Coursework 2: Sequential Circuit Design using Quartus

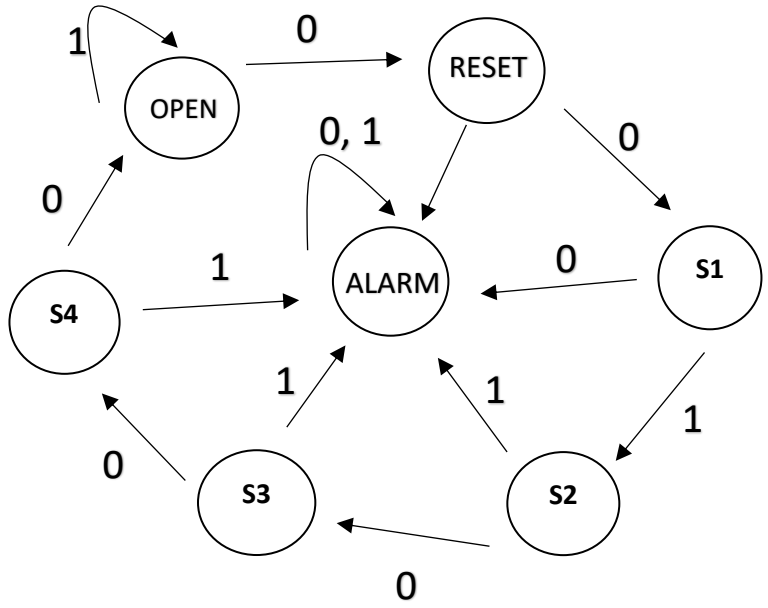
Department of Computing
Imperial College London

My unique number is **0007**. And my personal binary sequence number is **01000**.

Binary Sequence = Binary Equivalent of (1 + (7 mod 30))

= Binary Equivalent of 8

The state diagram



From the State Transition Diagram we will now generate The State Transition Table. There are 7 states. Therefore, we need a minimum of three flip- flops whose outputs are **Q2, Q1, Q0**. We have a free choice to assign the seven required states to the eight possible ones.

The State Transition Table

DATA	This State	Q2	Q1	Q0	Next State	D2	D1	D0
0	Reset	0	0	0	S1	0	0	1
0	S1	0	0	1	Alarm	1	1	1
0	S2	0	1	0	S3	1	0	0
0		0	1	1	X	X	X	X
0	S3	1	0	0	S4	1	0	1
0	S4	1	0	1	Open	1	1	0
0	Open	1	1	0	Reset	0	0	0
0	Alarm	1	1	1	Alarm	1	1	1
1	Reset	0	0	0	Alarm	1	1	1
1	S1	0	0	1	S2	0	1	0
1	S2	0	1	0	Alarm	1	1	1
1		0	1	1	X	X	X	X
1	S3	1	0	0	Alarm	1	1	1
1	S4	1	0	1	Alarm	1	1	1
1	Open	1	1	0	Open	0	0	0
1	Alarm	1	1	1	Alarm	1	1	1

The Karnaugh Maps

D2

Q_1Q_0

DATA Q_2	00	01	11	10
00	0	1	X	1
01	1	1	1	0
11	1	1	1	0
10	1	0	X	1

D1

DATA Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	X	0
01	0	1	1	0
11	1	1	1	0
10	1	1	X	1

D0

DATA Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	X	0
01	1	0	1	0
11	1	1	1	0
10	1	0	X	1

For **D2** we end up with two terms with three literals and three terms with two literals. The minimized expression is:

D2 = DATA'.Q0 + Q2.Q'1 + Q1.Q0 + DATA.Q'1.Q'0 + Q2'.Q1.Q'0
= Q0.(DATA' + Q1) + Q2.Q'1 + DATA.Q'1.Q'0 + Q2'.Q1.Q'0

For **D1** we end up with one term with one literal and one term with two and one term with three literals. The minimized expression is:

D1 = Q0 + DATA.Q'2 + DATA.Q'1.Q'0

We will not simplify that expression as the min-terms colored in red repeat.

For **D0** we end up with two terms with two literals and two terms with three literals. The minimized expression is:

D0 = Q'1.Q'0 + Q1Q0 + DATA'.Q'2.Q0 + DATA.Q2.Q0 + DATA.Q'2.Q'0
= Q0 XNOR Q1 + (DATA XNOR Q2).Q0 + DATA.Q'2.Q'0

We can now enter the true values of the “don’t cares” into the Karnaugh maps. Any don’t care within a circle will be a 1 and any don’t care outside all the circles will be a 0.

D2

DATA Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	1	1
01	1	1	1	0
11	1	1	1	0
10	1	0	0	1

D1

DATA Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	1	0
10	1	1	1	1

D0

DATA Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	X	0
01	1	0	1	0
11	1	1	1	0
10	1	0	X	1

The Transition table without the “don’t cares” can now be constructed from these new Karnaugh Maps.

DATA	This State	Q2	Q1	Q0	Next State	D2	D1	D0
0	Reset	0	0	0	S1	0	0	1
0	S1	0	0	1	Alarm	1	1	1
0	S2	0	1	0	S3	1	0	0
0		0	1	1	Alarm	1	1	1
0	S3	1	0	0	S4	1	0	1
0	S4	1	0	1	Open	1	1	0
0	Open	1	1	0	Reset	0	0	0
0	Alarm	1	1	1	Alarm	1	1	1
1	Reset	0	0	0	Alarm	1	1	1
1	S1	0	0	1	S2	0	1	0
1	S2	0	1	0	Alarm	1	1	1
1		0	1	1	Alarm	1	1	1
1	S3	1	0	0	Alarm	1	1	1
1	S4	1	0	1	Alarm	1	1	1
1	Open	1	1	0	Open	0	0	0
1	Alarm	1	1	1	Alarm	1	1	1

The output logic:

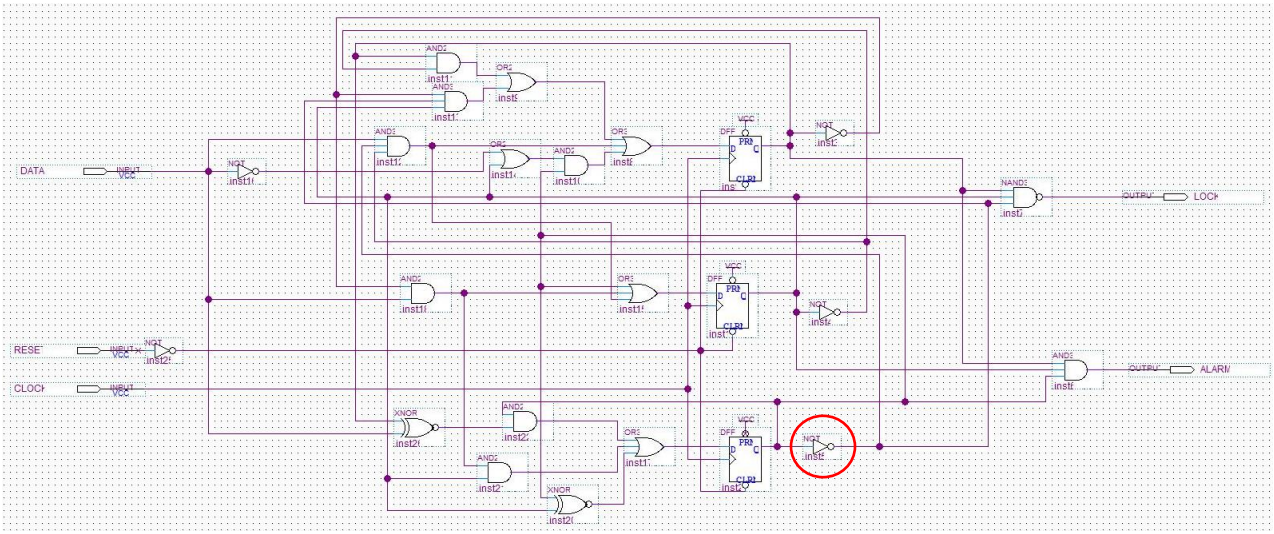
Lock = S1 + S2 + S3 + S4 + Reset + Alarm

= Open’

Lock = (Q2.Q1.Q’0)’

Alarm = Q2.Q1.Q0

The circuit implementation in Quartus ||



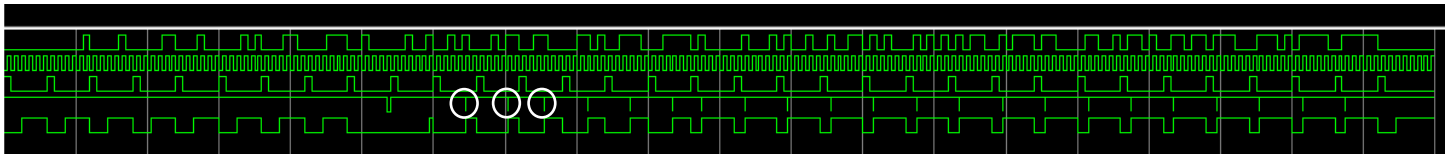
We used five AND2, three AND3, one NAND3, two OR2, three OR3, two XNOR2, five NOT gates and three DFF.

The total area = 5*27 + 3*34 +28 + 2*28 + 3*34 + 2*44 + 5*15 + 3*100

= 886

We can still simplify it and reduce the silicon areas by converting some of the AND and OR gates into NOR and NAND.

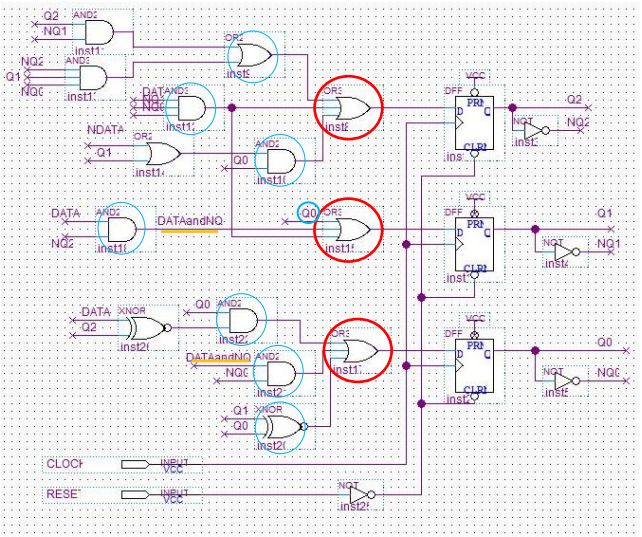
Zoomed out wave forms



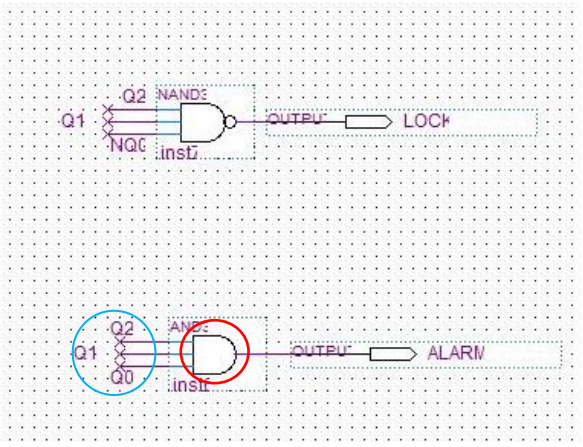
We have a number of spikes(circled on the wave simulation) because the different signals to the LOCK output traverse a different number of gates. One of the input goes through an inverter(circled in red in the above circuit) so there is a slight time delay. We can solve that by redesigning the output circuit so that the number of gates is the same for every signal.

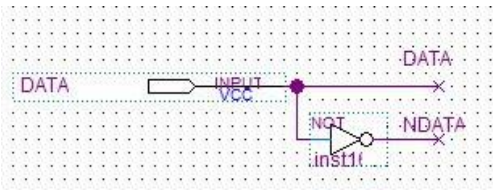
We can implement the circuit in Quartus II by naming the wires which will make it more readable and easy to follow. So we separate The State Sequencing Logic and The Output Logic.

The State Sequencing Logic



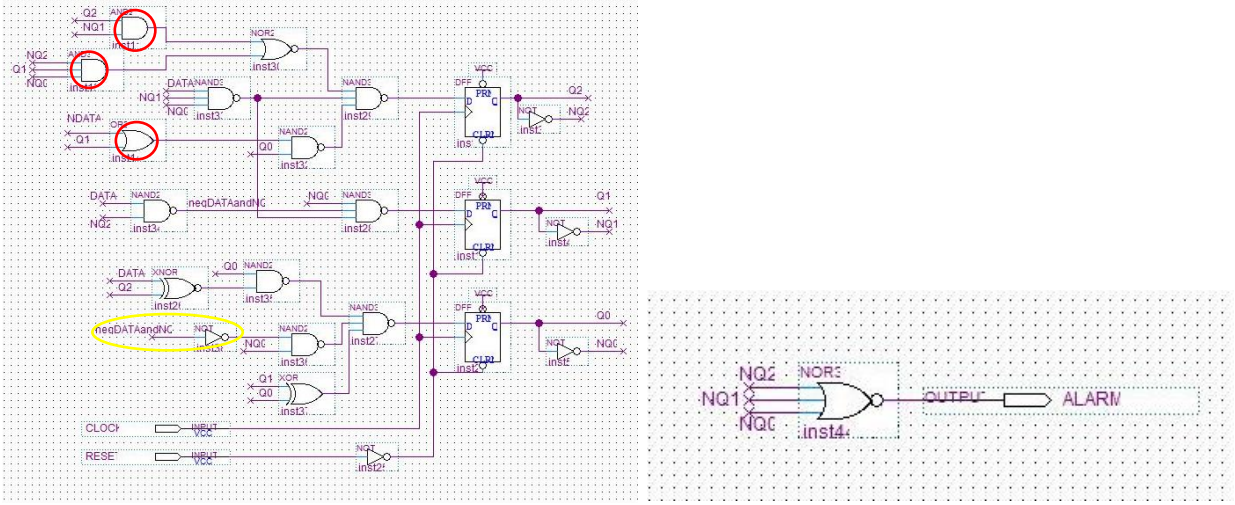
The output logic





We can know convert the OR gates and the AND gate circled in red into NAND/NOR gates by negating their inputs which are circled in blue.

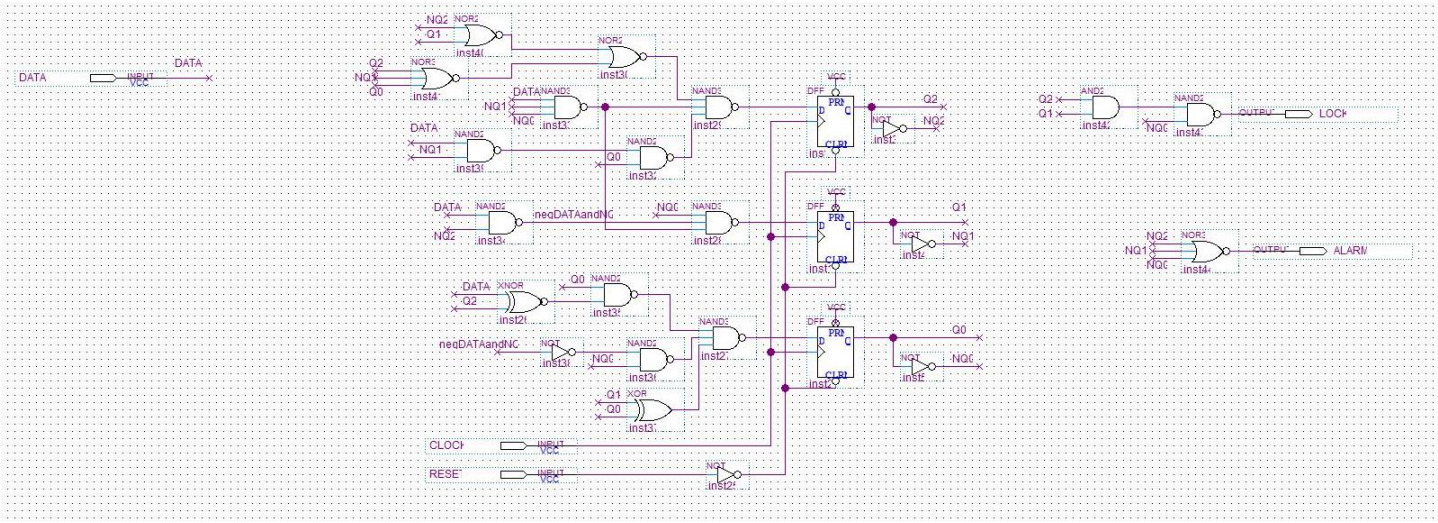
After the first simplification:



We negated the AND gate with DATA and NQ2 as inputs but its output is an input to one of the NAND2 gates (underlined in orange). That is why we need to add an inverter (yellow circle) so that we negate the negated input and not change its logic.

We can now further simplify the AND gates and the OR gate circled in red by negating their inputs. After doing that we see that there is no longer an input NDATA(DATA') so we can omit one inverter. We also need to fix the spikes which will be done by adding an AND gate to the output logic of the LOCK.

Final Circuit



Now we used five inverters, one AND2, six NAND2, four NAND3, two NOR2, two NOR3, one XOR2, one XNOR2 and three DFF.

Final silicon area

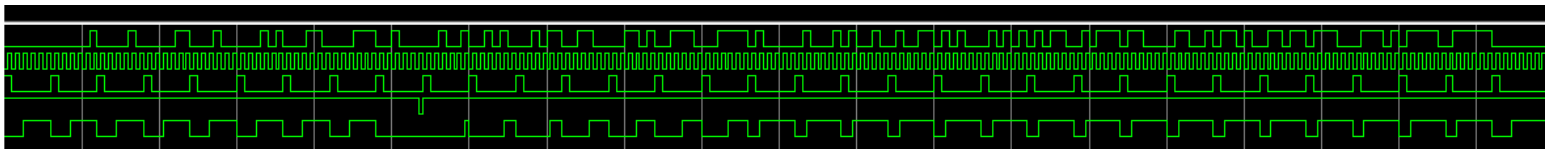
= 5*15 + 27 + 6*20 + 4*28 + 2*19 + 2*28 + 44 + 44 + 3*100

= 816

So the difference before and after the simplifiation in silicon areas is

886 – 816 = 70

Final zoomed out wave forms



Zoomed in wave forms