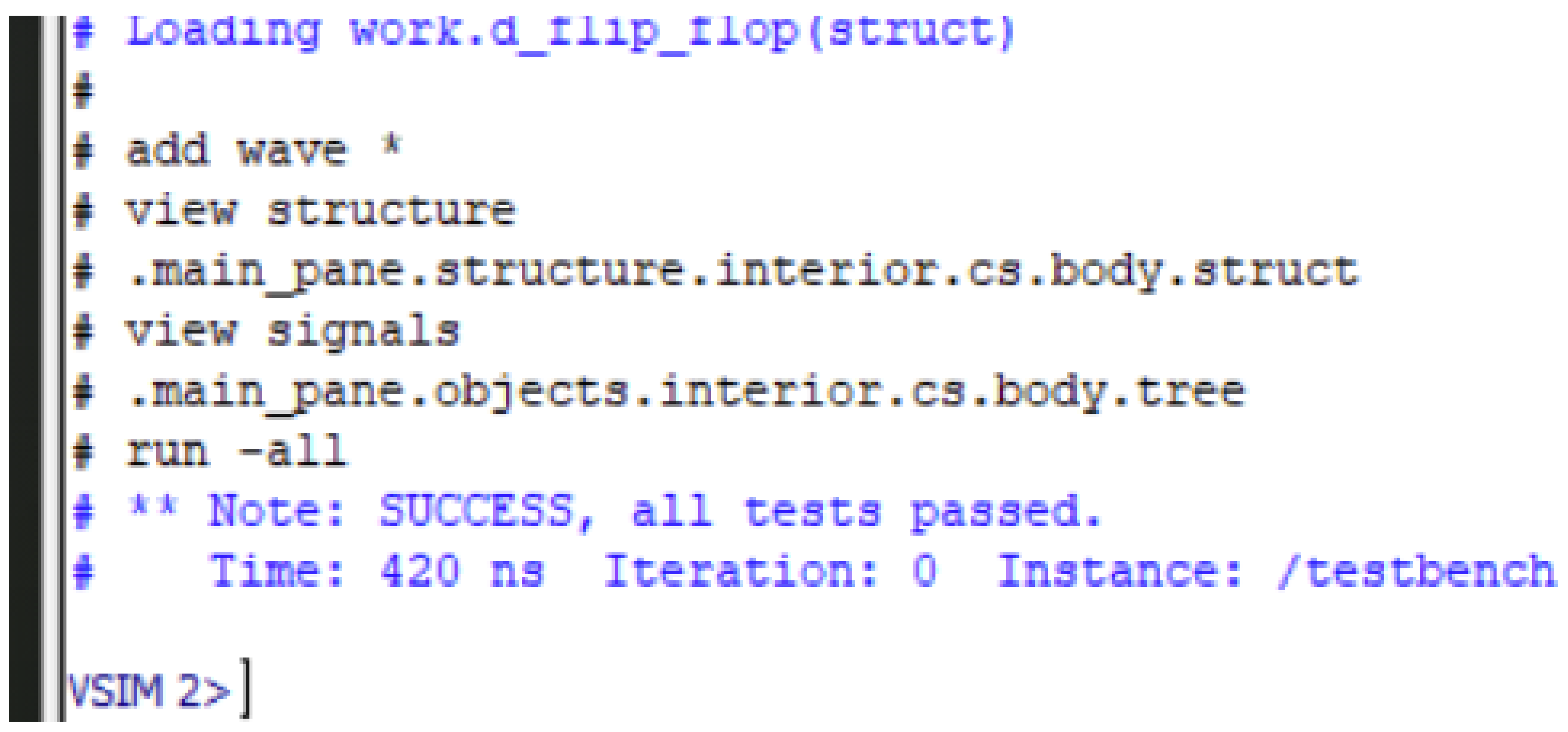
Now we may use our D flip flop entity to create the Asynchronous down counter as drawn before.



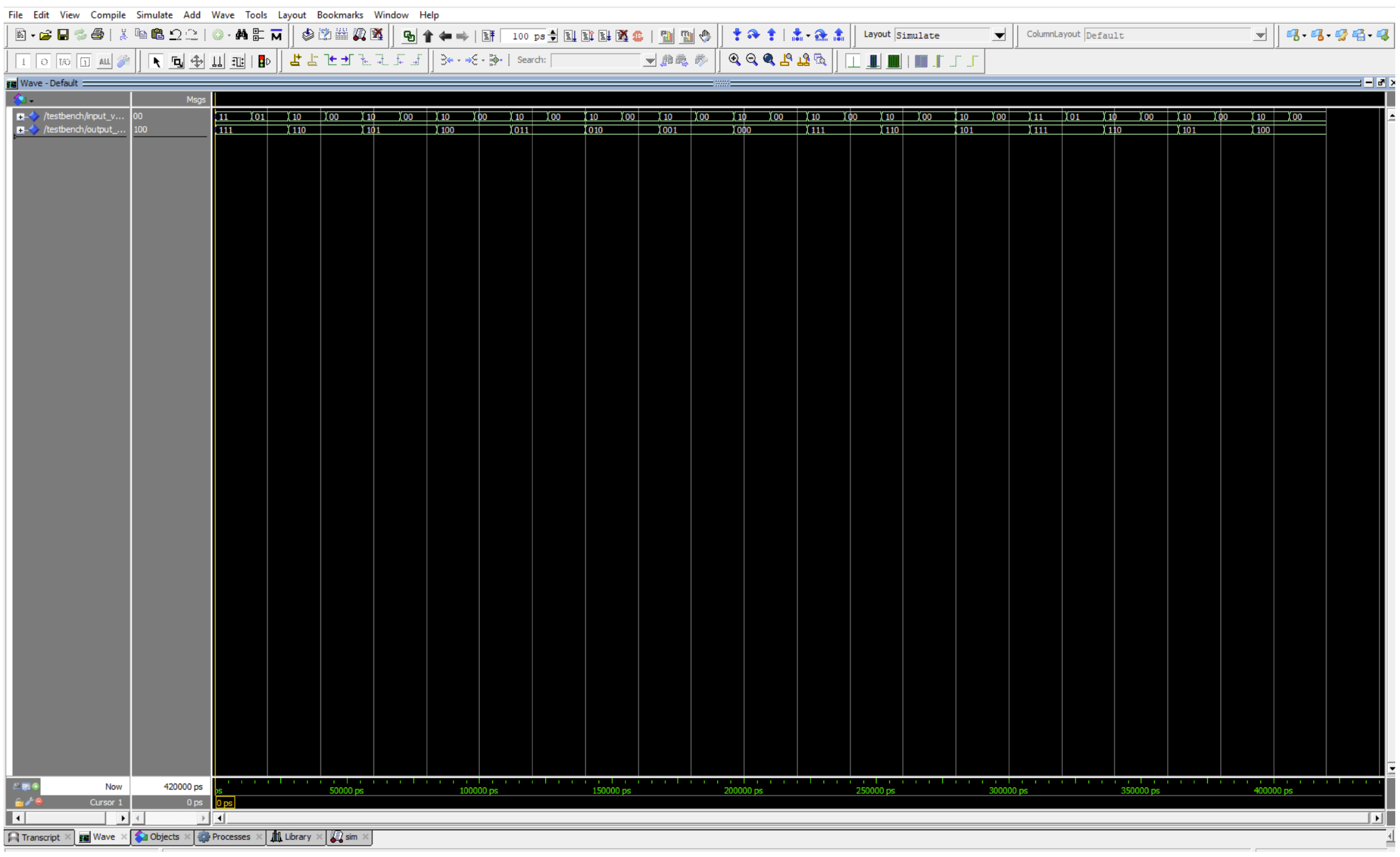
The output counter bits are $\langle Q_2\ Q_1\ Q_0 \rangle$

[illegible]
25/09/2025





```
# Loading work.d_flip_flop(struct)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 420 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```



File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
100 ps
Search:
Layout Simulate
ColumnLayout Default
Wave - Default
Msgs
/testbench/input_v... 00
/testbench/output_... 100
11 01 10 00 10 00 10 00 10 00 10 00 10 00 10 00 10 00 10 00 10 00 11 01 10 00 10 00 10 00
111 110 101 100 011 010 001 000 111 110 101 111 110 101 100
Now 420000 ps
Cursor 1 0 ps
50000 ps
100000 ps
150000 ps
200000 ps
250000 ps
300000 ps
350000 ps
400000 ps
0 ps
Transcript
Wave
Objects
Processes
Library
sim