**Lab Task 1:**

**Implement the FSM on FPGA**

**VHDL Code:**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** FSMdiv **is**

**port** **(**

clk **:** **in** std\_logic**;**

letstart **:** **in** std\_logic**;**

remaind **:** **out** std\_logic\_vector**(**3 **downto** 0**);**

quot **:** **out** std\_logic\_vector**(**3 **downto** 0**)**

**);**

**end** FSMdiv**;**

**architecture** behav **of** FSMdiv **is**

**signal** x **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** "1000"**;** -- 8

**signal** y **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** "0011"**;** -- 3

**signal** q **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** "0000"**;** -- For Quotient

--define state using one hot coding

**constant** initial **:** std\_logic\_vector**(**2 **downto** 0**)** **:=** "001"**;**

**constant** compute **:** std\_logic\_vector**(**2 **downto** 0**)** **:=** "010"**;**

**constant** done **:** std\_logic\_vector**(**2 **downto** 0**)** **:=** "100"**;**

-- State Memory

**signal** mystate **:** std\_logic\_vector**(**2 **downto** 0**)** **:=** initial**;**

**begin**

**process** **(**clk**)**

**begin**

**if** **(**clk'**event** **and** clk **=** '1'**)** **then**

**case** mystate **is**

------

**when** initial **=>**

-- Initialization of signals

x **<=** "1000"**;** -- 8

y **<=** "0011"**;** -- 3

q **<=** "0000"**;** -- Reset quotient

**if** **(**letstart **=** '1'**)** **then**

mystate **<=** compute**;**

**else**

mystate **<=** initial**;**

**end** **if;**

------

**when** compute **=>**

-- Perform subtraction and increment quotient

**if** **(**x **>=** y**)** **then**

x **<=** x **-** y**;**

q **<=** q **+** 1**;**

**end** **if;**

-- Check if we can continue subtracting

**if** **(**x **>=** y**)** **then**

mystate **<=** compute**;**

**else**

mystate **<=** done**;**

**end** **if;**

------

**when** done **=>**

-- Output remainder and quotient

remaind **<=** x**;**

quot **<=** q**;**

-- Go back to initial state

mystate **<=** initial**;**

------

**when** **others** **=>**

mystate **<=** initial**;**

------

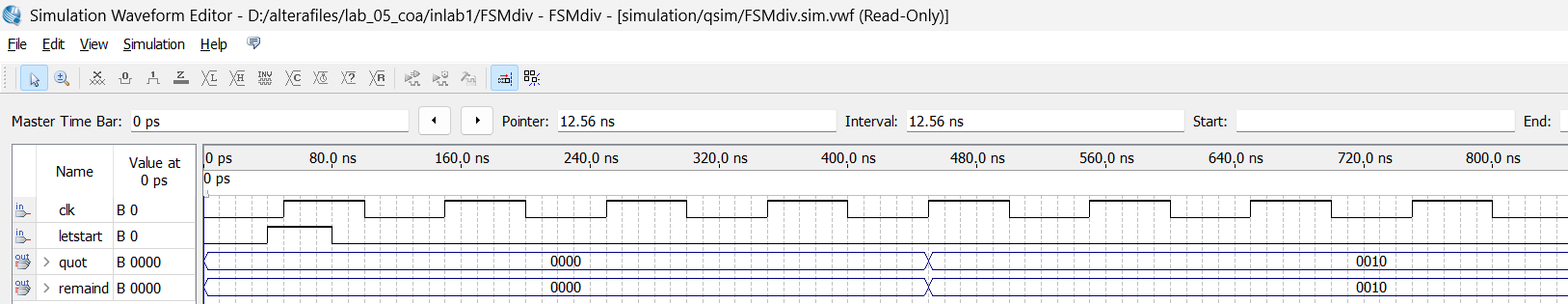
**end** **case;**

**end** **if;**

**end** **process;**

**end** behav**;**

**Simulation:**



Simulation 1: Compute quotient and remainder after enabling positive edge trigger. The output is generated in 3 cycles.

**Lab Task 2:**

**Implement a 4-bit sequence detector(1011)**

**VHDL Code:**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** FSM\_sequence\_detactor **is**

**port** **(**

clk **:** **in** std\_logic**;**

letstart **:** **in** std\_logic**;**

y **:** **out** std\_logic

**);**

**end** FSM\_sequence\_detactor**;**

**architecture** bhv **of** FSM\_sequence\_detactor **is**

--define state using one hot coding

**constant** state0 **:** std\_logic\_vector**(**4 **downto** 0**)** **:=** "00001"**;**

**constant** state1 **:** std\_logic\_vector**(**4 **downto** 0**)** **:=** "00010"**;**

**constant** state2 **:** std\_logic\_vector**(**4 **downto** 0**)** **:=** "00100"**;**

**constant** state3 **:** std\_logic\_vector**(**4 **downto** 0**)** **:=** "01000"**;**

**constant** state4 **:** std\_logic\_vector**(**4 **downto** 0**)** **:=** "10000"**;**

-- State Memory

**signal** mystate **:** std\_logic\_vector**(**4 **downto** 0**)** **:=** state0**;** -- 00001

**begin**

**process** **(**clk**)**

**begin**

**if** **(**clk'**event** **and** clk **=** '1'**)** **then** -- positive edge sensitive

**case** mystate **is**

------

**when** state0 **=>**

y **<=** '0'**;**

**if** **(**letstart **=** '1'**)** **then**

mystate **<=** state1**;**

**else** -- 0

mystate **<=** state0**;**

**end** **if;**

------

**when** state1 **=>**

y **<=** '0'**;**

**if** **(**letstart **=** '0'**)** **then**

mystate **<=** state2**;**

**else** -- 1

mystate **<=** state1**;**

**end** **if;**

------

**when** state2 **=>**

y **<=** '0'**;**

**if** **(**letstart **=** '1'**)** **then**

mystate **<=** state3**;**

**else** --0

mystate **<=** state0**;**

**end** **if;**

------

**when** state3 **=>**

y **<=** '1'**;**

**if** **(**letstart **=** '1'**)** **then**

mystate **<=** state4**;**

**else** -- 0

mystate **<=** state2**;**

**end** **if;**

------

**when** state4 **=>**

y **<=** '1'**;**

**if** **(**letstart **=** '1'**)** **then**

mystate **<=** state0**;**

**else** -- 0

mystate **<=** state0**;**

**end** **if;**

------

**when** **others** **=>**

mystate **<=** state0**;**

------

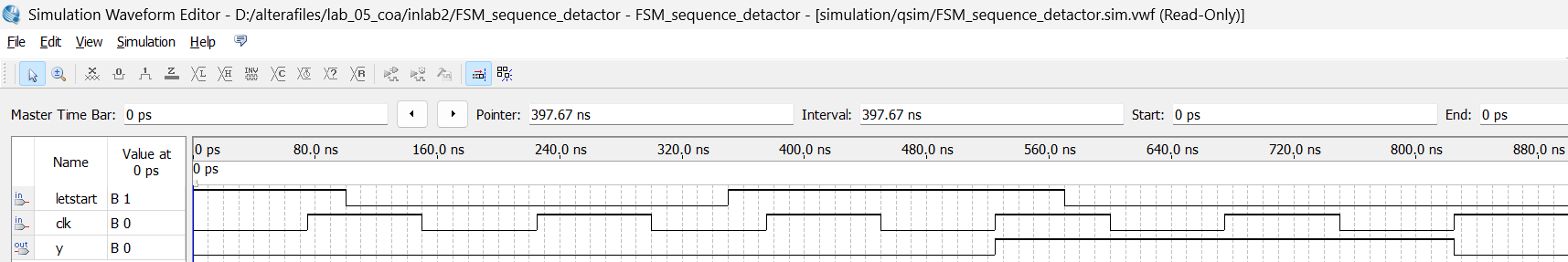
**end** **case;**

**end** **if;**

**end** **process;**

**end** bhv**;**

**Simulation:**



Simulation 2: detects the input(let start) pattern of 1011