Sequential Circuits

Topics

- Clocked synchronous state-machine design
- Basic sequential blocks: registers, counters, shift registers
- Implementation and simulation with Quartus Prime

Problems

1. *[Paper and pencil + Quartus Prime].* Design a sequence detector, according to Mealy's model, whose output, `y`, is '1' whenever input sequence "1101" occurs. Overlapped sequences are allowed. An example is given below, where the `x` input values are received one at each clock cycle (read them from left to right):

x	0	1	0	1	1	1	0	1	1	0	1	0	1	1
y	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Create a new project named "SeqDet1101" in *Quartus Prime* software. Create a new file for a logic diagram called "SeqDet1101.bdf" to implement detector based on logic gates and D flip-flops (use `dff` component in Quartus library). Perform functional simulation and check whether the detector works correctly for the input sequence given above.

2. *[Paper and pencil + Quartus Prime].* Design a sequential circuit that detects 5-bit long input sequences which start with "11" and contain exactly 3 "1"s. The circuit should work in such a way that once two initial "1"s are detected, the sequence is parsed to the end (with or without success), i.e. the following sequence can only start after three more bits are received. An example is given below, where the `x` input values are received one at each clock cycle (read them from left to right):

x	1	0	0	1	1	0	1	1	0	0	1	1	0	1	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Create a new project named "SeqDet3in5" in *Quartus Prime* software. Create a new file for a logic diagram called "SeqDet3in5.bdf" to implement detector based on logic gates and D flip-flops (use `dff` component in Quartus library). Perform functional simulation and check whether the detector works correctly for the input sequence given above.

3. *[Paper and pencil + Quartus Prime].* Design and implement a synchronous sequential circuit, according to Moore's model, that performs the arithmetic negation of a two's complement number of arbitrary length, that enters the circuit starting with its least significant bit. Admit that the circuit has a synchronous active-low reset input. An example is given below, where the `x` input values are received one at each clock cycle (read them from left to right):

```
x 0 1 0 1
y 0 1 1 0
```

The inserted number, represented in two's complement is 1010 (-6_{10}) and its calculated arithmetic negation is 0110 (6_{10}).

Create a new project named "SerialNeg" in *Quartus Prime* software. Create a new file for a logic diagram called "SerialNeg.bdf" to implement the circuit based on logic gates and D flip-flops (use `dff` component in Quartus library). Perform functional simulation and check whether the circuit works correctly for a number of input sequences.

4. *[Paper and pencil + Quartus Prime].* Design a 3-bit binary counter with a Gray code counting sequence. Use D flip-flops with asynchronous reset. Create a new project named "GrayCnt3" in *Quartus Prime* software. Create a new file for a logic diagram called "GrayCnt3.bdf" to implement the counter based on logic gates and D flip-flops. Perform functional simulation and check the counting sequence.

5. *[Paper and pencil + Quartus Prime].* Based on the component 74x163 (a synchronous 4-bit binary counter with active-low load and clear inputs) build a circuit that follows the indicated counting sequence: (4, 5, 6, 7, 8, 9, 10, 11, 12), (4, 5, 6, 7, 8, 9, 10, 11, 12), ... Create a new project named "SpecialCount4_12" in *Quartus Prime* software. Create a new file for a logic diagram called "SpecialCount4.bdf" to implement the counter based on logic gates and 74x163 synchronous counter (this is located in library maxplus2 and is named `74163`). Perform functional simulation and check the counting sequence.

6. *[Paper and pencil + Quartus Prime].* Based on the component 74x163 (a synchronous 4-bit binary counter with active-low load and clear inputs) build a circuit that follows the indicated counting sequence: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), ... Create a new project named "SpecialCount0_9" in *Quartus Prime* software. Create a new file for a logic diagram called "SpecialCount0.bdf" to implement the counter based on logic gates and 74x163 synchronous counter (this is located in library maxplus2 and is named `74163`). Perform functional simulation and check the counting sequence.

7. *[Paper and pencil + Quartus Prime].* Based on the 74x163 components design a mod-64 counter (a counter which counts from 0 to 63, recurrently). Determine the maximum operating frequency of the circuit taking into account the following timing specifications:

a. the flip-flops that make up the counter: t_{setup} = 10 ns, t_{hold} = 4 ns, t_{pHL} = 20 ns, t_{pLH} = 15 ns;
b. delay time of an elementary logic gate (if used): t_{gate} = 5 ns.

Create a new project named "Mod64Counter" in *Quartus Prime* software. Create a new file for a logic diagram called "Mod64Counter.bdf" to implement the counter based on 74x163 synchronous counters. Perform functional simulation and check the counting sequence.

8. *[Paper and pencil + Quartus Prime].* The circuit in Fig. 1a) is a 4-bit shift register, which shifts left (i.e. Q0 → Q3) and has a synchronous active-low load input. Design this circuit with logic gates and D flip-flops.
Create a new project named "CounterShiftReg" in *Quartus Prime* software. Create a new file for a logic diagram called "ShiftReg4.bdf" to implement the shift register. Perform functional simulation.
In Fig. 1b) the circuit is used as a special counter. Determine the state diagram of the counter and check your conclusions with simulation in Quartus Prime.

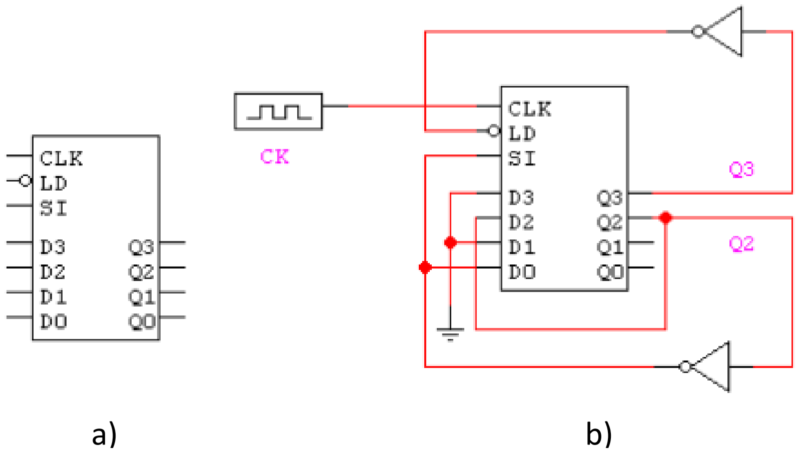

Fig. 1 – a) – a 4-bit shift register; b) – application of the shift register in a counting circuit.

9. *[Paper and pencil + Quartus Prime].* Design a delay line of length `N`, where `N` ranges from 1 to 16. The desired delay is specified with four inputs: `A3 A2 A1 A0` containing the unsigned binary representation of `N`-1. You may only use 4-bit shift registers (similar to those shown in Fig. 1) and multiplexers 4:1. Try to minimize the number of used components.
Hint: To minimize the number of blocks, note that (`A3 A2 A1 A0`) = (`A3 A2`) × 4 + `A1 A0`.
Create a new project named "DelayLine" in *Quartus Prime* software. Create a new file for a logic diagram called "DelayLine.bdf" to implement the delay line. You may make use of 74194 (4-bit bidirectional, parallel-in, parallel-out shift register) and 74153 (a dual 4:1 multiplexer) components available. Perform functional simulation of the circuit.

10. *[Paper and pencil + Quartus Prime]*. In lab. 8 you designed a 4-bit ripple-carry adder/subtractor, which is an example of an *iterative circuit*. Your adder/subtractor is composed of four identical modules, each of which has both primary inputs (`a` and `b`) and outputs (`s`) and cascading inputs (`cin`) and outputs (`cout`).

As you know, the function of an `n`-module iterative circuit can be performed by a sequential circuit that uses just one copy of the module but requires `n` steps (clock ticks) to obtain the result. Design such a sequential circuit that would act as an adder. Please note, that a serial binary adder circuit for addends of any length can be constructed from a full adder and a D flip-flop. The flip-flop, which stores the carry between successive bits of the addition, is cleared to 0 at reset. Addend bits are presented serially on the `a` and `b` inputs of the adder, starting with the least significant bits, and sum bits appear on `s` output in the same order.

Create a new project named "SerialAdder" in *Quartus Prime* software. Create a new file for a logic diagram called "SerialAdder.bdf" to implement the serial adder. Perform functional simulation of the circuit.