M1 Info – ARC - Lecture 6

Olivier HUREAU - Groupe 3

08/04/2020

1 Exercice 1

1.1 Step one

Even if with a one hot encoding, the truth tables for the transition and output functions is still the same such as :

c	q	q'
0	init	init
0	Sone	Sone
0	Stwo	Stwo
0	Sthree	Sthree
1	init	Sone
1	Sone	Stwo
1	Stwo	Sthree
1	Sthree	init

(a) transition

state q	Z_1 Z_0
init	00
Sone	01
Stwo	10
Sthree	11

(b) output functions

Figure 1: truth tables for the transition and output functions

1.2 Step 2

The 2-bit counters is composed of 4 states : {Init, Sone, Stwo, Sthree}.
As we know : "with One hot encoding nb_states bits are needed".
Then here we need 4 bits such as {Q0 ; Q1 ; Q2 ; Q3} for the encoding.
Thus we have :

- Init : 0001
- Sone : 0010
- Stwo : 0100
- Sthree : 1000

1.3 Step 3

c	q_3 q_2 q_1 q_0	q'_3 q'_2 q'_1 q'_0
0	0001	0001
0	0010	0010
0	0100	0100
0	1000	1000
1	0001	0010
1	0010	0100
1	0100	1000
1	1000	0001

(a) transition

q_3 q_2 q_1 q_0	Z_1 Z_0
0001	00
0010	01
0100	10
1000	11

(b) output functions

Figure 2: truth tables for the transition and output functions with one hot encoding

Boolean expressions for t and f are derived from the truth tables (and simplified):

- q'_0 = not(c).not(q3).not(q2).not(q1).q0 + c.q3.not(q2).not(q1).not(q0)= not(c).q0 + c.q3
- q'_1 = not(c).not(q3).not(q2).q1.not(q0) + c.not(q3).not(q2).not(q1).q0 = not(c).q1 + c.q0
- q'_2 = not(c).not(q3).q2.not(q1).not(q0) + c.not(q3).not(q2).q1.not(q0) = not(c).q2 + c.q1
- q'_3 = not(c).q3.not(q2).not(q1).not(q0) + c.not(q3).not(q2).not(q1).q0 = not(c).q3 + c.q0
- Z_0 = not(q3).not(q2).q1.not(q0) + q3.not(q2).not(q1).not(q0) = q1 + q3
- Z_1 = not(q3).q2.not(q1).not(q0) + q3.not(q2).not(q1).not(q0) = q2 + q3

2 Exercice 2

Here, we have 8 states so the one hot encoding will be on 8 bits Assume that the following binary encoding has been used: AT1a=00000001, AT1=00000010, B12=00000100, AT2=0000 1000, AT2a=00010000, B23=00100000, AT3=01000000, AT3a=10000000

And that the signal identifiers of the 8 resulting flips-flops are S7, S6, S5, S4, S3, S2, S1, and S0.

Then, the adaptation of the PSL assertions is:

```
always(r1 -> eventually!(S7=0 and S6=0 and S5=0 and S4=0 and S3=0 and S2=0 and S1=1 and S0=0 and))
always(r2 -> eventually!(S7=0 and S6=0 and S5=0 and S4=0 and S3=1 and S2=0 and S1=0 and S0=0 and))
always(r3 -> eventually!(S7=0 and S6=1 and S5=0 and S4=0 and S3=0 and S2=0 and S1=0 and S0=0 and))
```

3 Exercice 3

3.1 Bulding Truth table

A	B	q	q'
0	0	init	init
0	0	S1	init
0	1	init	init
0	1	S1	S1
1	0	init	init
1	0	S1	S1
1	1	init	S1
1	1	S1	S1

Figure 3: Transition truth table

As the aumtomaton is a mealy machine, outputs depends on the state and the value of A and B. Thus we have

A	B	q	Z_0
0	0	init	0
0	0	S1	1
0	1	init	1
0	1	S1	0
1	0	init	1
1	0	S1	0
1	1	init	0
1	1	S1	1

Figure 4: Output truth table

3.2 Choice encoding

As we are working with FPGA and the One hot encoding is more appropriate in the context of FPGA synthesis. The state encoding will be One hot. We have 2 states then 2 bits are needed.
That is :

- init = 01
- S1 = 10

3.3 Truth table with One hot encoding

:

A	B	q_1 q_0	q_1' q_0'
0	0	01	01
0	0	10	01
0	1	01	01
0	1	10	10
1	0	01	01
1	0	10	10
1	1	01	10
1	1	10	10

(a) transition

A	B	q_1 q_0	Z_0
0	0	01	0
0	0	10	1
0	1	01	1
0	1	10	0
1	0	01	1
1	0	10	0
1	1	01	0
1	1	10	1

(b) output functions

Figure 5: truth tables for the transition and output functions with One hot encoding

and Boolean expressions for t and f are derived from the truth tables (and simplified):

- Q0' = not(A).not(B).not(Q1).Q0 + not(A).not(B).Q1.not(Q0) + not(A).B.not(Q1).Q0 + A.not(B).not(Q1).Q0 = not(A).not(B) + not(A).B.not(Q1).Q0 + A.not(B).not(Q1).Q0
- Q1' : not(A).B.Q1.not(Q0) + A.not(B).Q1.not(Q0) + A.B.not(Q1).not(Q0) + A.B.not(Q1).not(Q0) = A.B + not(A).B.Q1.not(Q0) + A.not(B).Q1.not(Q0) +
- Z0 : not(A).not(B).Q1.not(Q0) + not(A).B.not(Q1).Q0 + A.not(B).Q1.not(Q0) + A.B.Q1.not(Q0)