M1 Info – ARC - TD5

Olivier HUREAU - Groupe 3

25/03/2020

1 Exercice 1

1.1 Assignements as three process

We can transform the three assignements as three process such as :

Process P1 :

```
Process(A)
Begin
Wait for 2 ns;
S1 <= not A;
End process;
```

Process P2 :

```
Process(S1,B)
Begin
Wait for 5 ns;
S2 <= S1 nand B;
End process;
```

Process P2 :

```
Process(S1, S2)
Begin
Wait for 4 ns;
SC <= S1 and S2;
End process;
```

1.2 Driver assignements

1.2.1 TC = 20 ns

A is updated to '1', the process P1 is resumed and the drivers are updates as follow :

```
S1 [ 1 / 22 ns |
S2 [
SC [
```

Next TC is 22

1.2.2 TC = 22 ns

S1 is updated to '1', then the transaction is removes from the drive.

S1 is part of the sensitivity list of P2 and P3 so P2 and P3 are released and the drivers are updates as follow

```
S1 [
S2 [ 0 / 27 ns |
SC [ 1 / 26 ns |
```

Next TC is 26

1.2.3 TC = 26 ns

SC is updated to '1', then the transaction is removes from the drive.
SC is no part of a sensitivity list, no process resumes.
the drivers are updates as follow

```
S1 [
S2 [ 0 / 27 ns |
SC [
```

Next TC is 27

1.2.4 TC = 27 ns

S2 is updated to '0', then the transaction is removes from the drive.
S2 is part of the sensitivity list of P3 so P3 resume
the drivers are updates as follow

```
S1 [
S2 [
SC [ 0 / 31 ns |
```

Next tc is 31

1.2.5 TC = 31 ns

SC is updated to '0', then the transaction is removes from the drive. SC is no part of a sensitivity list, no process resumes.
the drivers are updates as follow
S1 [S2 [SC [

1.2.6 All the drivers are empty

There is a stabilization, we stop here

2 Ecercice 2

2.1

The process equivalent to

```
x <= 1 after 5 ns, 2 after 10 ns, 1 after 15 ns, 0 after 20 ns;
```

is

```
Process
Begin
Wait for 5 ns;
x <= 1;
Wait for 5 ns;
x <= 5 ns;
Wait for 5 ns
x <= 1;
Wait for 5 ns;
x <= 0;
Wait;
end process
```

2.2

The sensitivity list is empty there (the assignments do not depends of others signals). Moreover, we do not want the process to loop and execute only one times.
Then the process does not resume.

3 Exercice 3

3.1 Assignements as three process

We can transform the three assignements as three process such as :
Process P1 :

```
Process(A, B)
Begin
Wait for 3ns;
X1 <= A or B;
End process;
```

Process P2 :

```
Process(X1)
Begin
X2 <= not X1
End process;
```

Process P3 :

```
Process(X2, C)
Begin
Wait for 4 ns;
Z <= X2 xor C;
End process;
```

3.1.1 TC = 22 ns

A is updated to '0' and C is updated to '1'. A is part of the sensitivity list of P1 and C is part of the sensitivity list of P3. So P3 and P1 resumes.
the drivers are updates as follow

```
X1 [ 0 / 25 ns
X2 [
Z  [ 1 / 26 ns
```

Next TC is 25

3.1.2 TC = 25 ns

X1 is updated to '0'. X1 is part of the sensitivity list of P2. So P2 resume.
the drivers are updates as follow

```
X1 [
X2 [ 1 / 25 ns
Z  [ 1 / 26 ns
```

Next TC is 25 + delta

3.1.3 TC = 25 ns + delta

X2 is updated to '1'. X2 is part of the sensitivity list of P3. So P3 resume.
the drivers are updates as follow

```
X1 [
X2 [
Z  [ 1 / 26 ns | 0 / 29 ns
```

Next TC is 26 ns

3.1.4 TC = 26 ns

Z is updated to '1', then the transaction is removes from the drive.
Z is no part of a sensitivity list, no process resumes.
the drivers are updates as follow

```
X1 [
X2 [
Z  [ 0 / 29 ns
```

Next TC is 29

3.1.5 TC = 29 ns

Z is updated to '0', then the transaction is removes from the drive.
Z is no part of a sensitivity list, no process resumes.
the drivers are updates as follow

```
X1 [
X2 [
Z  [
```

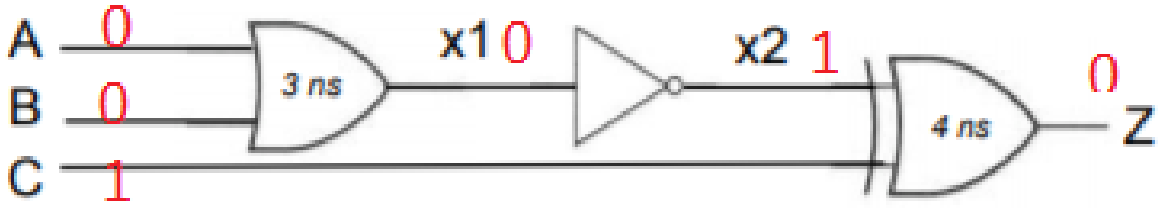
Next TC is 29

3.1.6 All the drivers are empty

There is a stabilization, we stop here

3.2 Results

At 29 ns, we reach the following configuration (A = '0' , B = '0', C = '1' , Z = '0' , X1 = '0', X2 = '1')



4 Exercise 4

4.1 Everytime a car wants to enter and the parking is not full, the input gate must open and must stay open until the car has come in or the timeout is reached.

We can rephrase as follow :

```
Everytime (
- A car want to enter
AND
- The Parking is not full
)

Then

Untill (
- !(Car Came out)
OR
- !(Timout reached)
)
The gate are open
```

4.1.1 PSL Forumula

```
Always (
(a car want to enter) and (parking is not full)
=> Gate Open untill! ( (timeout is reached) or (the car as come in) )
)
)
```

4.1.2 Give the corresponding PSL formula. Is it always satisfied? If not, enhance the formula to produce a version that is actually satisfied by this automaton

It does not satisfied the automaton beacause it must not have a ticket insserted to go to state OPEN_IN.

Then the formula to satisfied this automaton is :

```
Always (
(a car want to enter) and (parking is not full) and (no ticket is insserted)
=> Gate Open untill! ( (timeout is reached) or (the car as come in) )
)

)
```

4.2 Everytime a non-valid ticket is inserted, the output gate should not open

```
Always ( the ticket is non valid => the output gate stay closed )
```