Forelesning 06: Regular Sequential Circuits

Hieu Nguyen

Innold

- Konstruerer Mod-M teller ved bruk **synkron metode**
- Simulerer Mod-M teller
- Tester **Mod-16** teller i Basys-3 ved bruk **hex2seg krets**

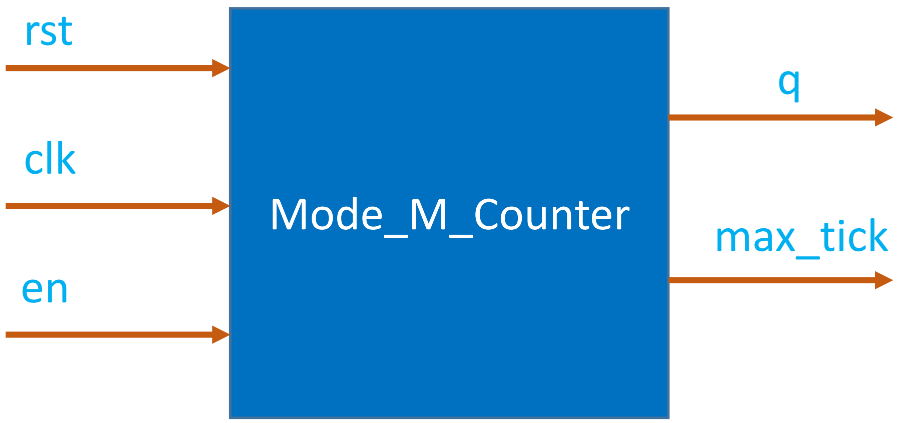
Design 05: Mod-M teller

- **Mod-M** teller er en krets som øker sin verdi 1 om gangen og starter fra **0** til **M-1** og hopper rundt (wraps around).
- For eksempel: Mod-10 teller skal telle 0, 1,2,3,4,5,6,7,8,9,0,1….
- **Mod-M** teller bør ha to **generics**
  - **M**: spesifiserer tellerbegrensning
  - **N**: spesifiserer antall nødvendige biter i state register. **N = log2 (M)**
  - **Eksempel**: Hvis en konstruerer Mod-16 teller. Dette betyr at **M = 16**, og telleren har 16 tilstander i intern minne. Derfor, trenger vi **N =4** D-FF stykker for å lagre alle tilstander.

Design 05: Mod-M teller

- **Inngangssignaler**
  - Clock signal - clk
  - Reset signal - rst
  - Enable signal – en
- **Utgangssignaler**
  - Tellers verdi – q
  - Max tick – max_tick

Entity Declaration



Design 05: Mod-M teller

State register

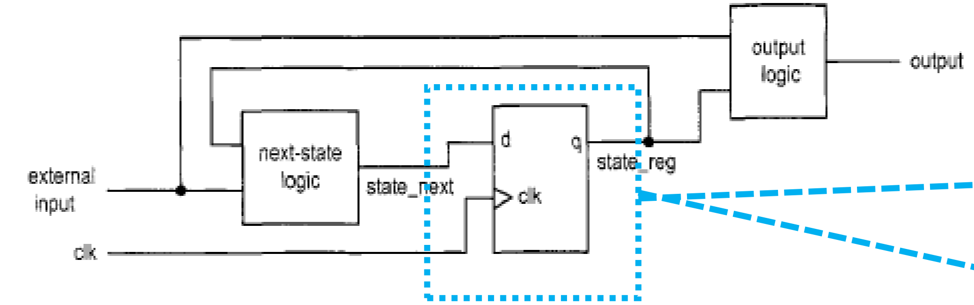
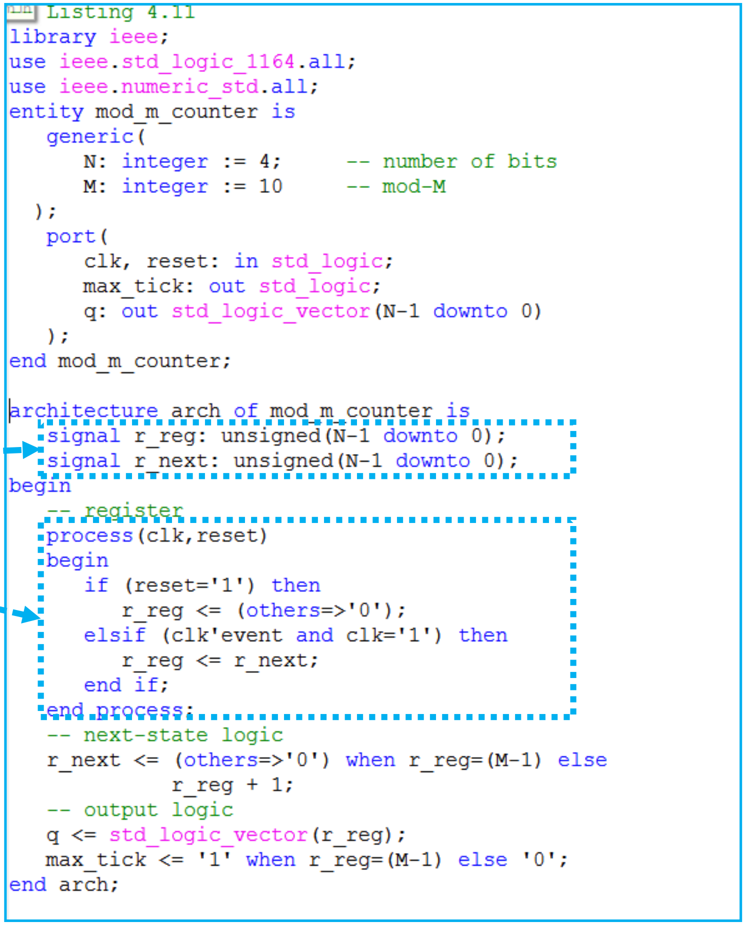


Figure 4.2 Block diagram of a synchronous system.



```
-- Listing 4.11
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity mod_m_counter is
   generic(
      N: integer := 4;     -- number of bits
      M: integer := 10     -- mod-M
  );
   port(
      clk, reset: in std_logic;
      max_tick: out std_logic;
      q: out std_logic_vector(N-1 downto 0)
   );
end mod_m_counter;

architecture arch of mod_m_counter is
   signal r_reg: unsigned(N-1 downto 0);
   signal r_next: unsigned(N-1 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic
   r_next <= (others=>'0') when r_reg=(M-1) else
             r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_tick <= '1' when r_reg=(M-1) else '0';
end arch;
```

Design 05: Mod-M Teller

Next state logic

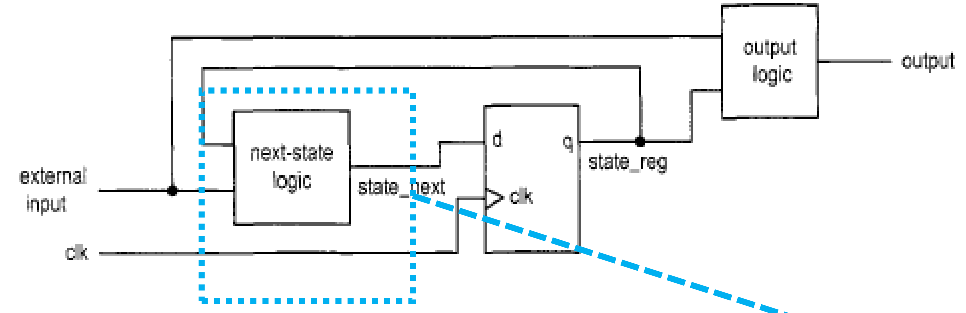
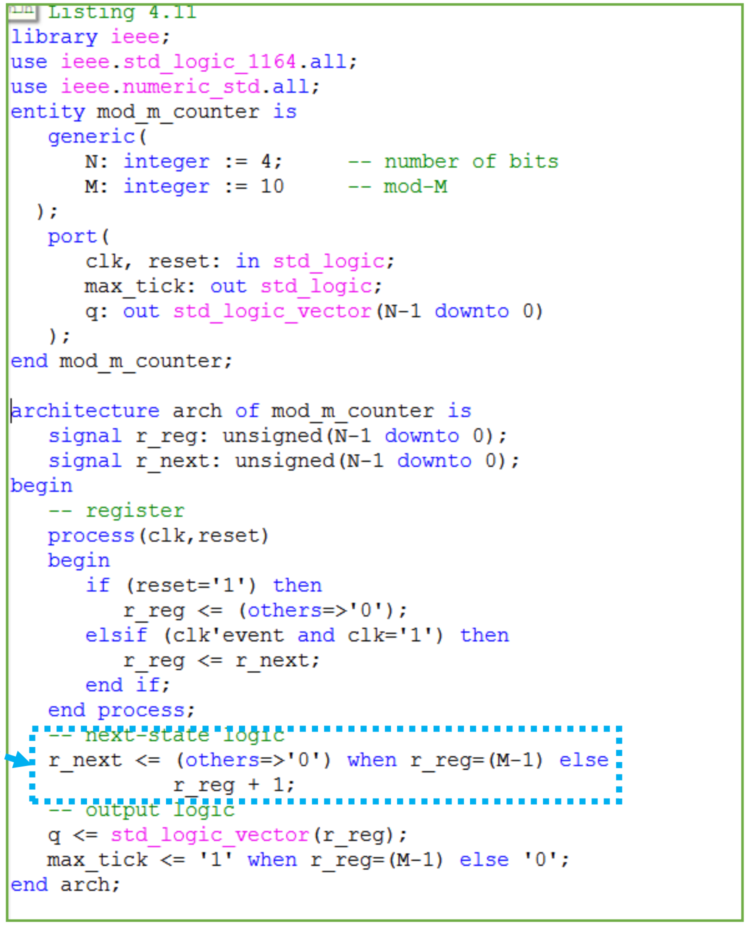


Figure 4.2 Block diagram of a synchronous system.



```
Listing 4.11
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity mod_m_counter is
   generic(
      N: integer := 4;     -- number of bits
      M: integer := 10     -- mod-M
  );
   port(
      clk, reset: in std_logic;
      max_tick: out std_logic;
      q: out std_logic_vector(N-1 downto 0)
   );
end mod_m_counter;

architecture arch of mod_m_counter is
   signal r_reg: unsigned(N-1 downto 0);
   signal r_next: unsigned(N-1 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic
   r_next <= (others=>'0') when r_reg=(M-1) else
             r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_tick <= '1' when r_reg=(M-1) else '0';
end arch;
```

Design 05: Mod-M Teller

Output Logic

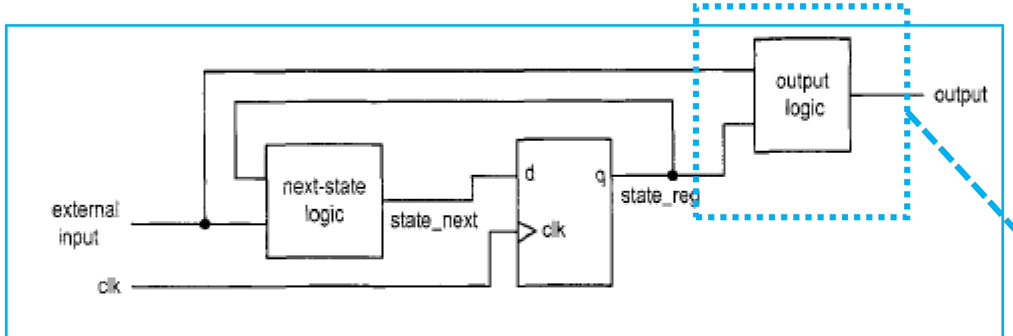
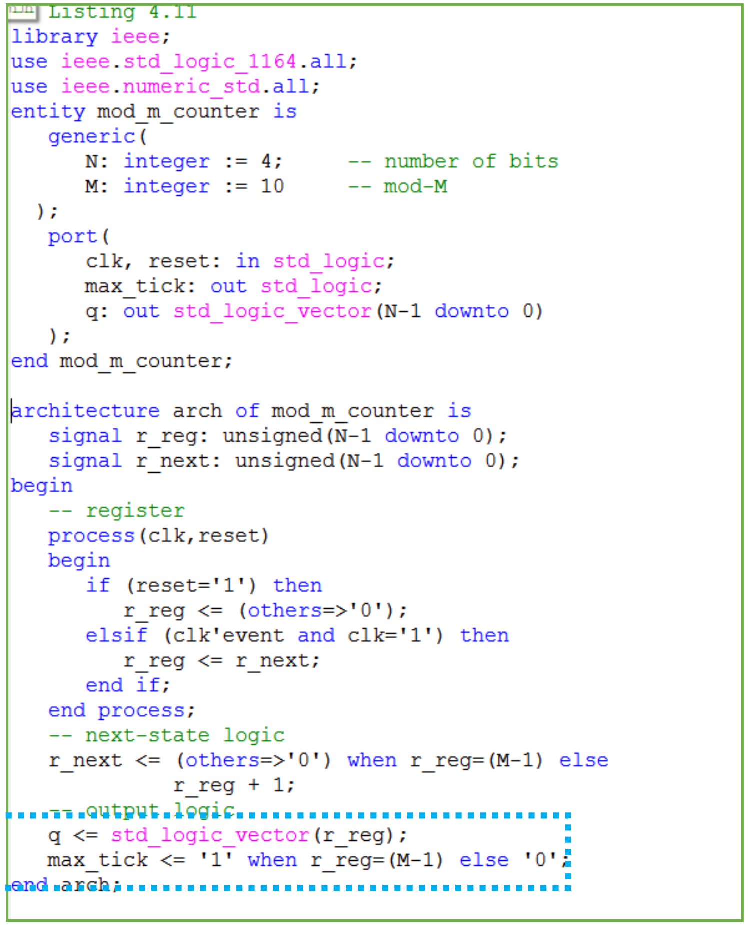


Figure 4.2 Block diagram of a synchronous system.



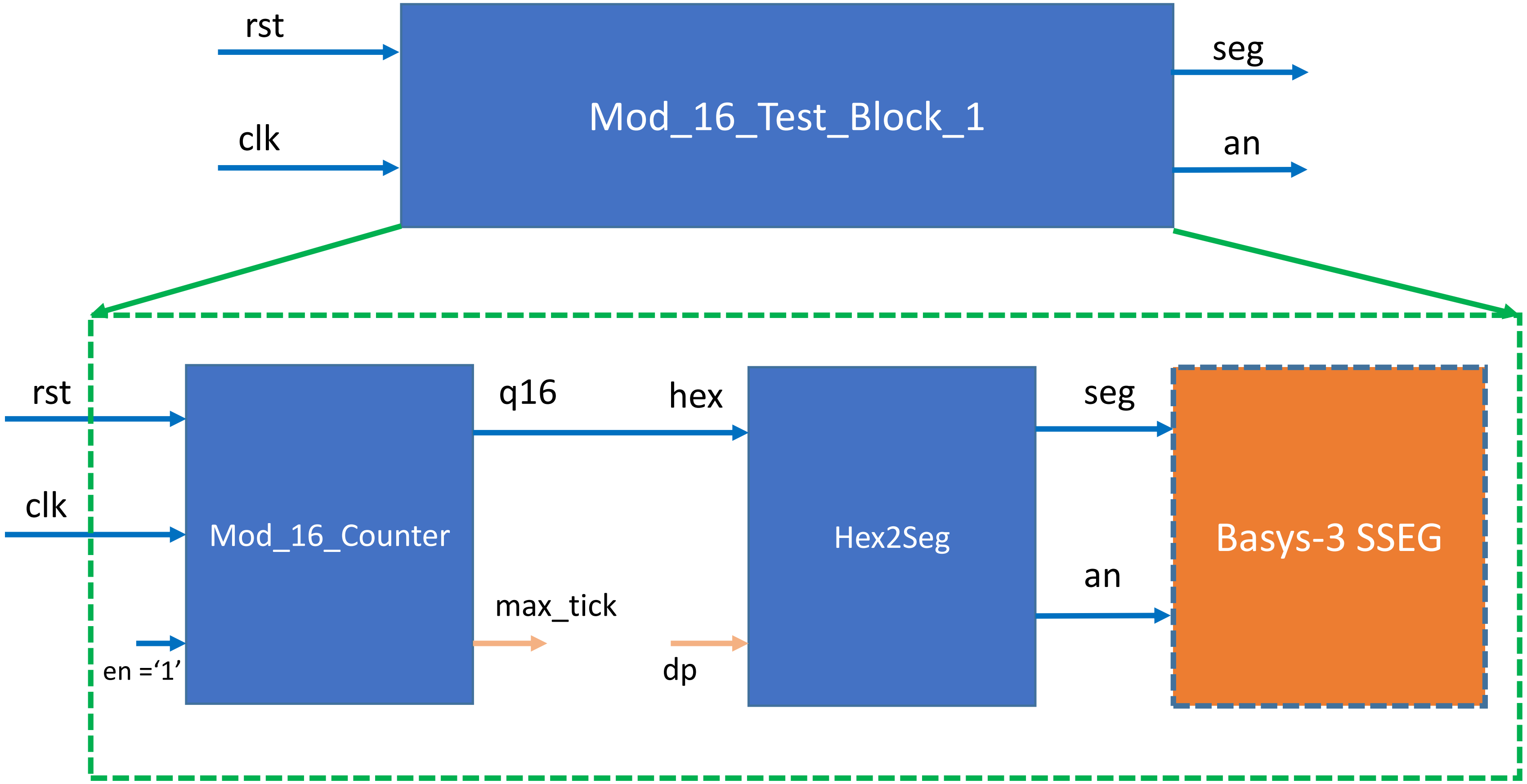
```
Listing 4.11
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity mod_m_counter is
   generic(
      N: integer := 4;     -- number of bits
      M: integer := 10     -- mod-M
  );
   port(
      clk, reset: in std_logic;
      max_tick: out std_logic;
      q: out std_logic_vector(N-1 downto 0)
   );
end mod_m_counter;

architecture arch of mod_m_counter is
   signal r_reg: unsigned(N-1 downto 0);
   signal r_next: unsigned(N-1 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic
   r_next <= (others=>'0') when r_reg=(M-1) else
             r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_tick <= '1' when r_reg=(M-1) else '0';
end arch;
```

Test Bench – Test Mod-16 Counter

Component Declaration	→	mod_m_counter
Constant Declaration	→	N,M, CLK_Period (10 ns)
Testbench | Internal Signal Declaration	→	rst, clk, en, q, max_tick
Component Instantiation | UUT	→	mod_m_counter
Clock Signal Generation	→	clk
Generation Of Other Input Signals	→	rst, en

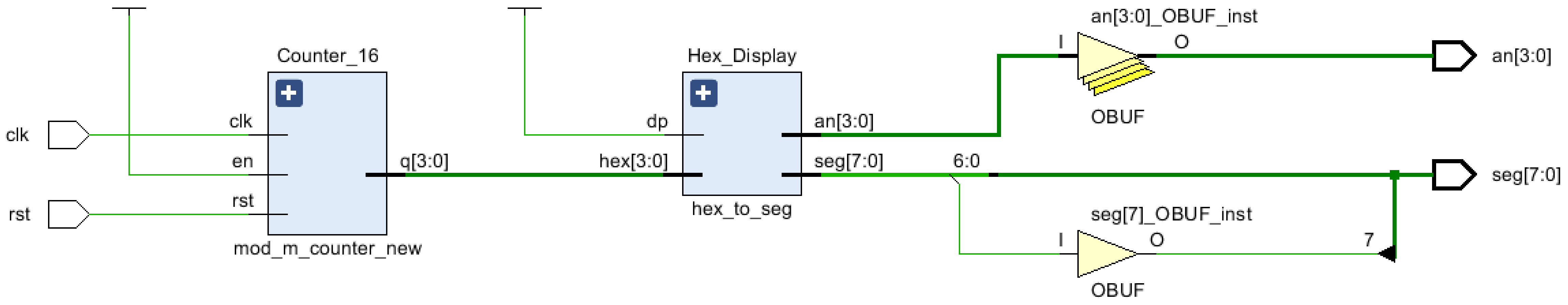
Mod-16 Counter | HardWare Testing (1)



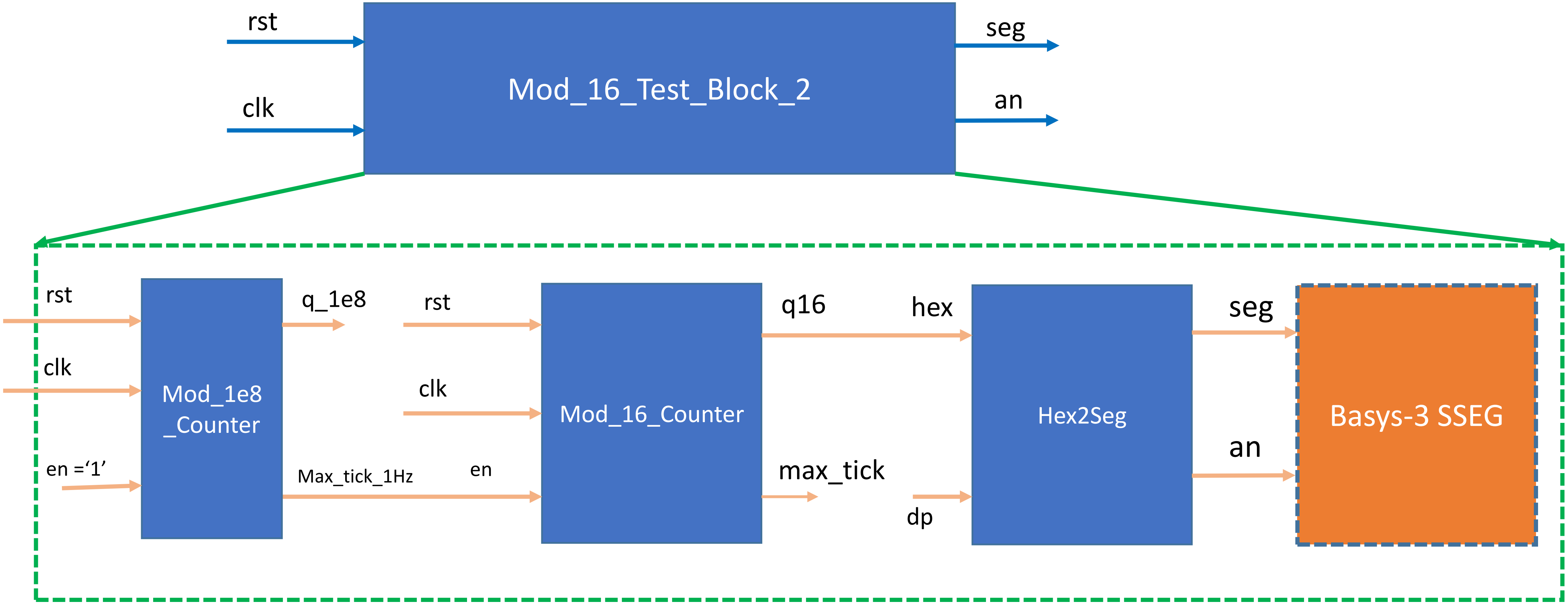
Pin Connections

Design	Basys-3
rst	btnC
clk	clk
seg	seg
an	an

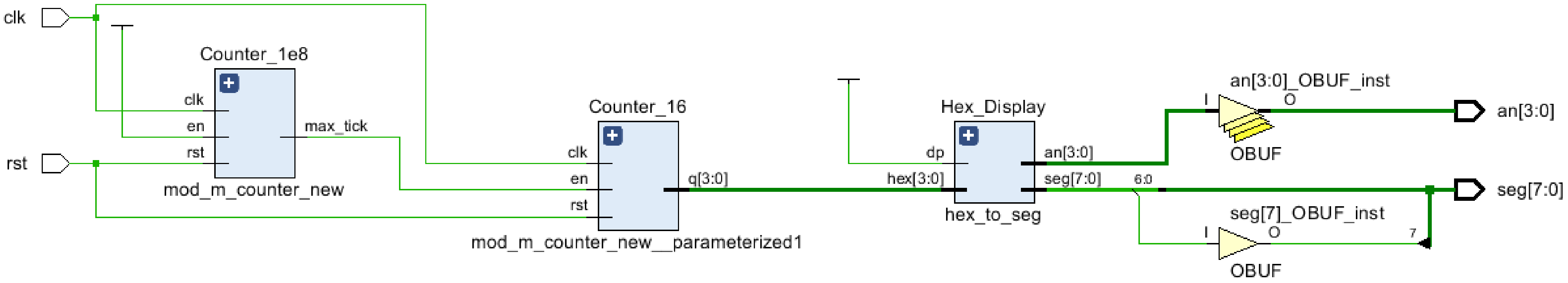
Design 05: Mod-M Teller | Test



Mod-16 Counter | HardWare Testing (2)



Design 05: Mod-M Teller | Test


clk
rst
Counter_1e8
clk
en
rst
max_tick
mod_m_counter_new
Counter_16
clk
en
rst
q[3:0]
mod_m_counter_new__parameterized1
Hex_Display
dp
hex[3:0]
an[3:0]
seg[7:0]
hex_to_seg
6:0
an[3:0]_OBUF_inst
I
O
OBUF
seg[7]_OBUF_inst
I
O
7
OBUF
an[3:0]
seg[7:0]