Design of 3-bit Syncronous Systems

Ilya Dmitrichenko

January 11, 2011

Abstract

This report consists of two parts. First part contains detailed implementation and simulation of a 3-bit binary counter designed according to given specification. Second part describes full implementation of another very fundamental digital system block, in this case it's a 3 bit controller circuit. Counters and controllers together with multiplexers, encoders as well as decoders and demultiplexers are the fundamental blocks of any digital electronics system. We are looking at these systems from very high level of abstraction, such as C programming language or VHDL for electronics design engineers. Although, recently some implementations were seen with even higher level of abstraction[1].

[1] High-level object-oriented *Python MyHDL* module library:
http://www.myhdl.org/doku.php/why

Contents

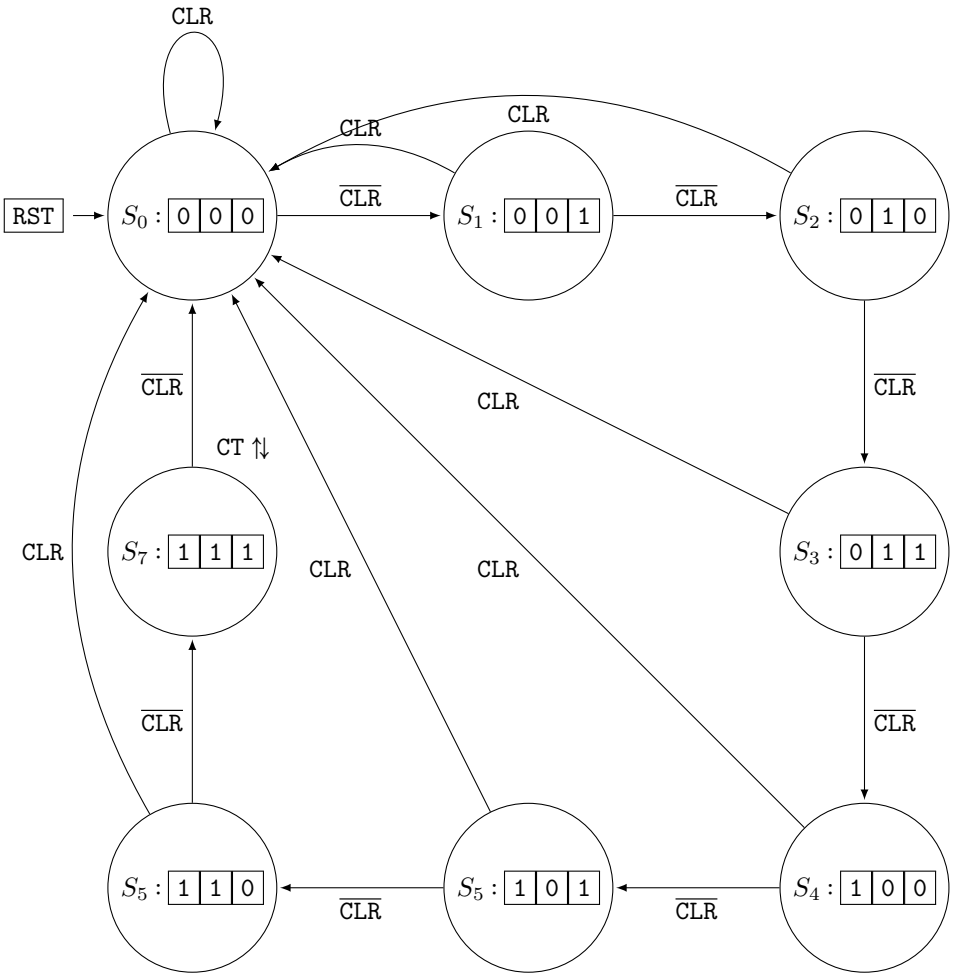
Part I

Synchronous 3-bit Counter

1.1 Design Specification

- Asynchronous Inputs:
  - `RST` - resets all output bits to `'000'` (*active-low*)
- Synchronous Inputs:
  - `CLR` - clears output bits to `'000'` (*active-high*)
- Output Flags:
  - `CT` - count terminate is asserted high when output is `'111'`

1.2 States and Transitions



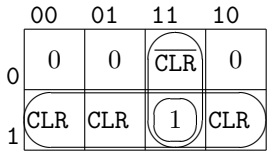
State	Q_2	Q_1	Q_0	CLR	Q_2	Q_1	Q_0	T_2	T_1	T_0	CT
S_0	0	0	0	0	0	0	1	0	0	1	0
				1	0	0	0	0	0	0	
S_1	0	0	1	0	0	1	0	0	1	1	0
				1	0	0	0	0	0	1	
S_2	0	1	0	0	0	1	1	0	0	1	0
				1	0	0	0	0	1	0	
S_3	0	1	1	0	1	0	0	1	1	1	0
				1	0	0	0	0	1	1	
S_4	1	0	0	0	1	0	1	0	0	1	0
				1	0	0	0	1	0	0	
S_5	1	0	1	0	1	1	0	0	1	1	0
				1	0	0	0	1	0	1	
S_6	1	1	0	0	1	1	1	0	0	1	0
				1	0	0	0	1	1	0	
S_7	1	1	1	x	0	0	0	1	1	1	1
				x	0	0	0	1	1	1	

1.3 Logic Implementation

There is no particular need to show a K-map for CT output, as it appears to be quite sreight forward:

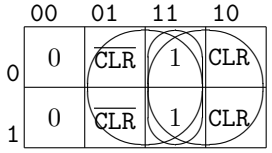
$$\texttt{CT} = Q_2.Q_1.Q_0$$

$T_2 = f(Q_2, Q_1, Q_0)$:



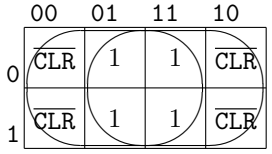
$$T_2 = Q_2.Q_1.Q_0 + Q_1.Q_0.\overline{CLR} + Q_2.CLR$$

$T_1 = f(Q_2, Q_1, Q_0)$:



$$T_1 = Q_1.Q_0 + Q_1.CLR + Q_0.\overline{CLR}$$

$T_0 = f(Q_2, Q_1, Q_0)$:



$$T_0 = Q_0 + \overline{CLR}$$

1.4 Software Simulation

The logic circuit (Figure 1.4.1) had been simulated with 10MHz clock rate, the waveform output is shown in Figure 1.2.

By observing the waveform, asynchronous `reset` versus synchronous `clear` control input operation had been verified to be certainly correct. Also the count output is in the range from 0 to 7 appears as expected for a 3-bit up-counter.

1.4.1 Timing Constraints

In terms of standard flip-flop delay timings, as an example, let's take devices from one of very large manufacturers - *NXP Semiconductors*[2]. All flip-flop ICs from *NXP* are characterised with nominal maximum operating frequency of lower bound at 25MHz and upper bound at 250MHz (only a few *NXP* devices have maximum frequency value outside of this range). The discrete logic ICs form *NXP*[3] have propagation delay starting as low as 200ps and maximum around 10ns.

The timing characteristics of a logic integrated circuit are defined by semiconductor fabrication technology, operating voltage and pin-count, therefore it very much depends on particular application requirements. Power consumption and product form-factor would play a significant role. However a counter, as it had been said, is a very generic component in complex logic circuits such as microcontrollers and microprocessors. If we were to implement a processor using discrete logic & flip-flop chips, it would not be possible to make it much faster then 200MHz. Also the circuit board exhibits impedance affecting this figure to a certain extend.

Larger scale of integration is widely used in modern design, reconfigurable circuits are seen as future of digital systems. Though, it is very important to understand the basic principles of logic synthesis as shown above.

[2] *NXP discrete flip-flop IC product range:* `http://ics.nxp.com/products/flipflops/`
[3] *NXP discrete logic IC product range:* `http://ics.nxp.com/products/gates/`

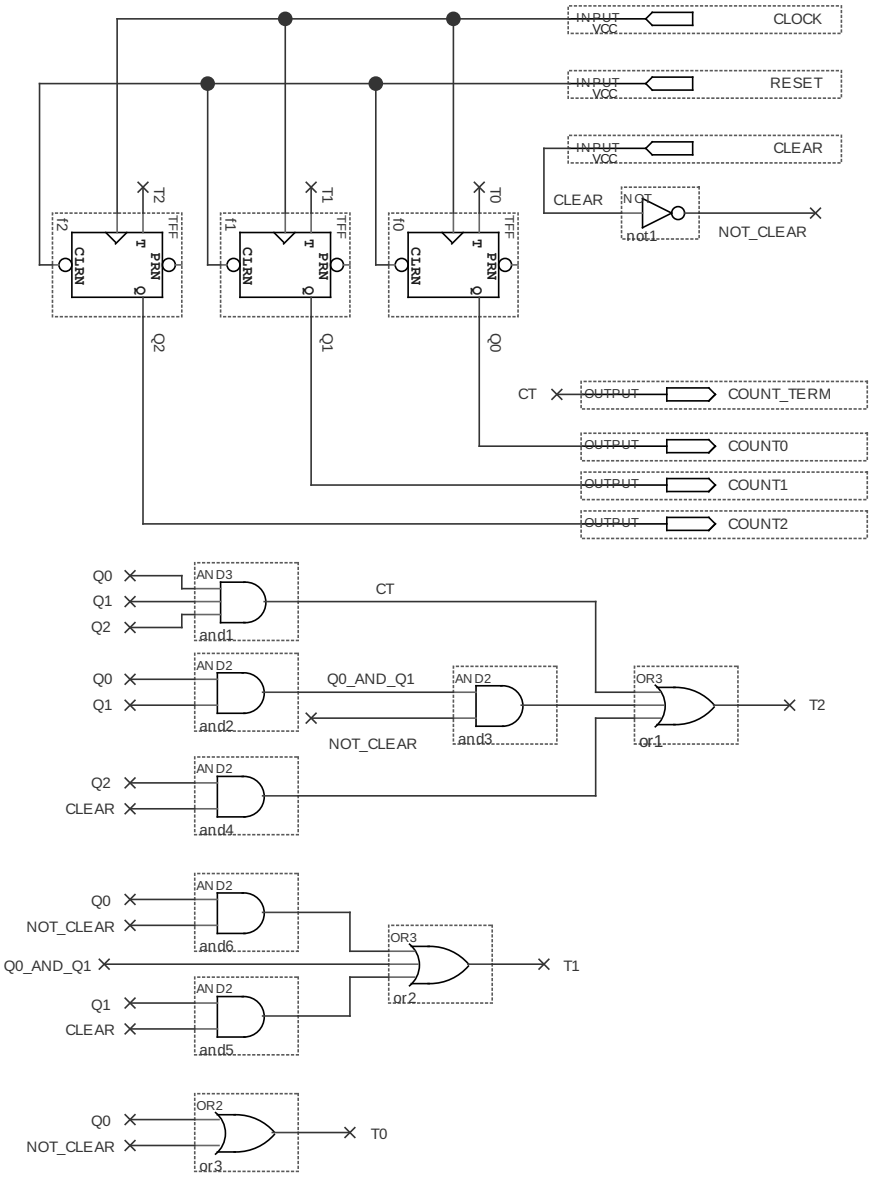


Figure 1.1: 3-bit counter logic circuit diagram.

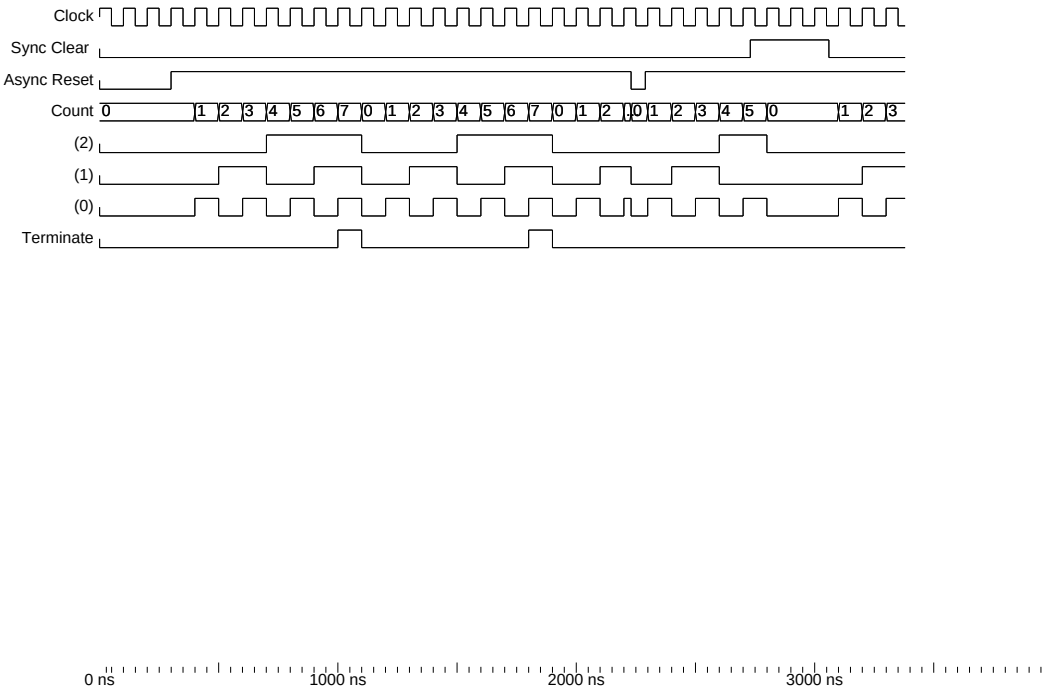


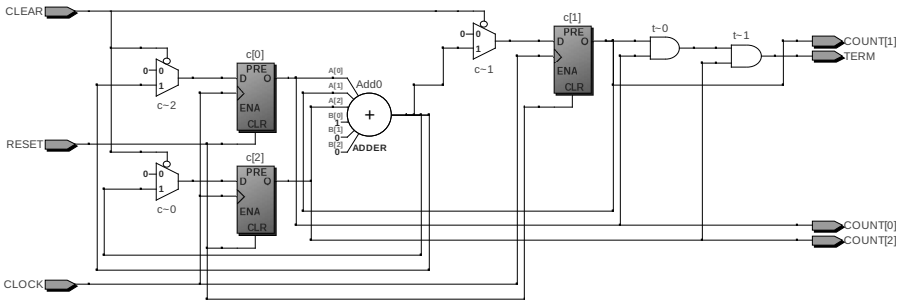
Figure 1.2: Simulation results at 10MHz clock rate (full-view)

1.4.2 VHDL

As a metter of practice and proof of concept, the counter was also implemented in VHDL and synthesized using *Altera Quartus II* design package. For the purpose of this report only graphical register transfer level (RTL) view of synthesized circuit is include together with VHDL code listing below.

The simulation described in previous section has also been performed using *Altera* software, but the circuit entry had been done using graphical schematic editor. Therefor the Boolean equation derived from Karnaugh maps above, were used instead of the higher-level behavioural code.

Figure 1.3: RTL view using D-type flip-flops, 2:1 multiplexers and 3-bit adder.




```
library IEEE;
    use IEEE.STD_LOGIC_1164.all;
    use IEEE.STD_LOGIC_ARITH.all;
    use IEEE.STD_LOGIC_UNSIGNED.all;

Entity counter1 is
  Port ( CLOCK: in      std_Logic;
         RESET: in      std_Logic;
         CLEAR: in      std_Logic;
         TERM:  out     std_Logic;
         COUNT: out     std_Logic_Vector ( 2 DOWNTO 0 )
        );
end counter1;

Architecture behavioral of counter1 is
  Signal c: std_Logic_Vector ( 2 DOWNTO 0 ) := "000";
  Signal t: std_Logic := '0';
begin

Process ( CLOCK, c, CLEAR, RESET ) begin
  If RESET = '1' then
        c <= "000";
  elsif ( RISING_EDGE(CLOCK) ) then
        If CLEAR = '0' then
              c <= c + 1;
        else
              c <= "000";
        end if;
  end if;
   t <= c(0) and c(1) and c(2);
end process;
   COUNT <= c; TERM <= t;

end behavioral;
```

Listing 1.1: *Simple 3-bit counter in VHDL*

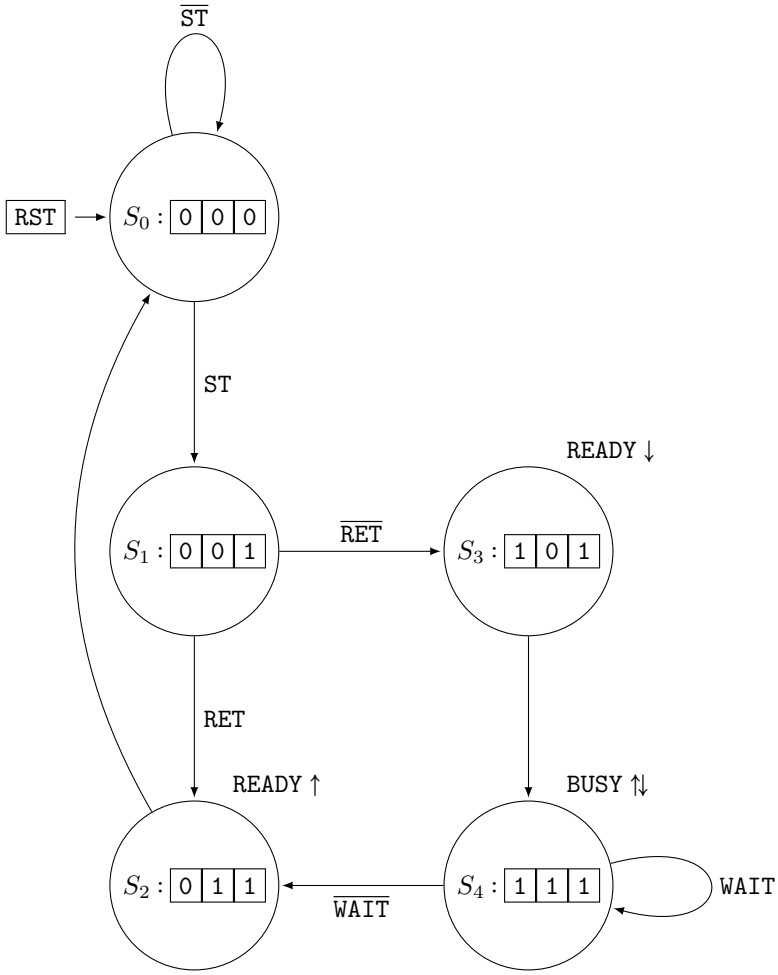
Part II

Synchronous 3-bit Controller

2.1 Design Specification

- Asynchronous Inputs:
  - `RST` - resets the state machine output to `'000'` (*active-high*)
- Syncronous Inputs:
  - `ST` - starts the machine up from S_0 (*active-low*)
  - `RET` - returns to S_0 via S_2 asserting `READY` flag high (*active-low*)
  - `WAIT` - remain in S_4 and keep `BUSY` flag high (*active-low*)
- Output Flags:
  - `BUSY` - asserted high while in S_4 (*active-high*)
  - `READY` - asserted high in S_3 and low in S_2 (*active-high*)

2.2 States and Transitions



State	Q_2	Q_1	Q_0	ST	RET	WAIT	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	READY	BUSY
S_0	0	0	0	0	x	x	0	0	1	0	x	0	x	1	x	0	1
				1	x	x	0	0	0	0	x	0	x	0	x		
S_1	0	0	1	x	0	x	0	1	1	0	x	1	x	x	0	0	1
				x	1	x	1	0	1	1	x	0	x	x	0		
S_2	0	1	1	x	x	x	0	0	0	0	x	x	1	x	1	0	1
				x	x	x	0	0	0	0	x	x	1	x	1		
S_3	1	0	1	x	x	x	1	1	1	x	0	1	x	x	0	0	0
				x	x	x	1	1	1	x	0	1	x	x	0		
S_4	1	1	1	x	x	0	1	1	1	x	0	x	0	x	0	1	0
				x	x	1	0	1	1	x	1	x	0	x	0		

2.3 Logic Implementation

Eight Karnaugh maps were used to produce Boolean equations and implement the logic circuit for the 3-bit controller.

READY $= f(Q_2, Q_1, Q_0)$:

	00	01	11	10
0	1	1	1	x
1	x	0	0	x

READY = $\overline{Q_2}$

BUSY $= f(Q_2, Q_1, Q_0)$:

	00	01	11	10
0	0	0	0	x
1	x	0	1	x

BUSY = $Q_2.Q_1$

$J_2 = f(Q_2, Q_1, Q_0)$:

	00	01	11	10
0	0	RET	0	x
1	x	x	x	x

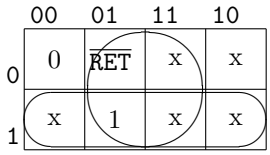
$J_2 = \overline{Q_1}\,.\,Q_0.\,RET$

$K_2 = f(Q_2, Q_1, Q_0)$:

	00	01	11	10
0	x	x	x	x
1	x	0	WAIT	x

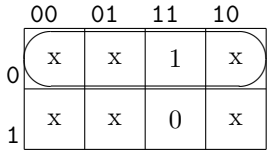
$K_2 = Q_1.\,WAIT$

$J_1 = f(Q_2, Q_1, Q_0)$:



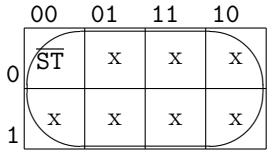
$J_1 = Q_0 \,.\, \overline{RET} + Q_2$

$K_1 = f(Q_2, Q_1, Q_0)$:



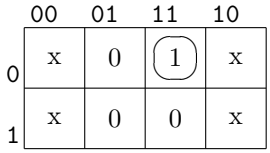
$K_1 = \overline{Q_2}$

$J_0 = f(Q_2, Q_1, Q_0)$:



$J_0 = \overline{ST}$

$K_0 = f(Q_2, Q_1, Q_0)$:



$K_0 = \overline{Q_2} \,.\, Q_1 . Q_0$

2.4 Software Simulation

Figure 2.4 illustrates the circuit and Figure 2.5 shows the simulation results with ideal gates and flip-flops (no propagation delay).

In the second stage of simulation 10ns delay had been applied for gates and 20ns for flip-flops. The result is illustrated by the waveforms in Figure 2.6, which demonstrates that circuit is still operational with these propagation delay values and 1MHz clock frequency. Figure 2.7 is to illustrate the effect of asynchronous reset and Figure 2.8 shows all possible values of control inputs and how the state outputs are being affected.

2.4.1 Timing Constraints

The clock period below 50ns, which is the total propagation delay of the critical path (3 gates and one flip-flop), is unsafe for circuit operation. Though, it may still perform near to specification when the clock period is $1\frac{1}{2}$ times greater then 50ns (i.e. at 75ns). However the safest minimum clock period would be double the total propagation delay of the critical path.
Note, that this simulation had only implemented a basic delay for the flip-flops and real flip-flops are know to have 3 timing values (setup, hold and clock-to-output, commonly referred to as t_S, t_H and t_{CTO}). Second important presumption taken for the simulation of the above circuit is that the AND-gates in use are available with 3 inputs. If only 2-input gates to be used, then the critical path for feed-back loops from Q_2 to K_0 and Q_1 to J_2 would increase by one extra gate (which adds extra 10ns to the total delay).

Since this controller can operate at 10MHz, it can be used in a variety of applications, such as for example a low sample rate analogue-to-digital converter (ADC), suitable for sensor interfacing.

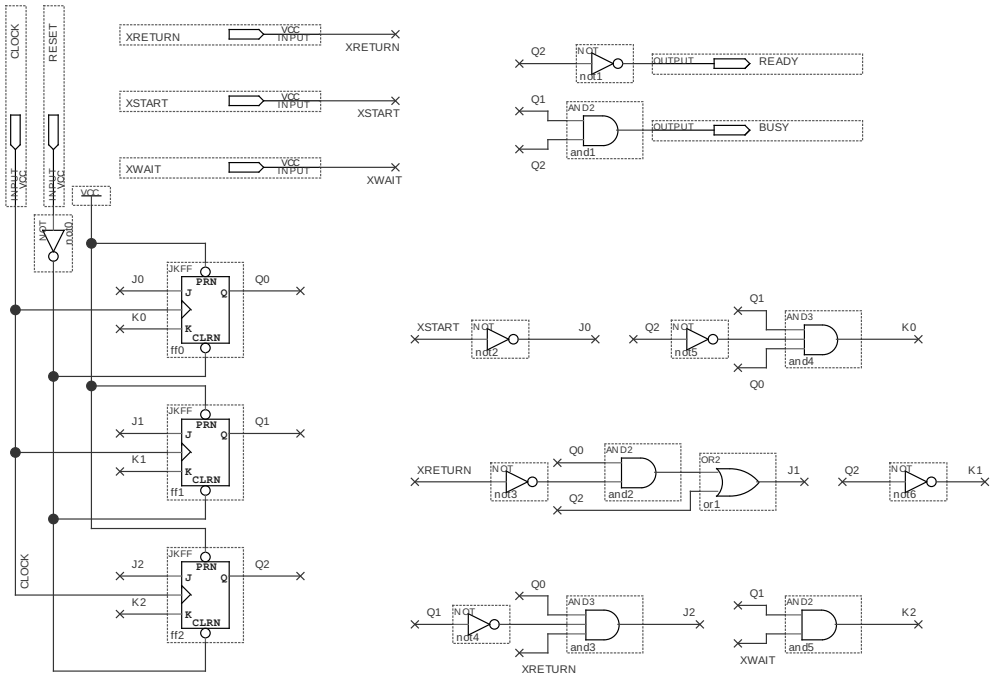


Figure 2.4: 3-bit Controller Circuit Diagram

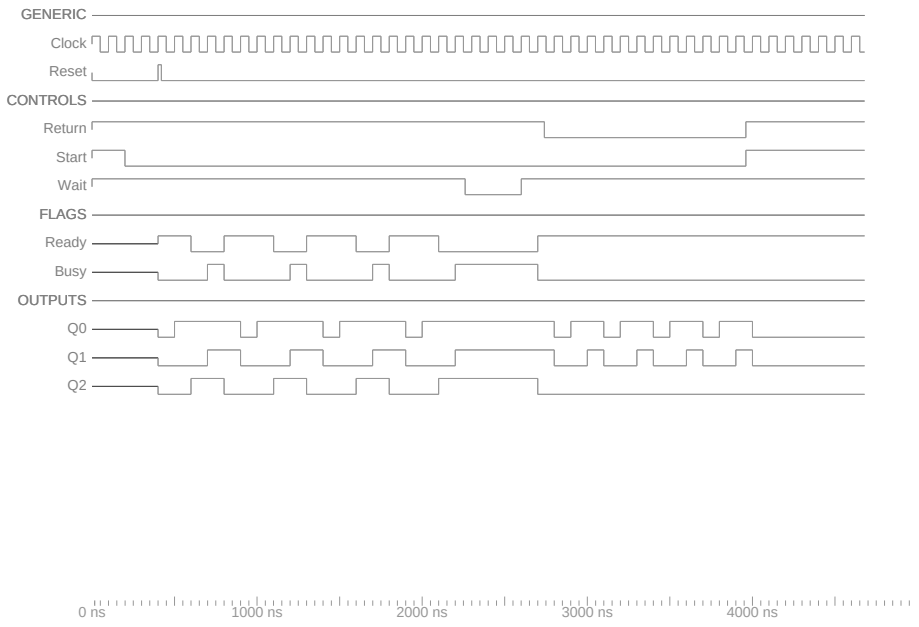


Figure 2.5: Simulation results (ideal components, zero-delay & 10MHz clock)

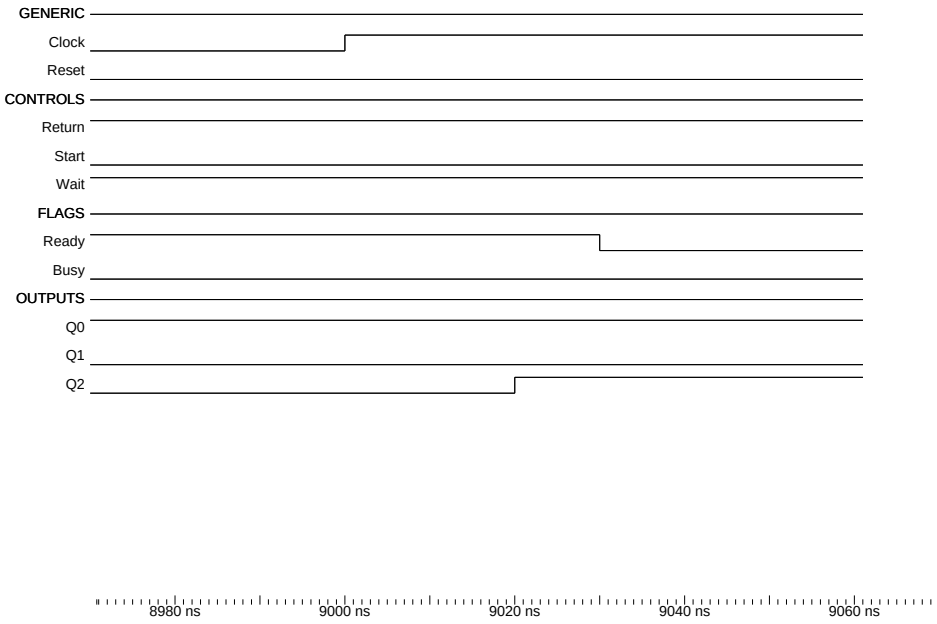


Figure 2.6: Simulation results (close-up to illustrate effect of delays at 1MHz)

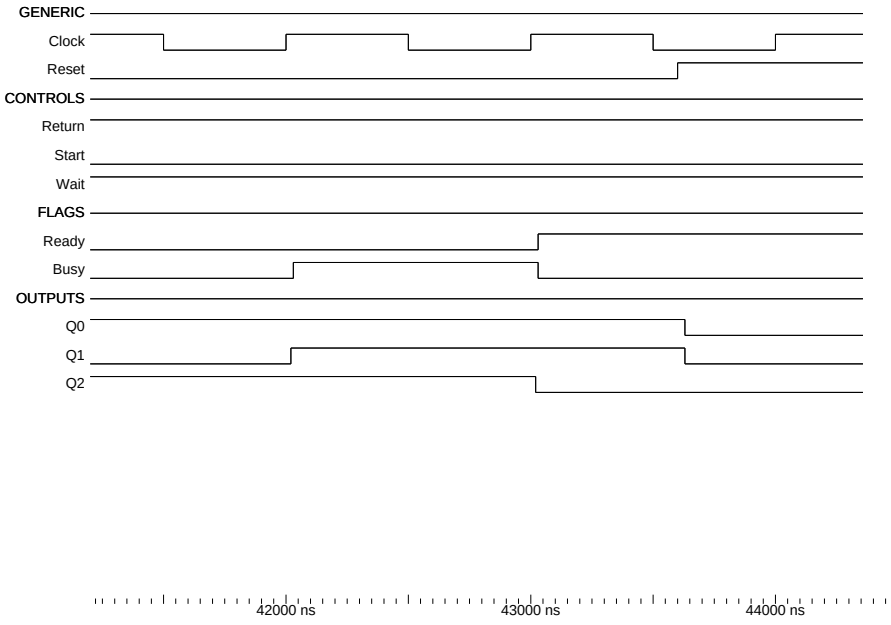


Figure 2.7: Simulation results (close-up to illustrate asynchronous reset)

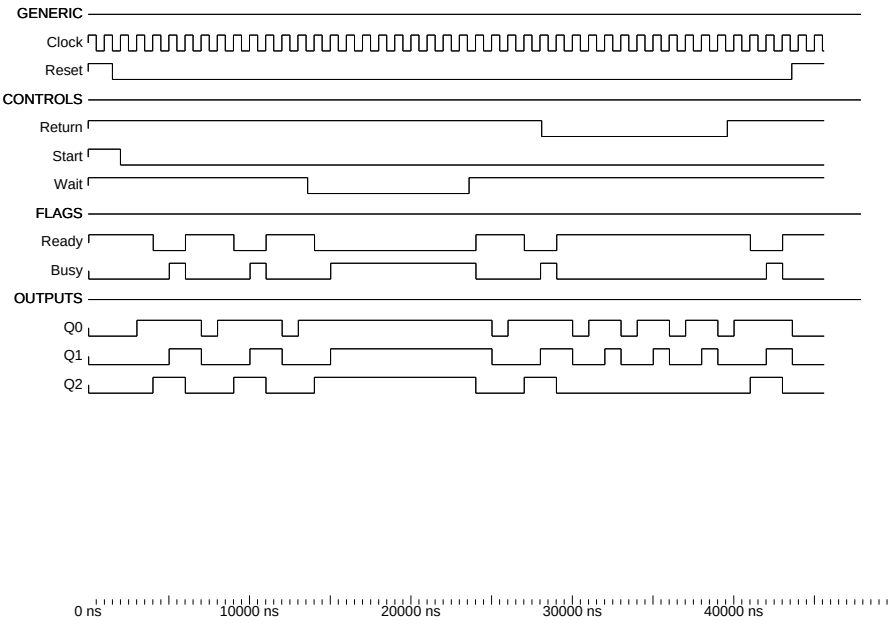


Figure 2.8: Simulation results (full view)

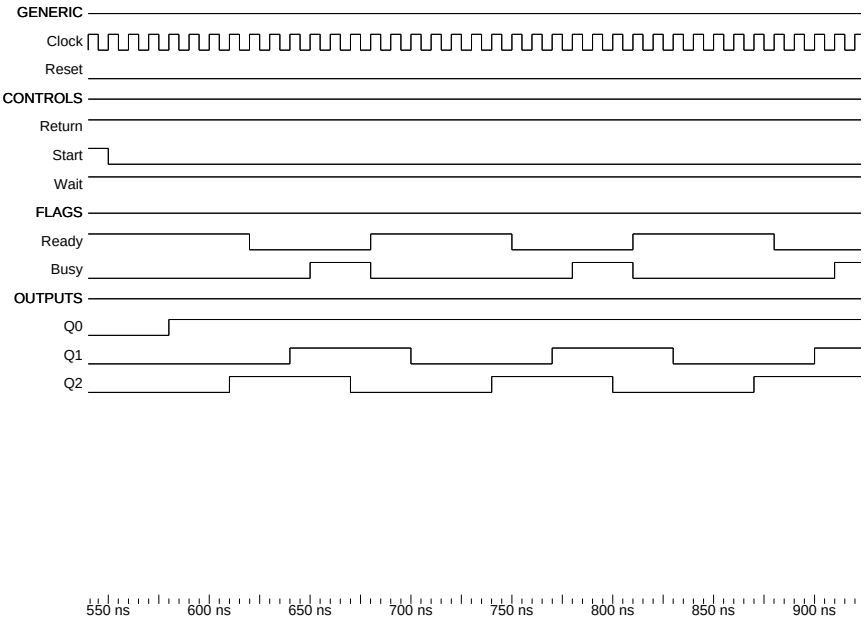


Figure 2.9: Simulation results (fast clock period of 10ns)

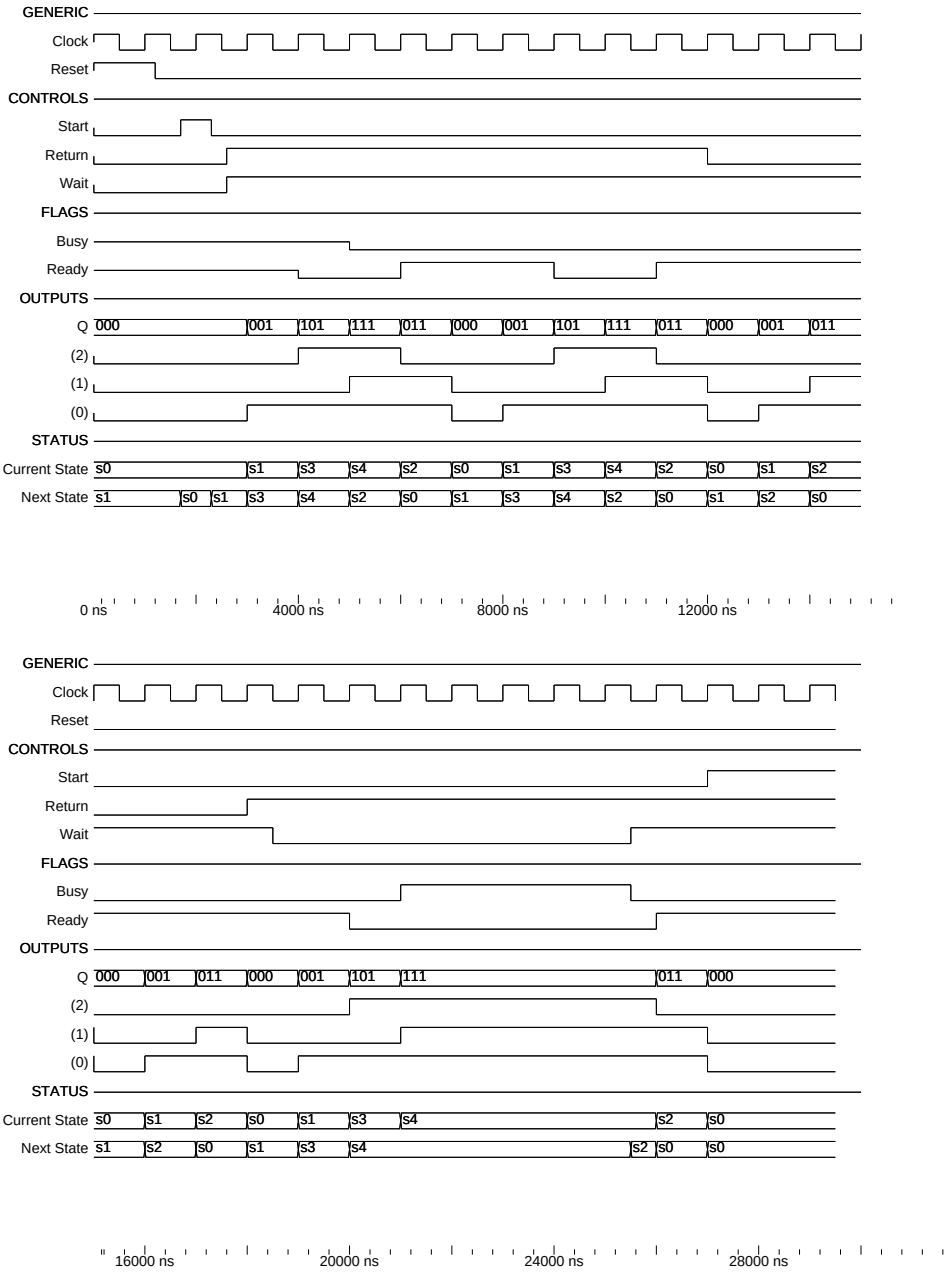


Figure 2.10: VHDL FSM simulation (1MHz clock frequency)

2.4.2 VHDL

The VHDL code prototype for this part of case study has also been implemented. The architecture is defined in terms of states and transitions, it uses two concurrent processes. One process implements the asynchronous reset function with clock sensing and other process is the finite state machine for the main controller function.

The code listed below has been verified in *Altera Quartus II* design package. The reference example was take from "VHDL Tutorial: Learn by Example" by Weijun Zhang[4]. However, for the purpose of simulation and proof of the above boolean equations a circuit diagram (Figure 2.4) had been implemented using graphical circuit entry in *Altera* software and then via conversion to VHDL it had been simulated in *ModelSim (Altera Edition)*. The code was then amended to simulate the gate and flip-flop propagation delay effects. The waveforms in Figure 2.10 are for the FSM code listed below.

```
library IEEE;
    use IEEE.STD_LOGIC_1164.all;
-- Refernce: http://esd.cs.ucr.edu/labs/tutorial/fsm.vhd

Entity controller1 is
  PORT (
    CLOCK:        in      std_Logic; --rising-edge
    RESET:        in      std_Logic; --active-hight
    XRUN:         in      std_Logic; --active-low
    XRET:         in      std_Logic; --active-low
    XWAIT:        in      std_Logic; --active-low
    XBUSY:        out     std_Logic;
    XREADY:       out     std_Logic;
    XOUT:         out     std_Logic_Vector ( 2 DOWNTO 0 ));
end controller1;

--- Finite state machine architecture
Architecture FSM of controller1 is
--- Define state names and signals
  type state_type is ( S0, S1, S2, S3, S4 );
  signal next_state, current_state: state_type;
  signal VOUT: std_Logic_Vector ( 2 DOWNTO 0 );
  signal VREADY, VBUSY: std_Logic;
begin
---- Control process for RESET and CLOCK input events
events: PROCESS ( CLOCK, RESET ) begin
    If ( RESET = '1' ) then
            current_state <= S0;
    elsif ( RISING_EDGE(CLOCK) ) then
            current_state <= next_state;
    end if;
  end process; -- events
```

[4] *Weijun Zhang "VHDL Tutorial: Learn by Example"*
http://esd.cs.ucr.edu/labs/tutorial/

```
---- Main controller process implementation
states: PROCESS ( current_state, XRUN, XRET, XWAIT )
   begin
  case current_state is
----- State #0 depends on XRUN input
        when S0 => VOUT <= "000";
              If XRUN = '1' then
                 next_state <= S0;
              elsif XRUN = '0' then
                 next_state <= S1;
              end if;
----- State #1 depends on XRET input
        when S1 => VOUT <= "001";
              If XRET = '0' then
                 next_state <= S2;
              elsif XRET = '1' then
                 next_state <= S3;
              end if;
----- State #2 asserts READY flag high
        when S2 => VOUT <= "011";
                 VREADY <= '1';
                 next_state <= S0;
----- State #3 asserts READY flag low
        when S3 => VOUT <= "101";
                 VREADY <= '0';
                 next_state <= S4;
----- State #4 keeps BUSY asserted if WAIT is high
        when S4 => VOUT <= "111";
              VBUSY <= '1';
              If XWAIT = '0' then
                 next_state <= S4;
              elsif XWAIT = '1' then
                 VBUSY <= '0';
                 next_state <= S2;
              end if;
----- State #0 is the default
        when OTHERS =>
                 VOUT <= "000";
                 next_state <= S0;
  end case;
  end process; -- states

---- Assign internal signals to pins
  XOUT   <=   VOUT;
  XREADY <=  VREADY;
  XBUSY  <=   VBUSY;
end FSM;
```

Listing 2.2: *VHDL implementation of 3-bit controller using two processes*