EE 789 : Assignment 1

Shift and Subtract 8 bit Divider

Ashish Pol
190070011

August 30, 2022

Contents

Aim

Design a shift and subtract (long-division) 8-bit divider in RTL. Given A, B, if the result is Q, then $A = (B \times Q) + R$, where R is an integer $<$ B.

Binary Division

The following example show the method of binary division 10100111 (dividend) divided by 1101(divisor) 167 (10100111) divided by 13 (1101) gives 12 (1100) quotient and 11 (1011) remainder

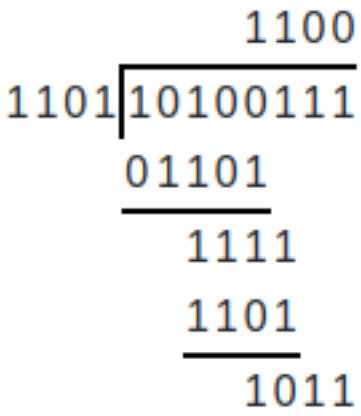

Figure 1: Binary Division

⇒ The MSB of the dividend is inspected first to determine if the divisor can be subtracted from this dividend digit.

⇒ The quotient is accordingly marked with a 1 if it is true; otherwise, it is marked with a 0.

⇒ Dividend's next digit is added after the remainder of the division process is carried out.

⇒ This procedure is repeated until the right LSB of the dividend is found.

In this example

⇒ We take the first bit of the dividend, which is 1. This is smaller than the divisor 1101, so this bit in the quotient is 0, and the remainder of 1 is kept.

⇒ The next digit of the dividend is added, but 10 is still smaller than the divisor 1101, so the quotient bit is 0.

⇒ The next digit of the dividend is added, but 101 is still smaller than the divisor 1101, so the quotient bit is 0.

⇒ The next digit of the dividend is added, but 1010 is still smaller than the divisor 1101, so the quotient bit is 0.

⇒ The next dividend digit is added: 10100. This is bigger than the divisor, so we subtract 1101 from 10100, which leaves a remainder of 111. The quotient bit is set to 1.

⇒ The dividend bit is added to the remainder, which results in 1111. This is bigger than 1101; subtract the divisor from it, which leaves 0010 as a remainder. The corresponding quotient bit is 1.

⇒ The next digit of the dividend is added, but 101 is still smaller than the divisor 1101, so the quotient bit is 0.

⇒ The last digit of the dividend is added, but 1011 is still smaller than the divisor 1101, so the quotient bit is 0.

⇒ The final result is the quotient 00001100 and the remainder 1011.

Pseudo code

```
-- Shift and add multiplier.
-- Uses a single 8-bit subtractor.
--
-- Does an 8-bit divide in
-- 1+8 clock cycles.
--
-- input start, numerator[7:0], denominator[7:0]
-- output done, remainder[7:0], quotient[7:0]
--
-- register tr[8:0]
-- register tq[7:0]
-- registers counter[3:0]
--
-- Default outs:
--
done=0, remainder, quotient= 0
rst:
if start then
tr[8:0] := 0
counter := 0
tq := numerator
goto loop
else
goto rst
endif
loop:
if(tr[7:0] >= denominator) then
-- 8-bit subtractor.
tr[8:1] := tr[7:0] - denominator;
-- shift left and concat 1.
tq := (tq[6:0] & '1');
else
-- shift left.
tr[8:1] := tr[7:0];
--shift left and concat 0.
tq := (tq[6:0] & '0');
endif
-- left shift the numerator into the remainder.
tr[0] := tq[7]
if (counter == 8) then
goto done_state
else
counter := (counter + 1)
goto loop
endif
done_state:
done = 1
```

```
-- tr is visible at remainder and tq is visible at quotient
remainder = tr[8:1]
quotient = tq[7:0]
if start then
tr[8:0] := 0
counter := 0
tq := numerator
goto loop
else
goto done_state
endif
```

RTL : VHDL implementation

```
library ieee;
use ieee.std_logic_1164.all;
package RtlFsmTypes is
    type StateSymbol is (rst, loop_state, done_state);
end package;

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library work;
use work.RtlFsmTypes.all;
entity ShiftAndSubtractDivider is
    port (
        reset, clk: in std_logic;
        start: in std_logic;
        numerator, denominator: in std_logic_vector(7 downto 0);
        done: out std_logic;
        quotient, remain: out std_logic_vector(7 downto 0)
     );
end entity;

architecture Obvious of ShiftAndSubtractDivider is
    signal Q: StateSymbol;
    signal tr: std_logic_vector(8 downto 0);
    signal tq: std_logic_vector(7 downto 0);
    signal counter: unsigned(3 downto 0);
begin
    process(clk, reset, start, Q, tr, tq, counter)
        variable nextQ : StateSymbol;
        variable done_var:   std_logic;
        variable next_tr: std_logic_vector(8 downto 0);
        variable next_tq: std_logic_vector(7 downto 0);
        variable next_counter: unsigned(3 downto 0);
        variable remain_var: std_logic_vector(7 downto 0);
        variable quotient_var: std_logic_vector(7 downto 0);
    begin
```

```
        nextQ := Q;
        next_tr := tr;
        next_tq := tq;
        next_counter := counter;
        done_var := '0';
        remain_var := (others => '0');
        quotient_var := (others => '0');

        case Q is
            when rst =>
                if(start = '1') then
                    next_tr := (others => '0');
                    next_tq := numerator;
                    next_counter := (others => '0');
                    nextQ := loop_state;
                end if;

            when loop_state =>
                if(tr(7 downto 0) >= denominator) then
                    next_tr(8 downto 1) := std_logic_vector(unsigned(tr(7 downto 0))
                    - unsigned(denominator));
                    next_tq := (tq(6 downto 0) & '1');
                else
                    next_tr(8 downto 1) := tr(7 downto 0);
                    next_tq := (tq(6 downto 0) & '0');
                end if;
                next_tr(0) := tq(7);

                if(counter = 8) then
                    nextQ := done_state;
                 else
                    next_counter := (counter + 1);
                 end if;

            when done_state =>
                done_var := '1';
                remain_var := tr(8 downto 1);
                quotient_var := tq(7 downto 0);
                if(start = '1') then
                    next_tr := (others => '0');
                    next_tq := numerator;
                    next_counter := (others => '0');
                    nextQ := loop_state;
                end if;
        end case;

        done <= done_var;
        quotient <= quotient_var;
        remain <= remain_var;

        if (clk'event and (clk = '1')) then
            if (reset = '1') then
```

```
                -- Note: reset state is imposed here.
                    Q <= rst;
            else
                    Q <= nextQ;
                    tr <= next_tr;
                    tq <= next_tq;
                    counter <= next_counter;
            end if;
        end if;
    end process;
  end Obvious;
```

Testbench

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TB is
end entity;

architecture TB_arch of TB is
   component ShiftAndSubtractDivider is
      port (
              reset, clk: in std_logic;
              start: in std_logic;
              numerator, denominator: in std_logic_vector(7 downto 0);
              done: out std_logic;
              quotient, remain: out std_logic_vector(7 downto 0)
           );
   end component;

  signal reset : std_logic := '1';
  signal clk : std_logic := '0';

  signal start: std_logic;
  signal numerator, denominator: std_logic_vector(7 downto 0);
  signal done: std_logic;
  signal quotient, remain: std_logic_vector(7 downto 0);
begin

        clk <= not clk after 5 ns;
        process
        begin
                start <= '0';
                wait until clk = '1';
                reset <= '0';

                for I in 1 to 255 loop
                        numerator <= std_logic_vector(to_unsigned(I,8));
                        for J in 1 to 255 loop
```

```
                                denominator <= std_logic_vector(to_unsigned(J,8));
                                start <= '1';
                                wait until clk = '1';
                                while true loop
                                    wait until clk = '1';
                                    -- start is asserted for only one cycle.
                                    start <= '0';
                                    if  (done = '1') then
                                        exit;
                                    end if;
                                end loop;
                                assert (to_integer(unsigned(quotient)) = (I/J) and
                                 to_integer(unsigned(remain)) = (I mod J)) report
                                        "Mismatch!" severity error;
                        end loop;
                end loop;
                assert false report "Test completed." severity note;

                wait;
        end process;

        dut: ShiftAndSubtractDivider
                port map (reset => reset, clk => clk,
                                start => start, numerator => numerator,
                                denominator => denominator,
                                done => done, quotient => quotient, remain => remain);
end TB_arch;
```