Annex C – The commented code

Team 7 - 20/01/2012

% We have modeled the system to be controlled by 3 parallel controllers, 1 for High Chamber, 1 for Low Chamber and 1 for both the sluices.

% Each Chamber and Door has been denoted by numbers for ease while writing the code and reading it.

% Sluice1 (1) is the input sluice, Sluice2 (2) is the output sluice, Low Chamber is 3 , High Chamber is 4 for e.g. moveWaferOut4_3 denotes the action of

% HighVacuumchamber(4) to move the wafer towards LowVacuumChamber(3) and moveWaferIn4_3 is the action by LowVacuumChamber(3) to move wafer in coming from

% HighVacuumChamber(4). Both these actions have been synchronised, as well as the others of the same kind. Several communication actions have been used to

% communicate the state of the chambers and doors.

% What they communicate has been explained later in the comments.

% The system model diagram in the report includes the numbers which represent doors.

% Datatype for knowing the state of the wafer inside a chamber; indirectly also tells the number of wafers inside the chamber

%Processed = 1 processed wafer

%Unprocessed = 1 unprocessed wafer

%Both = 1 unprocessed wafer and 1 processed wafer

%Empty = No wafer

%the combinations (unprocessed,unprocessed) and (processed,processed) in LVC are not possible

sort Procstate = struct Processed | Unprocessed | Both | Empty;

% Vacuumlevel has been assumed to take 2 possible values in each chamber either the corresponding threshold value or any random value(after the door is opened)

sort Vacuumlevel = struct ThreshHigh | Threshlow | Random;

% All the actions used, description of each action can be found in the report submitted

act maintainVacuum : Vacuumlevel;

act acceptWafer1,% Sluice1 can accept an unprocessed wafer

openDoor1, % Open the door of sluice1

moveWaferIn0_1,% move the wafer into sluice1

closeDoor1, % Close the door of sluice1

openDoor3, % Open the 2nd door of Sluice1

openDoor4, % Open the low chamber - Sluice1 door;

moveWaferOut1_3,%Move the wafer out of sluice 1 towards low chamber

moveWaferIn1_3, % Move the incoming wafer from Sluice1 into the low chamber

closeDoor4, %close the low chamber - Sluice1 door

closeDoor3, % close the 2nd door of Sluice1

openDoor7, % open the low chamber - high Chamber door

openDoor8, % open the high chamber - low chamber door

moveWaferOut3_4,%move the wafer from low to high chamber (as seen from the low vaccum chamber perspective)

moveWaferIn3_4, %take the wafer from low to high chamber (as seen from the high vaccum chamber perspective)

closeDoor7, % close the low chamber - high Chamber door

closeDoor8, % close the high chamber - low chamber door

StartProcessing,% Start processing inside the wafer

EndProcessing,% End processing inside the wafer

moveWaferOut4_3,% move the wafer from High to low chamber

moveWaferIn4_3, % take the incoming wafer from high chamber into low chamber

openDoor6, % open the door of Low chamber- sluice2

openDoor5, % open the door of sluice 2 - low chamber

moveWaferOut3_2, % move the wafer out of the low chamber towards sluice 2

moveWaferIn3_2, % move the wafer into the sluice 2 from low chamber

closeDoor6, % close the door of Low chamber- sluice2

closeDoor5, % close the door of sluice 2 - low chamber

openDoor2, % open the exit door of Sluice 2

moveWaferOut2_0, % move the wafer out of sluice 2

closeDoor2, % close the exit door of sluice 2

move, % synchronizes actions to move wafer from High vacuum chamber to low vacuum chamber

move1, % synchronizes actions to move wafer from low chamber to sluice 2

move2, %synchronizes actions to move wafer from sluice 1 to low chamber

move3 % synchronizes actions to mave wafer from low chamber to high chamber

;

act send1B, %send action to communicate the state of Door (3) to low Vacuum process from Sluice process

send1B_1, %send action to coomunicate the state of Door (5) to low vacuum process from Sluice process

send2B, %send action to communicate the sate of Door(7) to High vacuum Process

receive2B, %receive state of Door(3) action in Low Vacuum process

receive2B_1, %receive state of Door (5) in low vacuum process

receive3B, %receive state of Door(7) in High Vacuum process

com, %synchronization action for send1B|receive2B

com2, %synchronization action for send1B_1|receive2B_1

com3 %synchronization action for send2B|receive3B

:Bool;

act receive3P, %receive state of low vacuum chamber action in High vacuum chamber

receive2P, %receive state of high vacuum chamber action in low vacuum chamber

send2P, %send action to communicate the state of low vacuum chamber to high vacuum process

send3P, %send action to communicate the state of high vacuum chamber to low vacuum process

receive1P, %receive state of low vacuum chamber action in sluice process

send2P_1, %send action to communicate the state of low chamber to sluice process

send3P_E, %send action to communicate the high vacuum process Empty state to low vacuum process

receive2P_E, %receive state of high vacuum chamber action in low vacuum process

send2P_1_P, %send action to communicate the state of low vacuum process to sluice process

receive1P_P, %receive state of low vacuum chamber in sluice process

com5, %synchronization action for send2P|receive3P

com7, %synchronization action for send2P_1_P|receive1P_P

com1, %synchronization action for send2P_1|receive1P

com6, %synchronization action for send3P_E|receive2P_E

com4 %synchronization action for send3P|receive2P

:Procstate;

% function to calculate next state after a wafer is moved in

map predState : Procstate#Procstate -> Procstate;

```
% function to calculate next state after a wafer is moved out

map prevState : Procstate#Procstate -> Procstate;


% equations to calculate next state after a wafer is moved in

eqn predState(Unprocessed,Processed)= Both;

eqn predState(Empty,Processed)= Processed;

eqn predState(Empty,Unprocessed) = Unprocessed;

eqn predState (Empty, Empty) = Empty;

eqn predState(Processed, Unprocessed)= Both;

eqn predState (Processed, Empty)= Processed;

eqn predState(Unprocessed, Empty) = Unprocessed;

eqn predState (Both, Empty) = Both;

eqn predState (Empty, Both) = Both;


% equations for calculating the next state if a wafer is moved out.

eqn prevState(Processed,Processed) = Empty;

eqn prevState(Both,Processed)= Unprocessed;

eqn prevState(Unprocessed,Unprocessed)= Empty;

eqn prevState(Both,Unprocessed)= Processed;

eqn prevState(Processed, Empty) = Processed;

eqn prevState(Unprocessed, Empty)= Unprocessed;

eqn prevState(Empty, Empty)= Empty;

eqn prevState(Both, Empty) = Both;

eqn prevState(Both, Both)= Empty;
```

proc

%HIGH VACUUM CHAMBER

HighVacuum (HighVacuumlevel: Vacuumlevel, Highstate : Procstate, HighDoor:Bool)

% maintainvacuum if vacuumlevel is not at the required level

= (HighVacuumlevel!= ThreshHigh)-> maintainVacuum(ThreshHigh).HighVacuum(HighVacuumlevel = ThreshHigh)

+

% if high chamber is empty,send it's state and if lowvacuum chamber has an unprocessed wafer and the door is open, open the door of the high chamber

(Highstate == Empty && HighVacuumlevel == ThreshHigh) -> send3P_E(Empty).sum LowHighDoorState:Bool . receive3B(LowHighDoorState) . (LowHigh DoorState == true) -> openDoor8.HighVacuum(HighDoor = true)

+

% if high chamber is empty and highdoor is open, move the wafer in

(Highstate == Empty && HighDoor == true && HighVacuumlevel == ThreshHigh) ->moveWaferIn3_4.closeDoor8.HighVacuum(Highstate = Unprocessed, HighVacuumlevel = Random, HighDoor = false)

+

% if high chamber has a processed wafer and the highdoor is closed and LowHigh door is open, open the high door, move the wafer out and close the door else stay in the same state.

(Highstate == Processed && HighDoor ==false && HighVacuumlevel == ThreshHigh) -> sum LowHighDoorState :Bool . receive3B(LowHighDoorState).(LowHighDoorState == true) ->(openDoor8.moveWaferOut4_3.closeDoor8.HighVacuum(Highstate=Empty, HighVacuumlevel=Random, HighDoor = false))<>HighVacuum(Highstate=Processed, HighVacuumlevel=ThreshHigh, HighDoor = false)

+

% if HVC has a unprocessed wafer inside, start it's processing, end the processing and send the current state to low chamber

% assuming that the wafer undergoes some processing inside the HVC

(Highstate == Unprocessed && HighVacuumlevel == ThreshHigh)->(StartProcessing . EndProcessing.send3P(Processed). HighVacuum(Highstate = Processed, HighDoor = false));

%LOW VACUUM CHAMBER PROCESS

proc

Low (Current_Lowstate : Procstate, LowVacuum : Vacuumlevel, LowHighDoorstate : Bool,LowSluice1Door :Bool) =

% if vacuumlevel is not what is required

(LowVacuum!= Threshlow)-> maintainVacuum(Threshlow). Low(LowVacuum = Threshlow)

+

% if there is no unprocessed wafer inside the LVC, Low High door and Low sluice 1 door are closed,sen d current state to sluice1, read the door of sluice 1 , if it is open, open Low sluice1 door ,move the wafer in and close the door

((Current_Lowstate == Empty || Current_Lowstate == Processed) && LowHighDoorstate == false && LowSluice1Door == false && LowVacuum == Threshlow) -> send2P_1_P(Current_Lowstate).sum Sluice1LowDoorstate:Bool . receive2B(Sluice1LowDoorstate) . (Sluice1LowDoorstate == true) -> openDoor4. moveWaferIn1_3. closeDoor4.Low(Current_Lowstate = predState(Current_Lowstate,Unprocessed), LowVacuum = Random,LowSluice1Door= false)<>(Low(Current_Lowstate = predState(Current_Lowstate, Empty),LowVacuum = Threshlow,LowSluice1Door= false))

+

% if LVC has a processed wafer inside, Low high door is closed, send it's state to sluice 2,read the sluice 2 low door state, if open, indicating that it can take a processed wafer, move the wafer out, close the door else stay in the same state.

((Current_Lowstate ==Processed | | Current_Lowstate == Both)&& LowHighDoorstate == false && LowVacuum == Threshlow) -> send2P_1(Current_Lowstate).sum Sluice2LowDoorstate:Bool . receive2B_1(Sluice2LowDoorstate) . (Sluice2LowDoorstate ==true) -> (openDoor6. moveWaferOut3_2. closeDoor6.Low(Current_Lowstate = prevState(Current_Lowstate,Processed), LowVacuum = Random))<>Low(Current_Lowstate = prevState(Current_Lowstate,Empty),LowVacuum = Threshlow)

+

% if there is no processed wafer inside the LVC , LowHigh door is closed,read HVC's state,if it has a processed wafer, (open the door else keep it closed), send the door state to high chamber

(Current_Lowstate != Processed && Current_Lowstate != Both && LowHighDoorstate == false && LowVacuum == Threshlow)->sum HighState:Procstate . receive2P(HighState).(HighState == Processed)->(openDoor7.send2B(true).Low(LowHighDoorstate = true))<>(send2B(false).Low(LowHighDoorstate = false, LowVacuum = Threshlow))

+

% if the Low High door is open and HVC has a processed wafer, move the wafer in, clos the low high door

(Current_Lowstate != Processed && Current_Lowstate != Both && LowHighDoorstate == true && LowVacuum == Threshlow)->moveWaferIn4_3.closeDoor7.Low(Current_Lowstate = predState(Current_Lowstate,Processed) , LowVacuum = Random, LowHighDoorstate = false)

+

% if LVC has a unprocessed wafer, low high door is closed,read the HVC's state, if empty(open the low high door) else (let it remain closed) and send the door state to highchamber.

((Current_Lowstate == Unprocessed | | Current_Lowstate == Both)&& LowHighDoorstate == false && LowVacuum == Threshlow) ->sum HighState:Procstate . receive2P_E(HighState).(HighState == Empty)->openDoor7.send2B(true).Low(LowHighDoorstate = true)<>send2B(false).Low(LowHighDoorstate = false)

+

% if the LVC has a unprocessed wafer and low high door is open move it into HVC , close the low high door

(LowHighDoorstate == true && Current_Lowstate != Empty && Current_Lowstate !=Processed && LowVacuum == Threshlow)-> moveWaferOut3_4.closeDoor7.Low(Current_Lowstate = prevState(Current_Lowstate,Unprocessed), LowVacuum = Random, LowHighDoorstate = false);

%SLUICE CONTROL PROCESS

proc

Sluices (UnprocessedAccept :Bool,Sluice1LowDoorState: Bool,Sluice2LowDoorState: Bool ,SluiceState: Procstate)

% If sluice 1 is ready to accept a unprocessed wafer, move it in

= ((UnprocessedAccept == true) -> acceptWafer1.openDoor1.moveWaferIn0_1.closeDoor1.Sluices(UnprocessedAccept = false,SluiceState=predState(SluiceState,Unprocessed)))

+

% if sluice 2 has a processed wafer, open door,move it out and close the exit door

((SluiceState == Processed | | SluiceState == Both) -
>openDoor2.moveWaferOut2_0.closeDoor2.Sluices(SluiceState=prevState(SluiceState,Processed)))

+

% if sluice 1 has a unprocessed wafer, read the state of LVC,if it doe s not have a unprocessed wafer,(open the Sluic1 low door and send it's state to LVC, move the wafer out towards LVC and close the Sluice1 Low door) else stay in the same state

((SluiceState == Unprocessed || SluiceState == Both) && Sluice1LowDoorState == false) ->sum Current_Lowstate :Procstate . receive1P_P(Current_Lowstate).(Current_Lowstate != Unprocessed && Current_Lowstate != Both)->openDoor3.send1B(true).(moveWaferOut1_3.closeDoor3.Sluices(UnprocessedAccept = true,Sluice1LowDoorState = false ,SluiceState=prevState(SluiceState,Unprocessed)))<>(Sluices(Sluice1LowDoorState = false))

+

% if sluice 2 does not have a wafer and LVC has a processed wafer, read the state of LVC, if it has a processed wafer, open door of sluice 2, send it's state to LVC

((SluiceState == Empty || SluiceState == Unprocessed) && Sluice2LowDoorState ==false) ->sum Current_Lowstate :Procstate . receive1P(Current_Lowstate).(Current_Lowstate == Processed || Current_Lowstate == Both) ->openDoor5.send1B_1(true).Sluices(Sluice2LowDoorState = true)

+

%if sluice2low door is open, move wafer in, close the Sluice 2 low door

((Sluice2LowDoorState == true) ->moveWaferIn3_2 . closeDoor5 .Sluices(SluiceState=predState(SluiceState,Processed), Sluice2LowDoorState = false));

init

% allow all the external actions and synchronised communication actions

allow({openDoor1,moveWaferIn0_1,closeDoor1,openDoor3,closeDoor3,openDoor5,closeDoor5,openDoor2,moveWaferOut2_0,closeDoor2,acceptWafer1,openDoor6,StartProcessing,EndProcessing,closeDoor6,closeDoor4,openDoor4,openDoor7,closeDoor7,openDoor8,closeDoor8,

move1,move,move2,move3,com,com1,com2,com3,com4,com5,com6,com7,maintainVacuum},

```
        comm({ moveWaferOut4_3|moveWaferIn4_3 -> move,
            moveWaferOut3_2|moveWaferIn3_2 ->move1,
            moveWaferOut1_3|moveWaferIn1_3 ->move2,
            moveWaferOut3_4|moveWaferIn3_4 ->move3,
            send1B|receive2B ->com,
            send1B_1|receive2B_1->com2,
            send2P|receive3P ->com5,
                                    send2P_1|receive1P->com1,
            send2B|receive3B ->com3,
            send3P|receive2P ->com4,
            send3P_E|receive2P_E->com6,
                        send2P_1_P|receive1P_P ->com7},
          Sluices(true, false,false, Empty)||
          Low(Empty, Random, false,false)||
          HighVacuum (Random, Empty, false)
    )


);
```