Lab 1A

VHDL Code Including Sub Components

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- top entity synchronous Fibonacci generator
-- start with a high signal reset input to reset system.
-- input of high signal to count increments to next value in memory.
-- values in memory are Fibonacci numbers in ascending order.
-- inputs count and reset are de-bounced within a parametrizable debouncer.
-- the num of clk cycles the input signal has to be active for a debouncer
-- output signal change is the Count_Value number.

entity Fibonacci_Gen_Val_1st_14_Val is
GENERIC ( Count_Value : natural := 50000000); -- 50 million
    Port ( clk : in STD_LOGIC;
           count : in STD_LOGIC;
           rst : in STD_LOGIC;
           Fib_val : out STD_LOGIC_VECTOR (7 downto 0));
end Fibonacci_Gen_Val_1st_14_Val;

architecture Behavioral of Fibonacci_Gen_Val_1st_14_Val is

-- debounced signals
signal int_debounced_rst_sig : std_logic;
signal int_debounced_en_sig : std_logic;

-- counter value to select memory location
signal int_counter_and_address_val: UNSIGNED (3 downto 0);

begin

COUNT_0_13: entity work.Counter_0_to_13
port map (
 clk => clk,
 rst => int_debounced_rst_sig,  --debounced reset signal
  en => int_debounced_en_sig,   --debounced enable signal

--tens unit 4 bit output to decoder


Address_Value_count => int_counter_and_address_val);

--Parameterizable debouncer for input count
 Enable_Debounce: entity work.Parameterizable_Debouncer
 Generic map( Count_Value => Count_Value)
 port map (
 clk => clk,
 Sig => count,
 Debounced_Signal => int_debounced_en_sig);
```

```
--Parameterizable debouncer for input reset
 Reset_Debounce: entity work.Parameterizable_Debouncer
 Generic map( Count_Value => Count_Value)
 port map (
 clk => clk,
 Sig => rst ,
 Debounced_Signal => int_debounced_rst_sig );

-- Read Only Memory Asynchronous
Async_ROM_Fib_val_adr_0_to_13 : entity work.Async_ROM14x8
port map (
 Address => int_counter_and_address_val ,
 DataOut =>  Fib_val);

end Behavioral;
```

Figure 2 Top Entity Fibonacci Generator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
-- Synchronous counter values 0-13
-- reset output = 0
-- enable high signal to allow counter to increment at rising edge of
clock
-- counter value rolls over from 13 to 0
entity Counter_0_to_13 is
    Port ( clk : in STD_LOGIC;
           rst : in STD_LOGIC;
           en  : in STD_LOGIC;
           Address_Value_count : out UNSIGNED (3 downto 0));
end Counter_0_to_13;

architecture Behavioral of Counter_0_to_13 is

signal Int_Address_Value : UNSIGNED(3 downto 0);

begin

CNT_SYNC_0_TO_13 : process(clk)
 begin
 if(rising_edge(clk)) then
    if (rst= '1') then
        Int_Address_Value <=(others => '0');
    else
        if (en = '1') then
           if(Int_Address_Value = 13) then
               Int_Address_Value <=(others => '0');
           else
                Int_Address_Value <= (Int_Address_Value +1);
            end if;
       end if;
    end if;
 end if;
 end process CNT_SYNC_0_TO_13;
 Address_Value_count <= Int_Address_Value;

end Behavioral;
```

Figure 1 Synchronous Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.DigEng.all;

-- Synchronous signal Parameterizable debouncer
--output signal high when "Count_Value -1" number of clk signals at input has been high
-- implements Synchronous counter(reset and enable only),  -
-- Synchronous flip flop (no enable, no reset)
-- implements Asynchronousa comparator, 2 AND gates and 3 Inverters.

entity Parameterizable_Debouncer is
   GENERIC ( Count_Value : natural := 50000000); -- 50 million

    Port ( clk : in STD_LOGIC;
           Sig : in STD_LOGIC;
           Debounced_Signal : out STD_LOGIC);
end Parameterizable_Debouncer;

architecture Behavioral of Parameterizable_Debouncer is
-- connects input Sig to inverter and and gate
signal Int_Input_Sig           : STD_LOGIC;
-- connects inverted input Sig to reset input of parameterizable counter
signal Int_Invert_Input_Sig    : STD_LOGIC;
-- connects inverted comparator result AND input signal to enable
-- signal for parameterizable counter
signal Int_Invert_Comp_and_Sig : STD_LOGIC;
-- comparator result connects to D Flip Flop input, and 2 AND gates (1 signal inverted)
signal Int_Comparator_result   : STD_LOGIC;
-- connects the parameterizable counter output to comparator input
signal Int_Counter_output      : UNSIGNED (((log2(Count_Value))-1) downto 0);
-- connects D Flip Flop to output AND gate (signal inverted on input to AND)
signal Int_DFF_Out             : STD_LOGIC;
-- connects final input signal to entity output Debounced_Signal
signal Int_Debounce_Out        : STD_LOGIC;

begin
Int_Input_Sig           <= Sig;
Int_Invert_Input_Sig    <= not(Int_Input_Sig);
Int_Invert_Comp_and_Sig <= ((not(Int_Comparator_result)) and (Int_Input_Sig));
Int_Debounce_Out        <= ((not(Int_DFF_Out)) and (Int_Comparator_result));
Debounced_Signal        <= Int_Debounce_Out;

-- comparator compares output of counter to generic value input "Count_Value"
Int_Comparator_result <= '1' when Int_Counter_output = (Count_Value-1) else '0';

-- D type flip flop holds output value for 1 clk cycle
D_FlipFlop: entity work.D_type_FF
port map (
clk => clk,
D =>  Int_Comparator_result,
Q => Int_DFF_Out);

-- parameterizable counter, counts up to value : "Count_Value" - 1
Counter: entity work.Param_Count_to_N
Generic map( Count_Value => Count_Value)
port map (
 clk => clk,
 Rst => Int_Invert_Input_Sig,-- inverted input"Signal"
 EN  => Int_Invert_Comp_and_Sig,-- "Sig" & (!output of param counter -1)
 Current_Num => Int_Counter_output);-- current counter value

end Behavioral;
```

Figure 3 Parameterizable Debouncer

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- D Type Flip Flop
-- no enable
-- no reset

entity D_type_FF is
    Port (
            -- Inputs
            clk : in STD_LOGIC;
            D   : in STD_LOGIC;
            -- Output
            Q   : out STD_LOGIC);
end D_type_FF;

architecture Behavioral of D_type_FF is

begin
 process(clk)
   begin
     if(rising_edge(clk)) then
          if(clk='1') then
                 Q <= D;
             End if;
         End if;
   end process;
end Behavioral;
```

Figure 4 D type Flip Flop (No Reset or Enable)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Asynchronous ROM
-- memory addresses hold the first 13 number within the
-- fibonacci sequence in ascending order.

entity Async_ROM14x8 is
    Port ( Address : in UNSIGNED (3 downto 0);
           DataOut : out STD_LOGIC_VECTOR (7 downto 0));
end Async_ROM14x8;

architecture Behavioral of Async_ROM14x8 is

type ROM_Array is array (0 to 15) of std_logic_vector(7 downto 0);
 constant Content: ROM_Array := (
 0  => B"00000000", --dec value = 0
 1  => B"00000001", --dec value = 1
 2  => B"00000001", --dec value = 1
 3  => B"00000010", --dec value = 2
 4  => B"00000011", --dec value = 3
 5  => B"00000101", --dec value = 5
 6  => B"00001000", --dec value = 8
 7  => B"00001101", --dec value = 13
 8  => B"00010101", --dec value = 21
 9  => B"00100010", --dec value = 34
 10 => B"00110111", --dec value = 55
 11 => B"01011001", --dec value = 89
 12 => B"10010000", --dec value = 144
 13 => B"11101001", --dec value = 233
 others => (others => '0')); -- ALL OTHER VALUES ARE OUTPUT ZERO
begin
DataOut <= Content(to_integer(Address));
end Behavioral;
```

Figure 5 Asynchronous ROM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.DigEng.all;

--Parameterizable Synchronous Counter
-- reset sets value back to 0 at rising clk edge
-- current count value increments by 1 when enable high and rising edge
of clk
-- when "Count_Value" - 1 has been reached the counter rolls back over
to 0
entity Param_Count_to_N is
   GENERIC ( Count_Value : natural := 50000000); -- 50 million

    Port ( Clk : in STD_LOGIC;
           Rst : in STD_LOGIC;
           En : in STD_LOGIC;
           Current_Num : out UNSIGNED (((log2(Count_Value)) - 1) downto
0));
end Param_Count_to_N;

architecture Behavioral of Param_Count_to_N is

signal Int_Current_Value : UNSIGNED(((log2(Count_Value)) - 1)downto 0);

begin

Sync_Param_Counter : process(clk)
 begin
 if(rising_edge(clk)) then
    if (Rst= '1') then
        Int_Current_Value <=(others => '0');
    else
        if (En = '1') then
           if( Int_Current_Value = Count_Value-1 ) then
               Int_Current_Value <=(others => '0');
           else
                Int_Current_Value <= (Int_Current_Value +1);
            end if;
       end if;
    end if;
 end if;
 end process Sync_Param_Counter;
 Current_Num <= Int_Current_Value;

end Behavioral;
```

Figure 6 Parameterizable Synchronous Counter

VHDL Test Bench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- testing procedure
-- full system reset to aquire stable known values
-- tests correct functionality of both debouncers
-- input signals for debouncers must be help for the value
-- of "Count_Value" - 1 clk cycles before the debouncer output
-- will change. both debouncers will be tested for less than
-- the minimum clk cycles and above the minumum clk cycles
-- to prove predicted functionality.
-- Fibonacci sequencer will be tesetd via incrementing values
-- then resetting the system and incrementing again until
-- sequencer value rolls back over to zero.

entity Fibonacci_Gen_Val_1st_14_VAl_tb is

end Fibonacci_Gen_Val_1st_14_VAl_tb;

architecture Behavioral of Fibonacci_Gen_Val_1st_14_VAl_tb is

-- Constants
-- debouncer counter value
constant Count_Value  : natural := 5;
constant clk_period : time := 10ns;

--input signals
signal clk : STD_LOGIC;
signal rst : STD_LOGIC;
signal count  : STD_LOGIC;

--output signals
signal Fib_val : STD_LOGIC_VECTOR (7 downto 0);


begin

UUT: entity work.Fibonacci_Gen_Val_1st_14_val
-- maps counter value to the debouncer counters
GENERIC MAP( Count_Value => Count_Value)

    PORT MAP (clk  => clk ,
              rst => rst,
              Count  => Count,
              Fib_val => Fib_val);

-- Clock process
clk_process :process
begin
 clk <= '0';
 wait for clk_period/2;
 clk <= '1';
 wait for clk_period/2;
end process;

TEST : process

begin
-- wait 100 ns for global reset tot finish set by Xilinx
wait for 100 ns;
```

```
wait until falling_edge(clk);

-- set inputs to known values
 wait for clk_period;
 rst   <= '0';
 count <= '0';
 wait for clk_period;


-- below minimum number of clk cycles needed, debounce output
-- signal will be low
 rst <= '1';
 wait for clk_period*3;
 rst <= '0';
 wait for clk_period*3;

-- above minimum number of clk cycles needed, debounce output
 -- signal will be high
 rst <= '1';
 wait for clk_period*4;
 rst <= '0';
  wait for clk_period*4;
 -- below minimum number of clk cycles needed, debounce output
 -- signal will be low
  count <= '1';
 wait for clk_period*3;
 count <= '0';
 wait for clk_period*3;

-- above minimum number of clk cycles needed, debounce signal
-- will be high
 for i in 0 to 10 loop
         wait for clk_period*4;
         count <= '1';
         wait for clk_period*4;
         count <= '0';
     END LOOP;
-- shows reset debouncer works correctly and shows accurate system reset
wait for clk_period*4;
rst <= '1';
wait for clk_period*4;
rst <= '0';
 -- allows the sequencer to increment the output value for correct functionality
 for i in 0 to 15 loop
                wait for clk_period*4;
                count <= '1';
                wait for clk_period*4;
                count <= '0';
            END LOOP;

 wait; --waits forever
end process;

end Behavioral;
```

Figure 7 Debouncer Testbench

Simulations

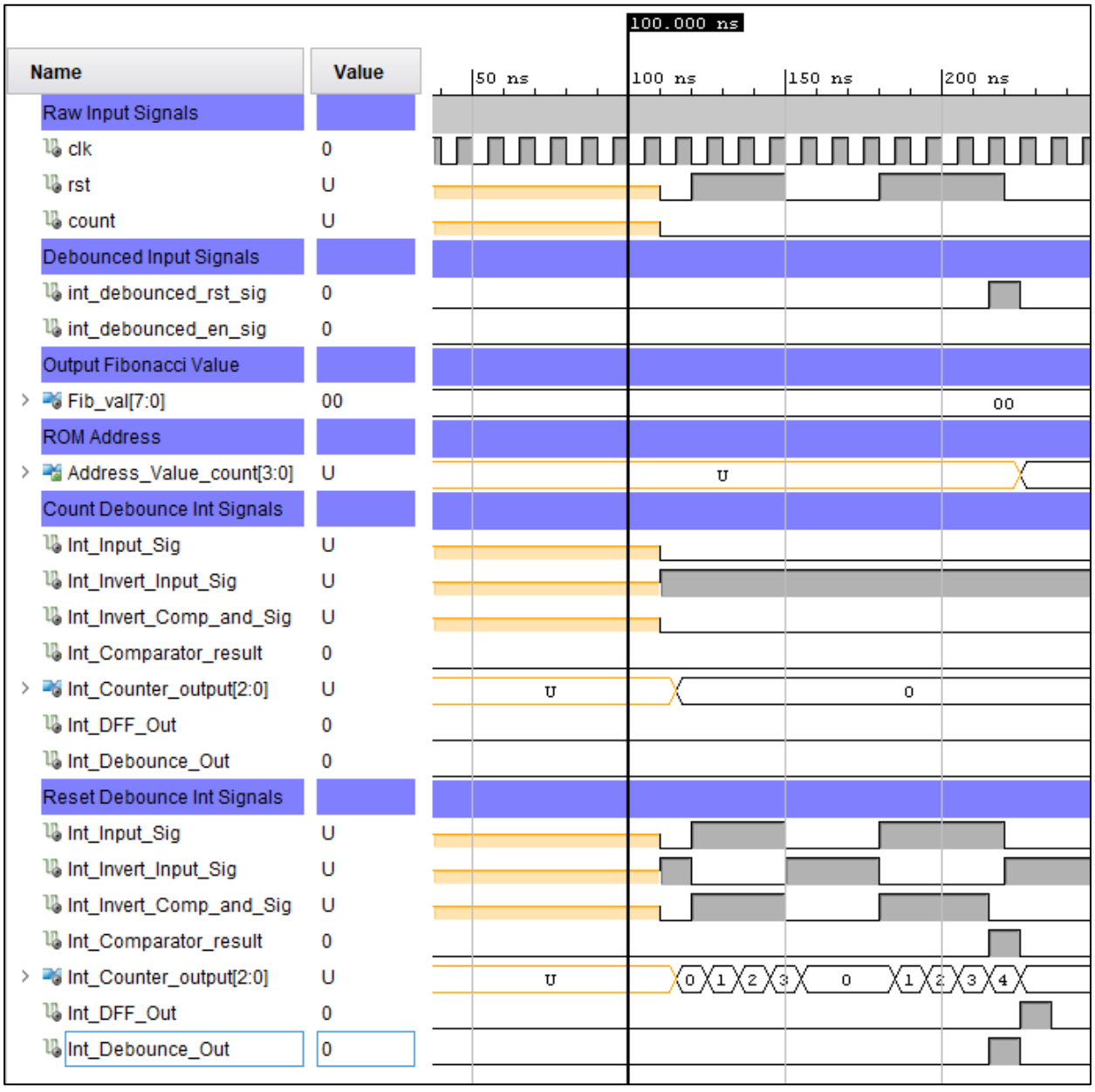

Figure 8 100ns initialisation time

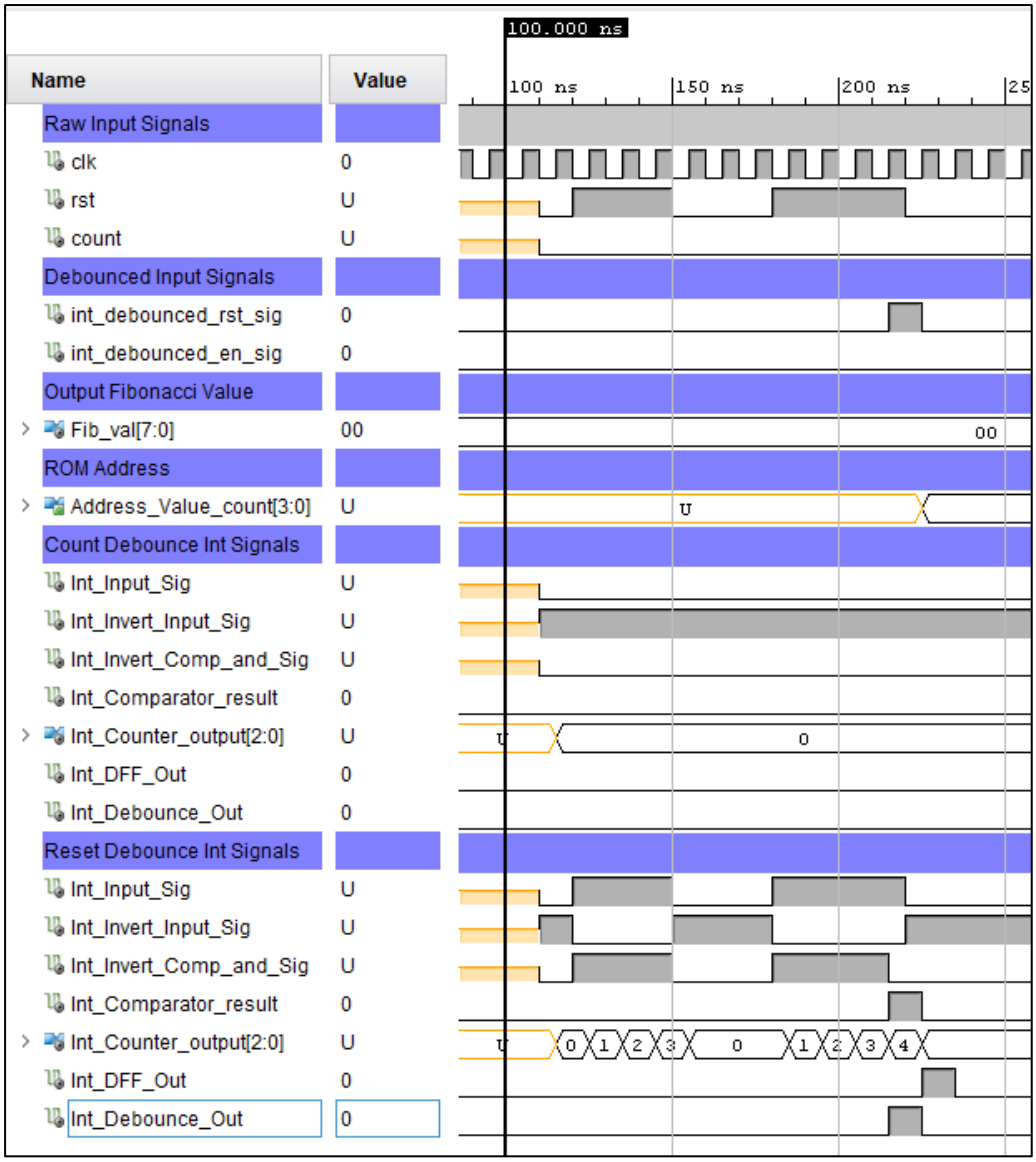

Figure 9 Reset of debouncers to known values & Raw Reset signal only Debounced on the 4th clock cycle count(5)-1

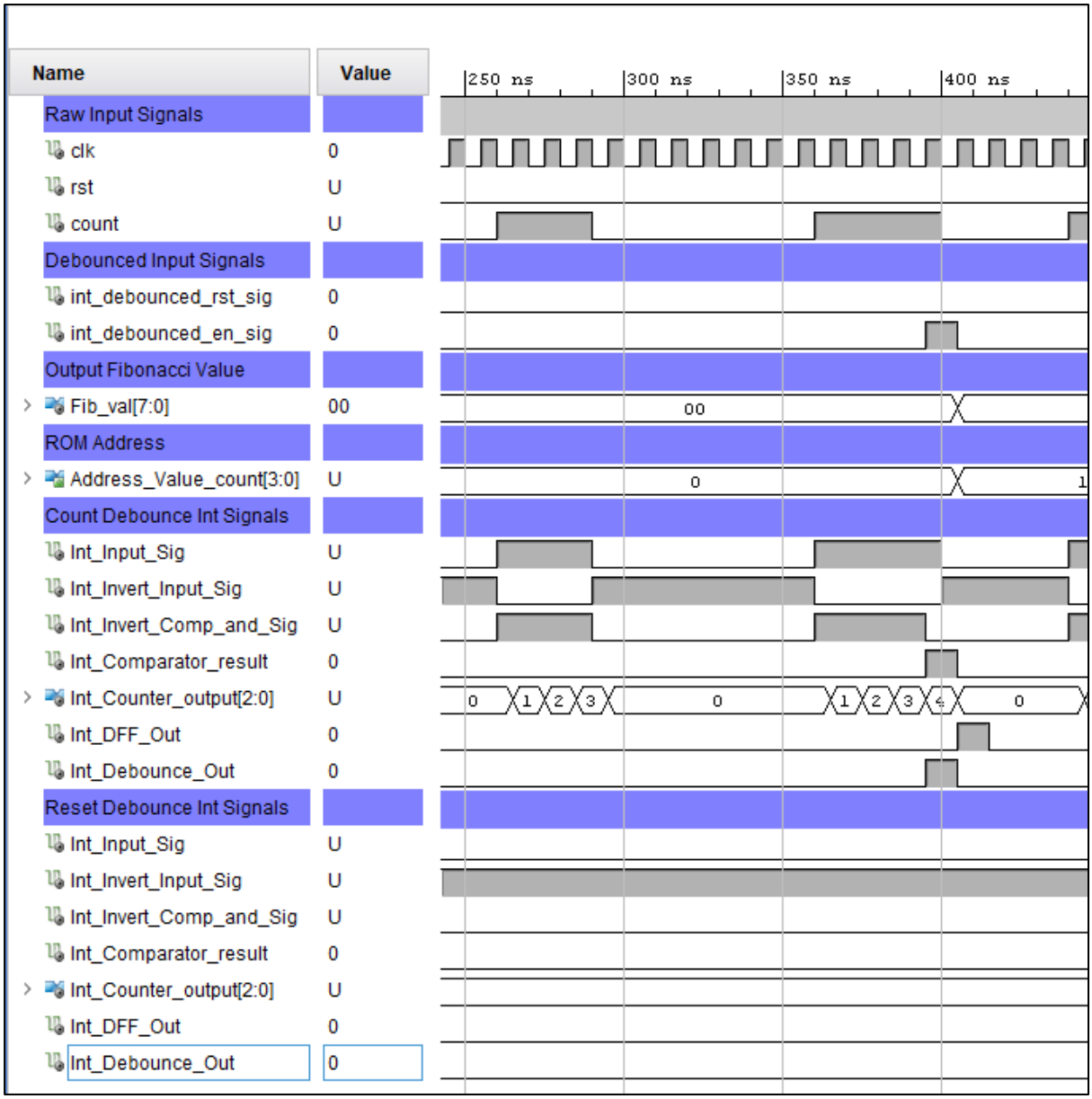

Figure 10 Raw Count signal only Debounced on the 4th clock cycle count(5)-1

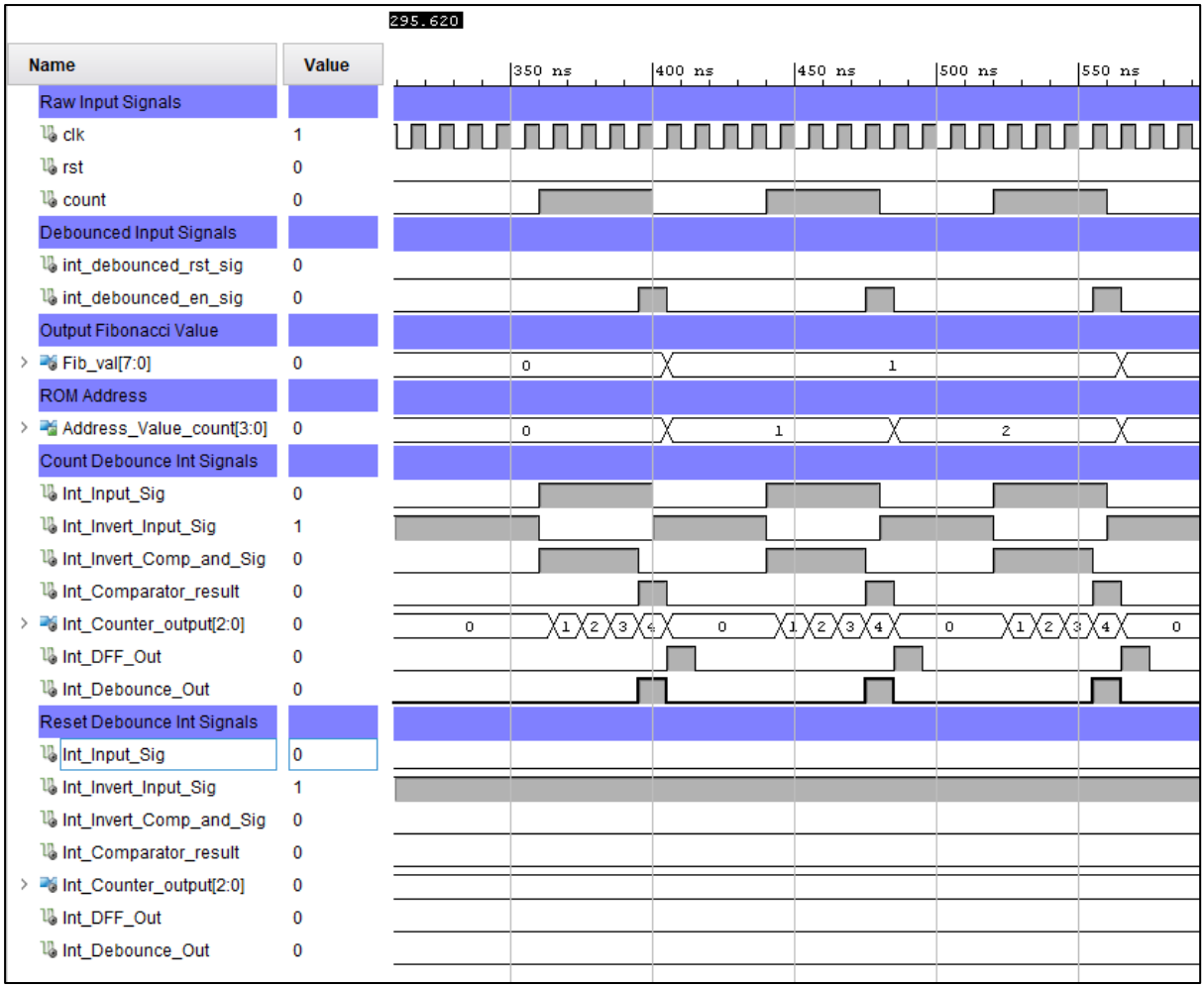

Figure 11 Count debounced signal workin in succesion to increment ROM Address and Fibonacci output value

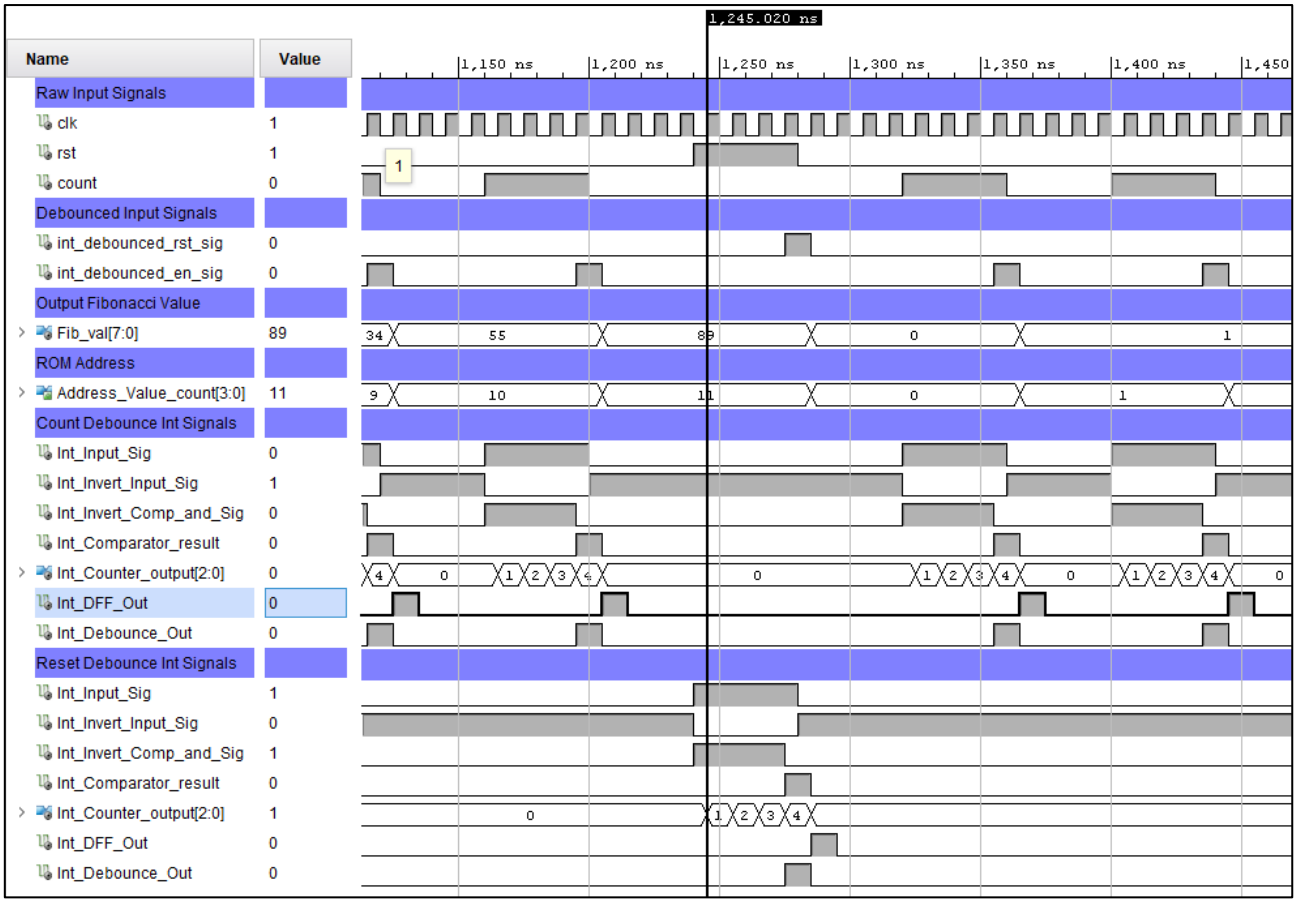

Figure 12 Zoomed Reset of entire system Fibonacci value and ROM Address back to zero

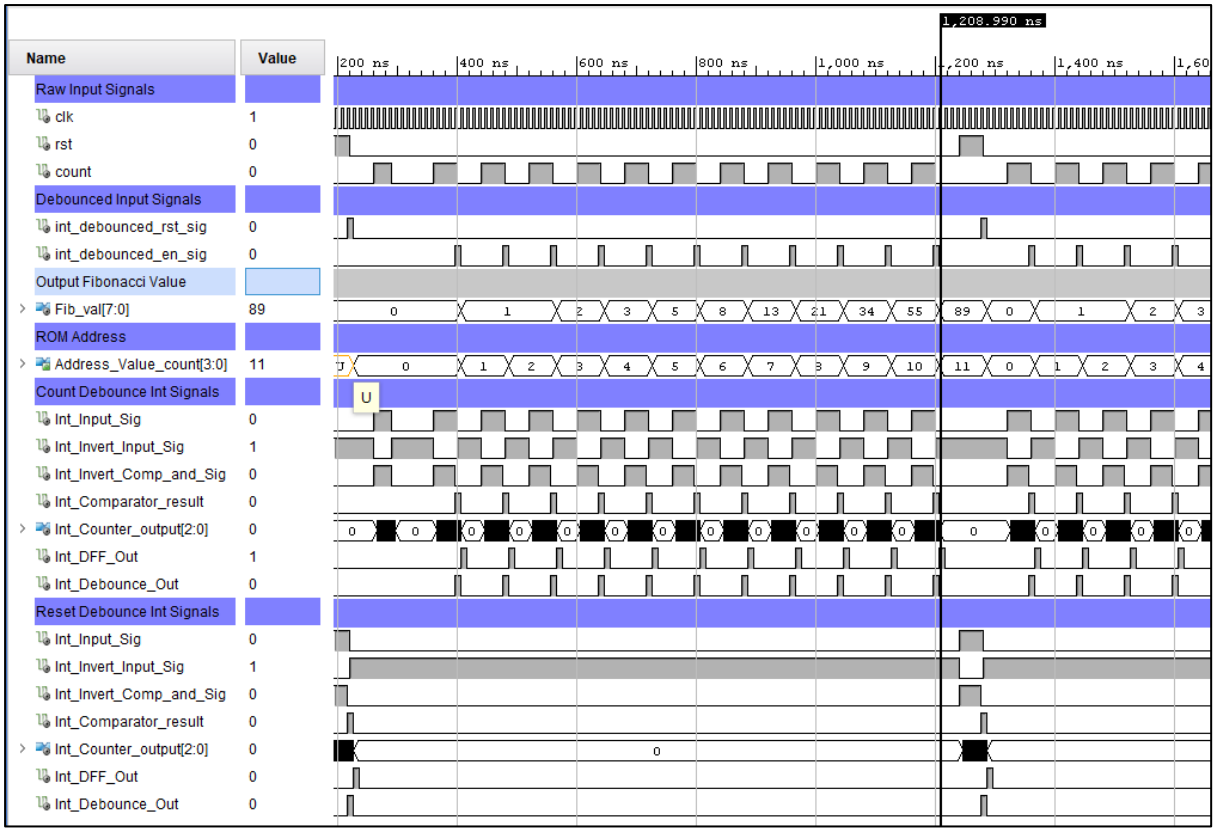

Figure 13 Incrementing ROM Address and Fibonacci output value until system Reset

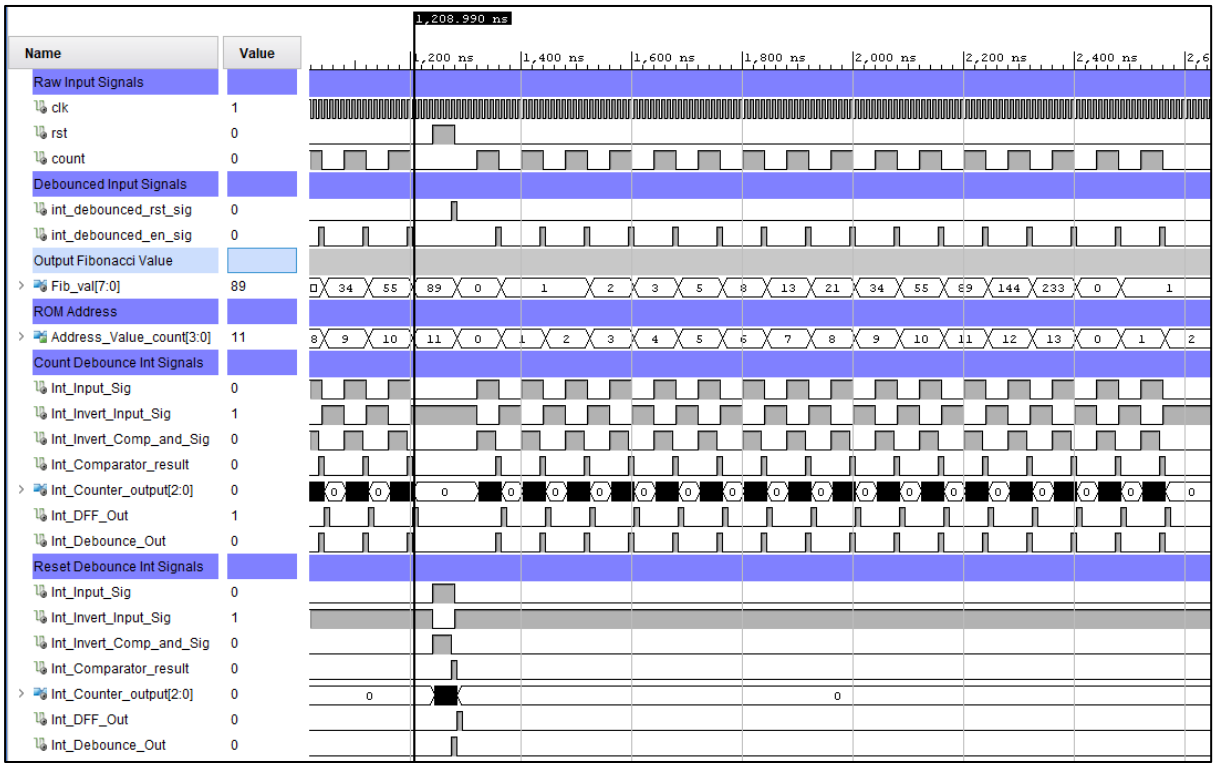

Figure 14 Full run through of system from reset and ROM address roll over back to zero

RTL Component Statistics & RTL Hierarchical Component Statistics

Counter size is 26 bits as expected as you can count up to 50,000,00 using 26 bits

```
---------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
         2 Input     26 Bit       Adders := 2
         2 Input      4 Bit       Adders := 1
+---Registers :
                     26 Bit     Registers := 2
                      4 Bit     Registers := 1
                      1 Bit     Registers := 2
+---Muxes :
         2 Input     26 Bit        Muxes := 2
         2 Input      4 Bit        Muxes := 1
         2 Input      1 Bit        Muxes := 2
---------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Start RTL Hierarchical Component Statistics
---------------------------------------------------------------------------
Hierarchical RTL Component report
Module Counter_0_to_13
Detailed RTL Component Info :
+---Adders :
         2 Input      4 Bit       Adders := 1
+---Registers :
                      4 Bit     Registers := 1
+---Muxes :
         2 Input      4 Bit        Muxes := 1
Module D_type_FF
Detailed RTL Component Info :
+---Registers :
                      1 Bit     Registers := 1
Module Param_Count_to_N
Detailed RTL Component Info :
+---Adders :
         2 Input     26 Bit       Adders := 1
+---Registers :
                     26 Bit     Registers := 1
+---Muxes :
         2 Input     26 Bit        Muxes := 1
Module Parameterizable_Debouncer
Detailed RTL Component Info :
+---Muxes :
         2 Input      1 Bit        Muxes := 1
---------------------------------------------------------------------------
Finished RTL Hierarchical Component Statistics
---------------------------------------------------------------------------
```

Figure 15 Component Statistics

XDC file

```
set_property PACKAGE_PIN T18 [get_ports {count}]
set_property IOSTANDARD LVCMOS18 [get_ports {count}]
set_property PACKAGE_PIN R16 [get_ports {rst}]
set_property IOSTANDARD LVCMOS18 [get_ports {rst}]

set_property PACKAGE_PIN Y9 [get_ports {clk}]
set_property IOSTANDARD LVCMOS18 [get_ports {clk}]

set_property PACKAGE_PIN T22 [get_ports {Fib_val[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[0]}]
set_property PACKAGE_PIN T21 [get_ports {Fib_val[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[1]}]
set_property PACKAGE_PIN U22 [get_ports {Fib_val[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[2]}]
set_property PACKAGE_PIN U21 [get_ports {Fib_val[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[3]}]
set_property PACKAGE_PIN V22 [get_ports {Fib_val[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[4]}]
set_property PACKAGE_PIN W22 [get_ports {Fib_val[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[5]}]
set_property PACKAGE_PIN U19 [get_ports {Fib_val[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[6]}]
set_property PACKAGE_PIN U14 [get_ports {Fib_val[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports {Fib_val[7]}]
```

Figure 16 XDC File Pin Assignment

Lab 1B

VHDL Code Modified Components

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- top entity synchronous fibonacci generator
-- start with a high signal reset input to reset system.
-- input of high signal to count increments to next value in memory.
-- values in memory are fibonnacci numbers in ascending order.
-- inputs count and reset are debounced within a parameterizable
debouncer.
-- the num of clk cycles the input signal has to be active for a
debouncer -
-- output signal change is the Count_Value number.
-- the set of debouncers that is implmemented is controlled vis the
--" Simulation"  variable, if true the Xilinx debouncers are implemented
-- if false the parameterizable debouncers are implemented.

entity Fibonacci_Gen_Val_1st_14_Val is
GENERIC ( Count_Value : natural := 50000000; -- 50 million
          Simulation : boolean := true);

    Port ( clk : in STD_LOGIC;
           count : in STD_LOGIC;
           rst : in STD_LOGIC;
           Fib_val : out STD_LOGIC_VECTOR (7 downto 0));
end Fibonacci_Gen_Val_1st_14_Val;

architecture Behavioral of Fibonacci_Gen_Val_1st_14_Val is

-- debounced signals
signal int_debounced_rst_sig : std_logic;
signal int_debounced_en_sig : std_logic;

-- counter value to select memory location
signal int_counter_and_address_val: UNSIGNED (3 downto 0);

begin

COUNT_0_13: entity work.Counter_0_to_13
port map (
 clk => clk,
 rst => int_debounced_rst_sig,  --debounced reset signal
  en => int_debounced_en_sig,   --debounced enable signal

--tens unit 4 bit output to decoder
Address_Value_count => int_counter_and_address_val);

-- xilinx debouncer vhdl code
-- if value of SIMULATION is true this entity
-- is used
SIM_CNT_DEBOUNCE: if SIMULATION generate
 Enable_Debounce: entity work.Debouncer
 port map (
 clk => clk,
 sig => count,                      --debounced reset signal
 deb_sig => int_debounced_en_sig); --debounced enable signal
 end generate;
```

```
--Parameterizable debouncer for input enable signal
  -- if value of SIMULATION is false this entity
 -- is used
IMP_CNT_DEBOUNCE: if not SIMULATION generate
Enable_Debounce: entity work.Parameterizable_Debouncer
 Generic map( Count_Value => Count_Value)
 port map (
 clk => clk,
 Sig => count,                                  --debounced reset signal
 Debounced_Signal => int_debounced_en_sig);  --debounced enable signal
 end generate;


 -- xilinx debouncer vhdl code
 -- if value of SIMULATION is true this entity
 -- is used
SIM_RST_DEBOUNCE: if SIMULATION generate
Reset_Debounce: entity work.Debouncer
port map (
clk => clk,
sig => rst ,                                 --debounced reset signal
deb_sig => int_debounced_rst_sig );       --debounced enable signal
end generate;

 --Parameterizable debouncer for input reset
 -- if value of SIMULATION is false this entity
 -- is used
 IMP_RST_DEBOUNCE: if not SIMULATION generate
 Reset_Debounce: entity work.Parameterizable_Debouncer
 Generic map( Count_Value => Count_Value)
 port map (
 clk => clk,
 Sig => rst ,                                        --debounced reset signal
 Debounced_Signal => int_debounced_rst_sig );   --debounced enable
signal
 end generate;

Async_ROM_Fib_val_adr_0_to_13 : entity work.Async_ROM14x8
port map (
 Address => int_counter_and_address_val,
 DataOut =>  Fib_val);

end Behavioral;
```

Figure 17 Modified VHDL Component

VHDL Test Bench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
-- testing procedure
-- full system reset to aquire stable known values
-- tests correct functionality of both debouncers
-- input signals for debouncers must be help for the value
-- of "Count_Value" - 1 clk cycles before the debouncer output
-- will change. both debouncers will be tested for less than
-- the minimum clk cycles and above the minumum clk cycles
-- to prove predicted functionality.
-- Fibonacci sequencer will be tesetd via incrementing values
-- then resetting the system and incrementing again until
-- sequencer value rolls back over to zero.
-- Xilinx debouncers will be checked for implementation
-- via the change of the variable “Simulation” true for Xilinx
-- false for parameterizable debouncers.
-- clk_periods of below 5 prove Xilinx Debouncers implemented.
entity Fibonacci_Gen_Val_1st_14_VAl_tb is
end Fibonacci_Gen_Val_1st_14_VAl_tb;

architecture Behavioral of Fibonacci_Gen_Val_1st_14_VAl_tb is
-- Constants
-- debouncer counter values
constant Count_Value  : natural := 50000000;
constant clk_period : time := 10ns;
-- true = Xilynx debouncer
--false = Parameterizable debouncer
constant  Simulation : boolean := true;

--input signals
signal clk : STD_LOGIC;
signal rst : STD_LOGIC;
signal count  : STD_LOGIC;
--output signals
signal Fib_val : STD_LOGIC_VECTOR (7 downto 0);

begin

UUT: entity work.Fibonacci_Gen_Val_1st_14_val
-- maps counter value to the debouncer counters
-- maps simulation true or false value to choose
-- which debouncer desing to implement
GENERIC MAP( Count_Value => Count_Value,
             Simulation => Simulation )

PORT MAP (clk  => clk ,
          rst => rst,
          Count  => Count,
          Fib_val => Fib_val);

-- Clock process
clk_process :process
begin
 clk <= '0';
 wait for clk_period/2;
 clk <= '1';
 wait for clk_period/2;
end process;
```

```
TEST : process
begin
-- wait 100 ns for global reset tot finish set by Xilinx
wait for 100 ns;

wait until falling_edge(clk);

-- set inputs to known values
 wait for clk_period;
 rst   <= '0';
 count <= '0';
 wait for clk_period;


-- below minimum number of clk cycles needed, debounce signal
-- will be low
 rst <= '1';
 wait for clk_period*1;
 rst <= '0';
 wait for clk_period*1;

 -- above minimum number of clk cycles needed, debounce signal
 -- will be high causing system reset
  rst <= '1';
  wait for clk_period*2;
  rst <= '0';
  wait for clk_period*2;

 -- below minimum number of clk cycles needed, debounce signal
  -- will be low
  wait for clk_period*1;
  count <= '1';
  wait for clk_period*1;
  count <= '0';

 -- above minimum number of clk cycles needed, debounce signal
 -- will be high

 for i in 0 to 10 loop
         wait for clk_period*2;
         count <= '1';
         wait for clk_period*2;
         count <= '0';
     END LOOP;
-- shows reset debouncer works correctly and shows accurate system reset
wait for clk_period*2;
rst <= '1';
wait for clk_period*2;

      rst <= '0';
-- allows the sequencer to increment the output value for correct
functionality
 for i in 0 to 16 loop
               wait for clk_period*2;
               count <= '1';
               wait for clk_period*2;
               count <= '0';
           END LOOP;
 wait; --waits forever
end process;

end Behavioral;
```

Simulations

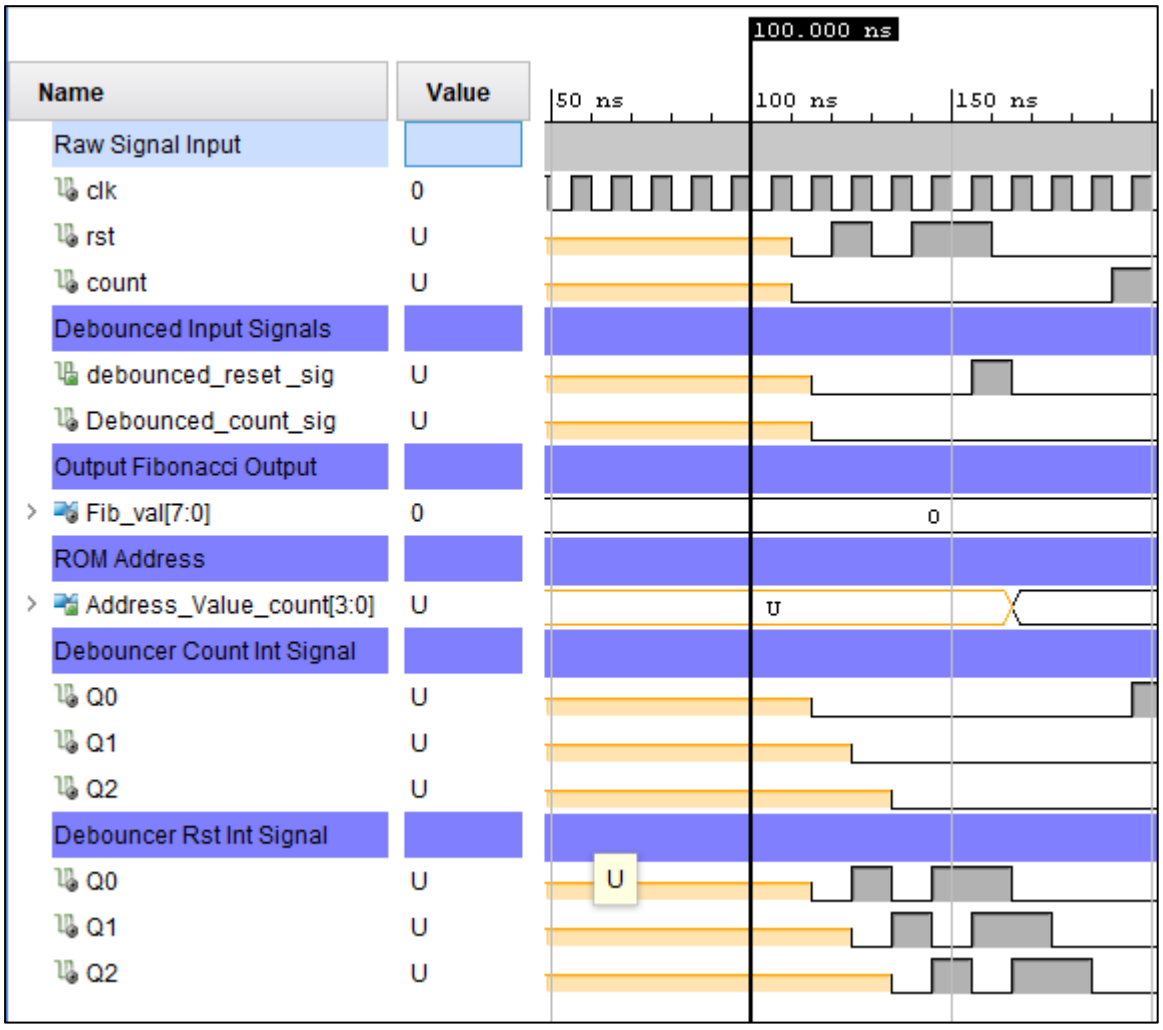

Figure 18 100ns Initialisation time, less than 2 clk signals on reset no Debounce and 2 clk signal reset input signal debounced, proves if generate construct works as Xilinx debouncers implemented

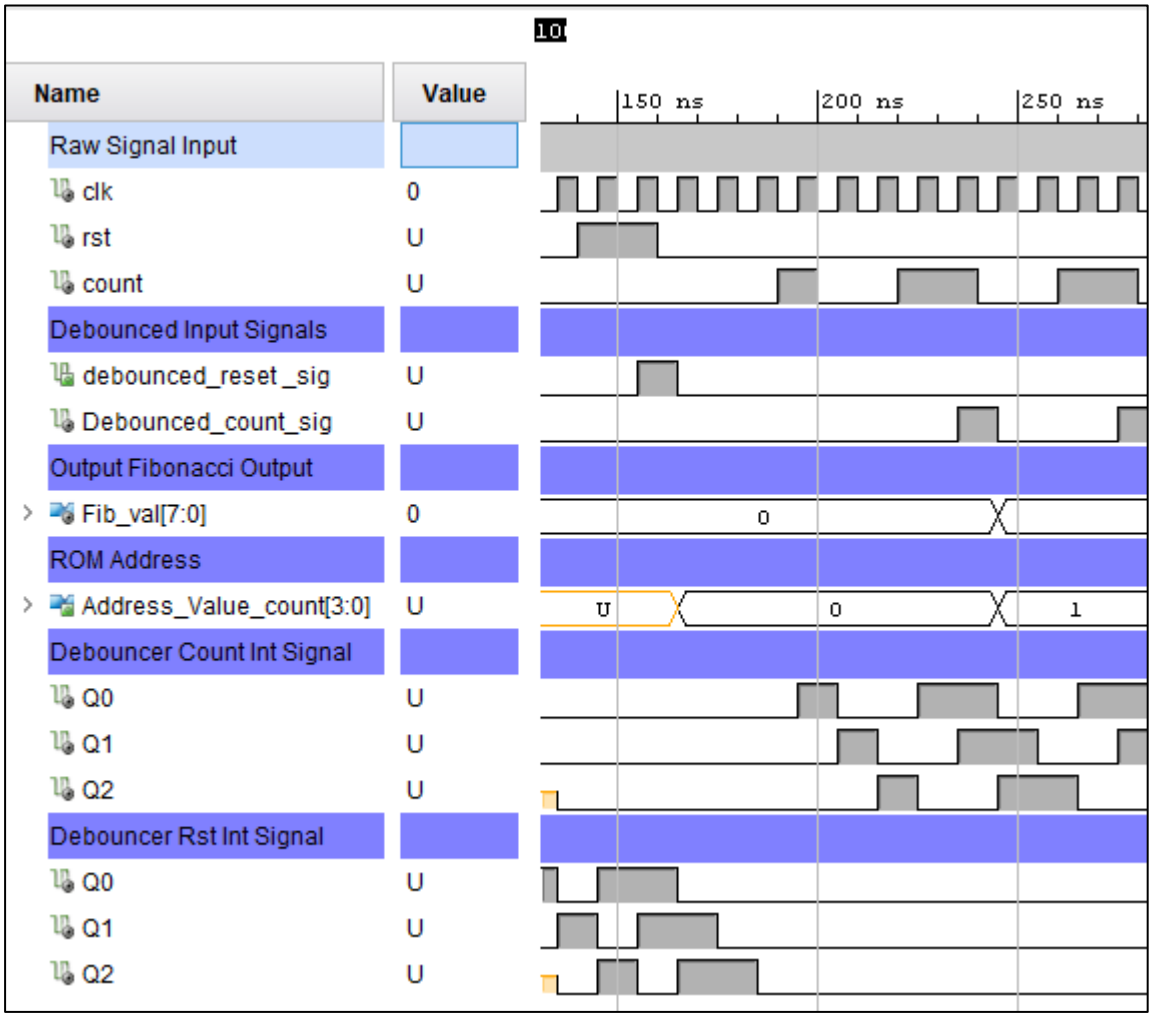

Figure 19 less than 2 clk signals on Count no Debounce and 2 clk signal count input signal debounced proves if generate construct works as Xilinx debouncers implemented

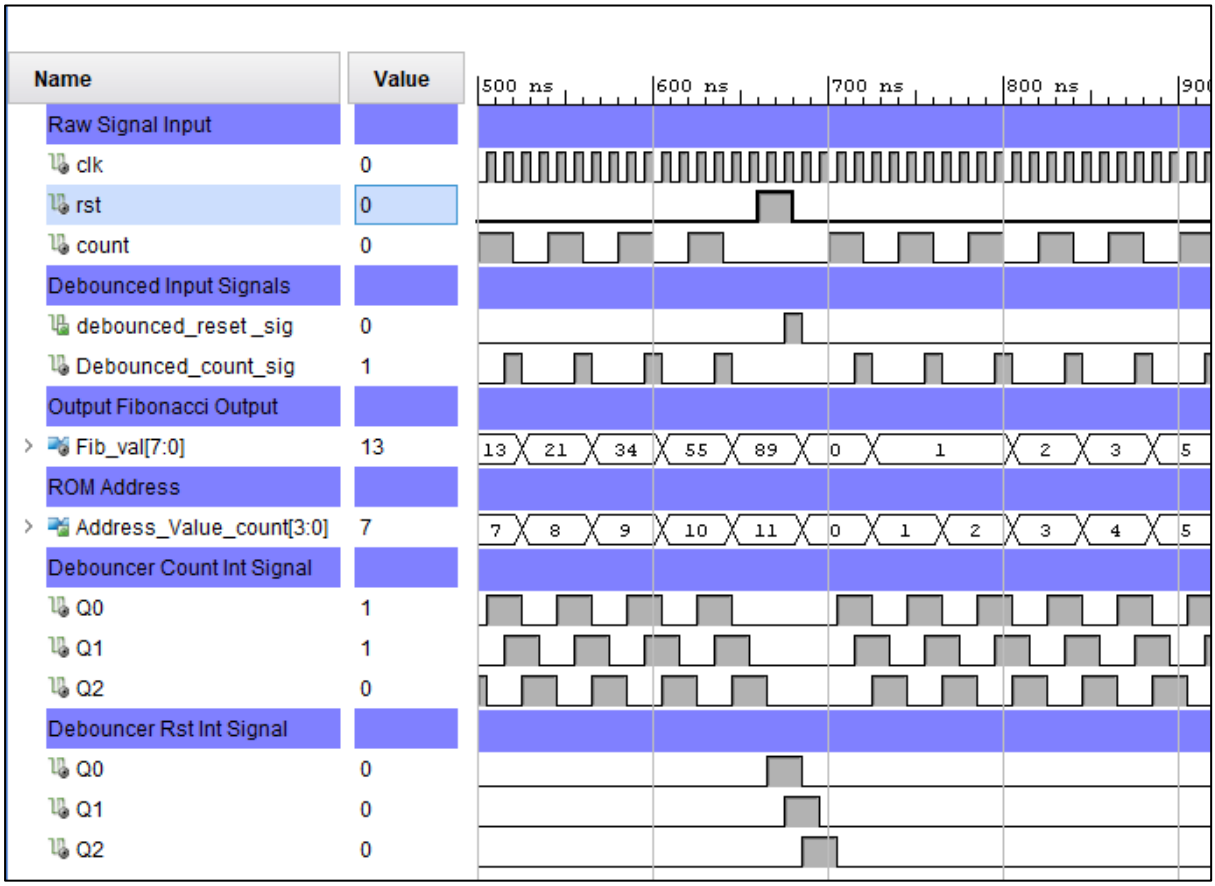

Figure 20 Reset of entire system bringing Fibonacci sequencer back to zero

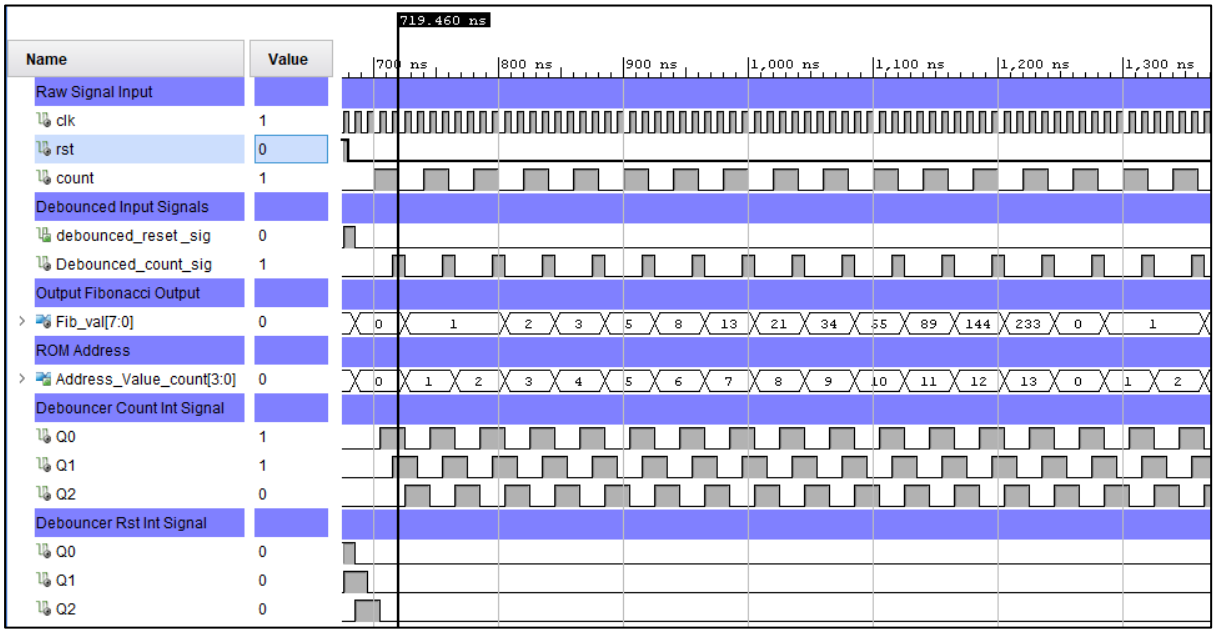

Figure 21 Full iteration of Fibonacci sequence and counter roll over back to zero

RTL Component Statistics & RTL Hierarchical Component Statistics

Counter size is 26 bits, as expected as you can count to 50 million using 2^26, even though the simulation implements the much smaller debouncer.

```
---------------------------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
       2 Input     26 Bit       Adders := 2
       2 Input      4 Bit       Adders := 1
+---Registers :
                   26 Bit     Registers := 2
                    4 Bit     Registers := 1
                    1 Bit     Registers := 2
+---Muxes :
       2 Input     26 Bit        Muxes := 2
       2 Input      4 Bit        Muxes := 1
       2 Input      1 Bit        Muxes := 2
---------------------------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------------------------
---------------------------------------------------------------------------
Start RTL Hierarchical Component Statistics
---------------------------------------------------------------------------
Hierarchical RTL Component report
Module Counter_0_to_13
Detailed RTL Component Info :
+---Adders :
       2 Input      4 Bit       Adders := 1
+---Registers :
                    4 Bit     Registers := 1
+---Muxes :
       2 Input      4 Bit        Muxes := 1
Module D_type_FF
Detailed RTL Component Info :
+---Registers :
                    1 Bit     Registers := 1
Module Param_Count_to_N
Detailed RTL Component Info :
+---Adders :
       2 Input     26 Bit       Adders := 1
+---Registers :
                   26 Bit     Registers := 1
+---Muxes :
       2 Input     26 Bit        Muxes := 1
Module Parameterizable_Debouncer
Detailed RTL Component Info :
+---Muxes :
       2 Input      1 Bit        Muxes := 1
---------------------------------------------------------------------------
Finished RTL Hierarchical Component Statistics
---------------------------------------------------------------------------
```