System Validation Project Report

Group 07

Ratnakar Madan, 4177916

Ravindra Aithal, 4179854

Alexandru Ionut Diaconescu, 4185927

Dan Avramescu, 4116496

Delft University of Technology

January 20, 2012

Group Email: system-validation@googlegroups.com

Individual emails: {ratnakar.madan@gmail.com, aithal.ravindra@gmail.com, alexandruionutdiaconescu@gmail.com, avramescu.dn@gmail.com}

Table of contents

*The annexes are attached as additional documents

Introduction

The goal of the project is to design a controller for a new generation of ASML wafer-stepper. A wafer-stepper is a device used in the processing of wafers. A wafer stepper exposes a small portion of the wafer, called a die. The area of that die is referred to as die size and this area can contain one or several chips.

The wafer stepper which has to be modeled allows the placement of components on wafers by using ultra violet light. Because of that, the wafer-stepper can function only in extreme vacuum. The system contains a low vacuum chamber with two sluices as entrance and exit for the wafers, and a high vacuum chamber. There are sensors to monitor the chambers and the doors. The low vacuum chamber can hold up to two wafers, while the high vacuum chamber can process maximum one wafer at a time.

The software design tasks are to provide the list of requirements to satisfy, the interactions for the system, the architecture of the structure of the system, the behavior of all controllers in the architecture and the verification of the system. The states inside the system can be monitored in order to know what actions to do.

There is a challenge for modeling the controller, since high precision has to be obtained while retaining sufficient robustness in the design. The assignment states that it also has to be mentioned how the controller is decomposed into parallel components and the internal actions used among the processes with the corresponding parameters they pass.

In our design, there are three parallel components having the description in the corresponding section of the report. Other challenges are related to the validity of the design, the technical aspects, the system maintenance and the verification of requirements.

This report contains the results of the project Team 7 made for the System Validation course. Each of the challenges mentioned above will be discussed in the following six sections. In section 1 we present the global requirements for the whole system. We describe in section 2, the interactions that are performed by our system. For the next section the global requirements were translated in the terms of the interactions from section 2. Further on, we document the three parallel controllers which make up the architecture which we designed in the fourth phase of our projects. In the 5th section we present the behavior of our systems with our commented code written in mCRL2 – for space and formatting consideration, we included the mCRL2 code and extensive comments in Annex A instead of including it in the report. The 6th section of our report is represented by the three verification techniques which we used in order to grow the confidence that our system is correctly modeled. In order to keep the report concise, we included the verification through simulation in a separate annex, B. At the end of our report we include our conclusions and acknowledgments for our project.

Phase 1: Global Requirements

In this phase the global requirements of the system were identified. These describe the mandatory or the forbidden behavior of the wafer-stepper. They have been kept the same throughout the project and haven't been changed, except for rephrasing to make the requirements more clear. They are as follows:

1. System will maintain vacuum in both high and low chambers.
2. Only unprocessed wafers can serve as inputs.
3. When a wafer is processed, route it out of the machine.
4. Both doors of one sluice cannot be opened unintentionally at the same time
5. Wafers cannot be placed in preoccupied positions; wafer cannot be moved out of a door if there is a wafer already waiting outside (not valid for outermost door of the sluices as we are not concerned with what is outside the machine).
6. The system should have a list of allowed malfunctions and raise alarm for it, e.g., even if 1 sluice is not working, reroute, and continue working with other sluice and raise alarm as well. The value provided by the sensors needs be checked for a valid range (plausibility).
7. The system will eventually take an unprocessed wafer into the high vacuum chamber.

Due to time constraint, requirement number six could not be modeled and hence it is not included in further phases of the project.

During this phase, the physical structure of the system was also modeled clearly identifying the position of each chamber, position of doors. IDs were assigned to them for convenience in writing the mcrl2 model.

The physical structure is shown in the Figure no. 1 below

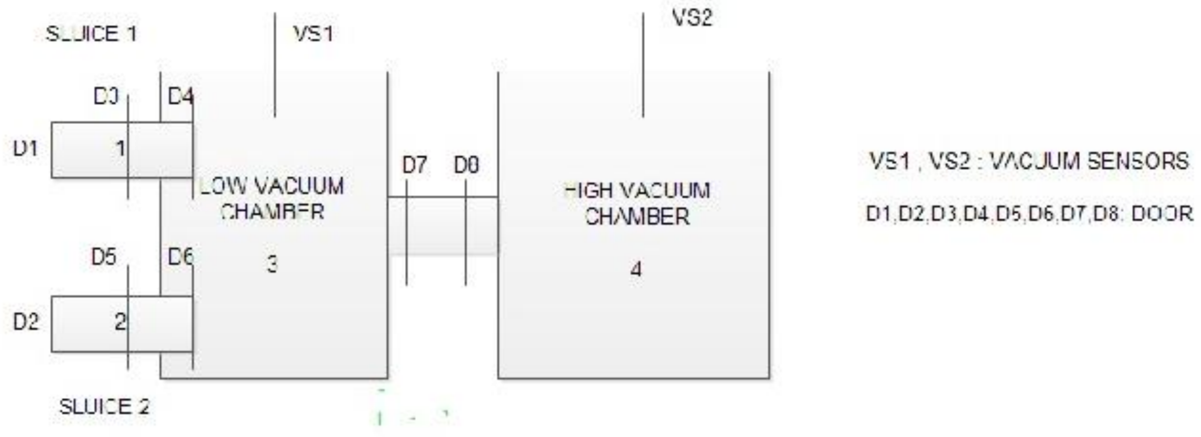


Figure 1 - The physical design of the system

Phase 2: Interactions

The goal of this phase was to identify the interactions that are relevant for the system. The possible interactions were extensively discussed in the team meetings in the initial phase of the project. Later some of them were modified as needed during the later phases of the project. The final version of the interactions is presented in the table below which also has a small description for these interactions. The parameters that these interactions pass are also included in the table. The global interactions are needed for the translation of the global requirements of in terms of those interactions. They are also representing the actions from our mcrl2 code part when modeling the behavior of controllers.

Action	Description	Parameter
openDoorDoorID*	Open the door with number	N/A
closeDoorDoorID*	Close the door with number	N/A
Movewafer0_1	Move a unprocessed wafer inside the sluice1	N/A
movewaferOut2_0	Move the wafer sluice2 outside	N/A
maintainVacuum	Perform the action (pump out the air) to maintain the desired level of vacuum in the chamber	Vacuum level
Startprocessing	Starts processing the unprocessed wafer	N/A
Endprocessing	Ends processing	N/A
acceptWafer1	Sluice one indicates that it can accept an unprocessed wafer	N/A
move	Moving the processed wafer from the high vacuum chamber to the low vacuum chamber	N/A
move1	Moving the processed wafer from the low vacuum chamber to sluice2	N/A
move2	Moving the unprocessed wafer from the sluice1 to the low vacuum chamber	N/A
move3	Moving the unprocessed wafer from the low vacuum chamber to the high vacuum chamber	N/A

Table 1 - Global interactions

Our code also includes a list of **internal actions** which can be seen in Table 2.

movewaferInSourceID_DestinationID	Move the wafer inside the (high/low) chamber or sluice
movewaferOutSourceID_DestinationID	Move the wafer outside the chamber (high/low) or sluice.
send1B	send action to communicate the state of Door (3) to low Vacuum process from Sluice process
send1B_1	send action to communicate 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
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

Table 2 - Internal interaction

Phase 3: Translation of global requirements in terms of interactions

The goal of this phase was to translate the global requirements of Phase 1 in terms of the interactions identified during Phase 2. The process took slightly more time than expected initially and was confusing to all team members leading to several iterations of writing translated requirements, modifying interactions of phase 2 and rewriting the translated requirements in terms of new interactions. The final set of translated requirements is listed below, x.a representing the global requirement and x.b representing the translated property:

1.a System will maintain vacuum in both high and low chambers.

1.b Every **openDoor(i)** action is eventually followed by a **closeDoor(i)** regarding the same door, after which **maintainVacuum** action will be performed in the particular vacuum chamber.

Here, the i parameter stands for the numbers 4,6, 7 and 8. The numbers 4, 6 and 7 indicate the door numbers for the doors of the low vacuum chamber and 8 indicates the door number for high vacuum chamber.

2.a Only unprocessed wafers can serve as inputs to a higher level (Sluice1 -> Low , Low -> High) and only processed wafers can serve as inputs for the exiting sluice (Low → Sluice2)

Only unprocessed wafers can serve as input to the high vacuum chamber and only processed wafers can serve as inputs for the Sluice2.

2.b Two consecutives **move3** actions cannot happen unless a **move** action did not happen between them.

Two consecutives **moveWaferIn0_1** actions cannot happen unless **move2** a move action did not happen between them.

Two consecutives **move1** actions cannot happen unless **moveWaferOut2_0** action did not happen between them.

Two consecutives **move2** actions cannot happen unless **move3** action did not happen between them.

Two consecutives **move** actions cannot happen unless **move1** did not happen between them.

3.a When wafer processed, route it out of the machine

3.b Each **EndProcessing** action is eventually is followed by a **move** action which is eventually followed by a **move1** action and then eventually by a **moveWaferOut2_0** action.

4.a Both doors of one sluice cannot be opened unintentionally at the same time

4.b Each **openDoor1** action cannot be followed by an **openDoor3** action unless a **closeDoor1** has happened in between and vice versa.

Each **openDoor5** action cannot be followed by an **openDoor2** action unless a **closeDoor5** has happened in between and vice versa.

5.a Wafers cannot be placed in pre-occupied positions, wafer cannot be moved out of a door if there is a wafer already waiting outside.(not valid for outermost door of the sluices as we are not concerned with what is outside the machine).

5.b Each **move3** action cannot be followed by an **move3** action unless a **move** has happened in between for High vacuum chamber and similarly for other chambers.

6.a The system should have a list of allowed malfunctions and raise alarm for it, for e.g. even if 1 sluice is not working, reroute, and continue working with other sluice and raise alarm as well. The value provided by the sensors needs be checked for a valid range (plausibility)

7.a The system will eventually take an unprocessed wafer into the high vacuum chamber.

7.b A **moveWaferIn0_1** action will eventually be followed by a **move** action.

Phase 4: Decomposition into n-parallel controllers

The goal of this phase was to decompose the single controller into n-parallel controllers. Various options were considered at the initial stage and after discussion narrowed down to two possible choices

3 controller architecture I

- A controller for all the doors
- A controller for sensing and maintaining vacuum
- A controller for controlling movement of wafer.

3 controller architecture II

- A controller for both the sluices
- A controller for Low Vacuum Chamber
- A controller for High Vacuum Chamber

Both the architectures were considered in detail and based on the factors

- Simpler architecture
- Less communication and more local decisions
- Ease of modeling

Architecture II was chosen considering its obvious advantages over the other architecture. In this architecture each controller is fully responsible for a chamber and only communicates to other controller in case the wafer has to be moved from one chamber to another or from sluice to Low vacuum chamber. This implies that most decisions and actions of maintaining vacuum and opening doors are local and lesser number of decisions requires communication. The decomposed/ parallel architecture is shown in the Figure 2 below.

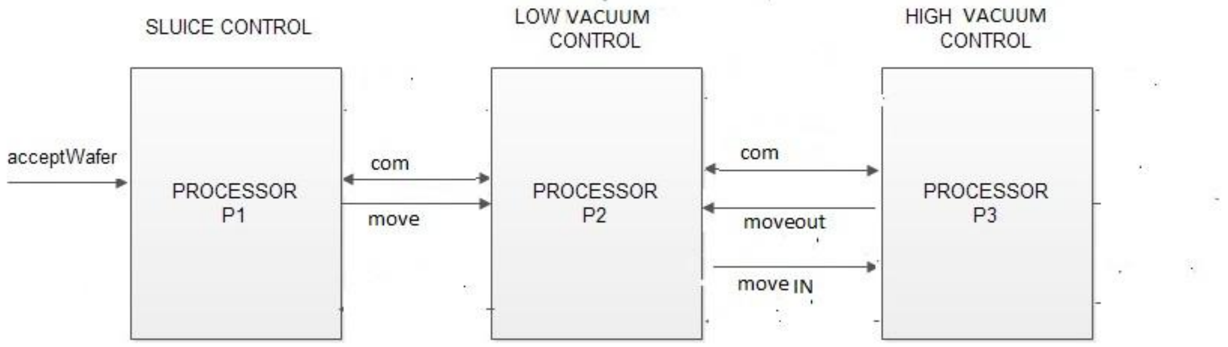


Figure 2 The decomposed architecture of the wafer stepper

To explain Figure 2, we give the description of the Processors below:

- PROCESSOR P1: Controls the doors related to the sluice side (D1,D2,D3,D5)
- PROCESSOR P2:Controls the doors and vacuum sensor related to the Low vacuum chamber (D4,D6,D7,VS1)
- PROCESSOR P2:Controls the doors and vacuum sensor related to the High vacuum chamber (D8,VS2)

Phase 5: Modeling the behavior of the system using mcrl2

The system has been designed to work with 3 parallel controllers

a) 1 for Low Vacuum chamber

b) 1 for both the sluices

c) 1 for High Vacuum chamber

Each controller takes local decisions (maintainVacuum, process wafer(only high vacuum chamber) and communicates with other chambers (state of the chamber(Processed, Unprocessed, Empty, Both) and state of the doors). If conditions are such that movement of wafer is possible from one chamber to another, then the moveWafer actions are synchronized for both the chambers and the wafer is moved after which the doors of the chambers involved are closed.

The code satisfies all the requirements that were set initially and has no deadlocks. This has been verified by using 3 verification techniques for specific requirements.

To review the mCRL2 code please read ***Annex A***, or use the mCRL2 file (**WaferStepper Team7.mcrl2**) that is included in our project.

Phase 6: Verification

To verify the correctness of our model with respect to the requirements listed in the previous sections, we used a combination of three verification techniques: verification through

a) Visualization of the reduced ltsgraph,

b) Simulation

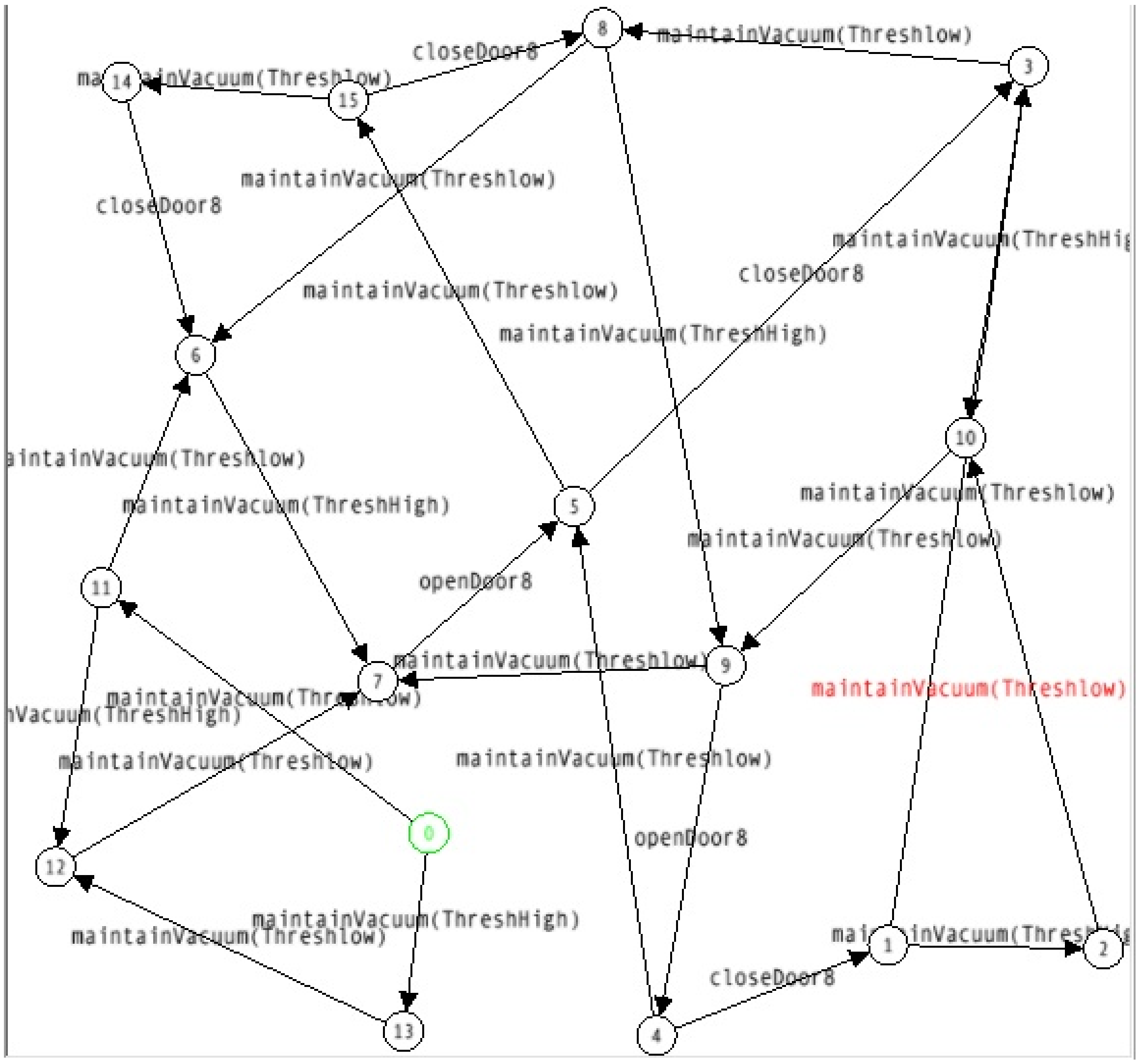
c) Model checking with the mu-calculus translation of our requirements and the corresponding tools from the mCRL2 toolset.

1. Verification through abstraction

The requirements were verified using the abstraction method. Only the necessary actions for a specific requirement were not "hidden" using the hide operator in the mcrl2 specification. The LTS graph images for each of the requirements show the LTS seen after the LTS file was converted using ltsconvert tool with branching bisimulation equivalence.

Requirement 1: System will maintain Vacuum in both high and low chambers.

The below lts graph was obtained by hiding all actions except the maintainVacuum action and the openDoor8 and closeDoor8 actions related to the High Vacuum chamber.



closeDoor8
maintainVacuum(Threshlow)
maintainVacuum(Threshlow)
closeDoor8
maintainVacuum(Threshlow)
maintainVacuum(ThreshHigh)
maintainVacuum(Threshlow)
closeDoor8
maintainVacuum(ThreshHigh)
maintainVacuum(Threshlow)
maintainVacuum(ThreshHigh)
maintainVacuum(Threshlow)
maintainVacuum(Threshlow)
openDoor8
maintainVacuum(Threshlow)
maintainVacuum(Threshlow)
maintainVacuum(Threshlow)
maintainVacuum(ThreshHigh)
maintainVacuum(Threshlow)
maintainVacuum(Threshlow)
openDoor8
maintainVacuum(ThreshHigh)
maintainVacuum(Threshlow)
closeDoor8

Figure 3 - The verification of requirement 1

As we can observe from the graph, the the system will do the action maintainVacuum(ThreshHigh) immediately after closeDoor8. It can also be seen that the system will perform maintainVacuum(ThreshHigh) and maintainVacuum(Threshlow) in the beginning and we cannot miss these actions since we are assuming that the vacuum levels of the system are not in range during the initialization.

It was also observed that low vacuum chamber process performs the maintainVacuum(Theshlow) action immediately after the door actions openDoor(4,6,7),closeDoor(4,6,7),

Requirement 2: Only unprocessed wafers can serve as input to higher level

The requirement was translated as:

Any wafer that has undergone Start Processing, End processing action sequence cannot undergo any action that takes it in back into the High Vacuum chamber.

Since we are not tracking the state of a particular wafer, the action sequence StartProcessing. EndProcessing can always be followed by move3 action which puts a wafer into the High chamber, but obviously for different wafers. Hence it is not possible to verify this requirement using abstraction method. However, this can be verified by taking the trace of the action flow and manually verifying that this particular action sequence cannot occur for a particular wafer.

We have done this by taking the trace when we allow only one wafer to enter the system till it is processed and moved out of the system. This clearly shows that a processing action sequence can never be followed by any action that moves the wafer back into the High Vacuum chamber. The trace can be found in the submitted zip file with the name "REQ_2".

Requirement 3: When wafer is processed route it out of the machine.

When wafer is processed route it out of the machine.

Each EndProcessing action is ultimately followed by movewaferout2_0.

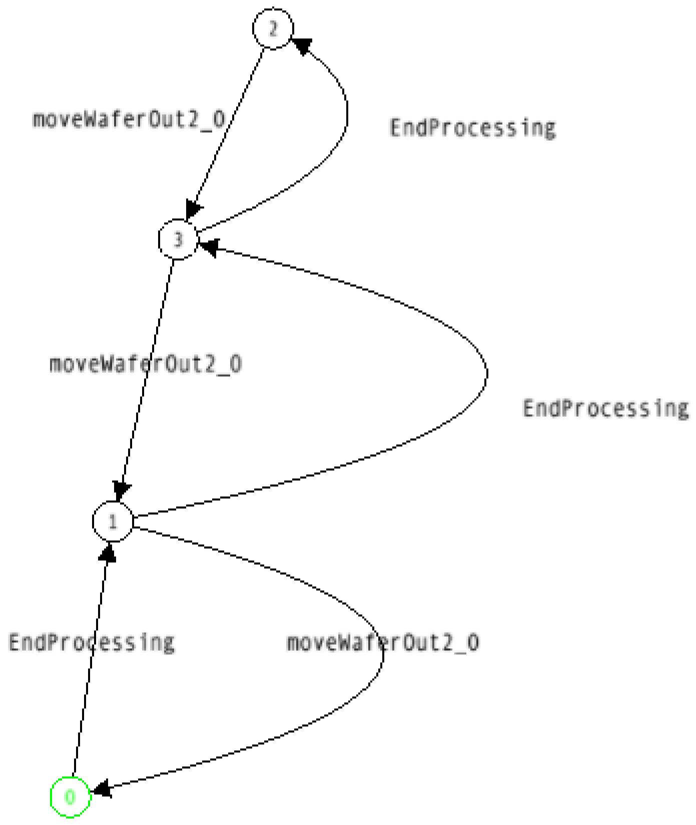


Figure 4 - The verification of requirement 3

The ltsgraph is obtained by hiding all the actions except those nneded for this requirement and then finding the weak trace equivalent using ltsconvert.

Since the system is designed such that it can have 3 processed wafers (1 in exit sluice, 1 in Low vacuum chamber and 1 in High Vacuum Chamber) So it can take any of the path where it may do a EndProcessing action followed by moveWaferOut2_0 or it may do another EndProcessing followed by another EndProcessing (while the previous wafer has been moved out to Low vacuum chamber) and then does a moveWaferOut2_0.

Requirement 4: Both doors of 1 sluice cannot be opened unintentionally at the same time

a) Each openDoor1 action is followed by closeDoor1 action before openDoor3 action. Similarly, every openDoor3 should be followed by closeDoor3 before doing openDoor1 for another wafer.

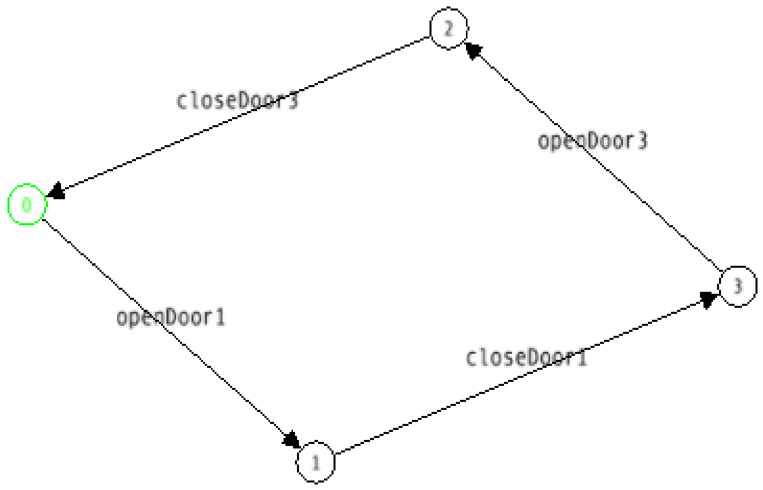


Figure 5 - The verification of requirement 4a

Figure : State space after hiding all actions except those required.

b) Each openDoor2 action is followed by closeDoor2 action before openDoor5 action. Similarly, every openDoor 5 should be followed by closeDoor5 before doing openDoor2 for another wafer.

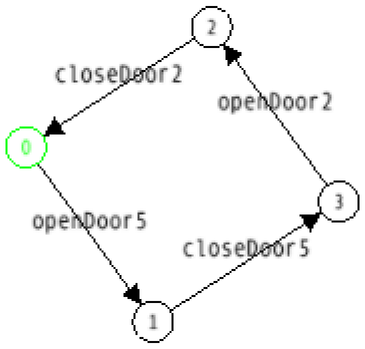


Figure 6 - The verification of requirement 4b

Requirement 5: Wafers cannot be placed in preoccupied positions.

Any move3 action should be followed by move action before another move3 action.

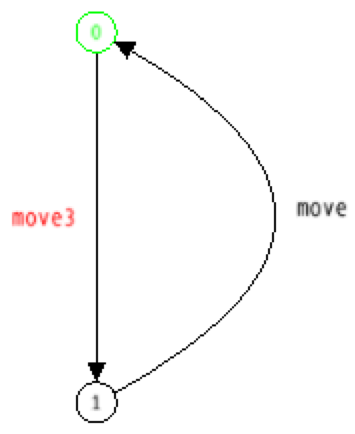


Figure 7 - The verification of requirement 5

Similarly the requirement is true for every action sequence mentioned below

a) move-move1
b) move1-moveWaferOut2_0
c) moveWaferIn0_1-move2
d) move2-move3

This makes sure that a wafer is not moved into a chamber before there is a position vacant for that wafer inside the chamber. Hence the wafers do not take pre-occupied positions.

Requirement 7: The system will eventually take unprocessed wafer into high vacuum chamber.

Each moveWaferIn0_1 is eventually followed by move3 action.

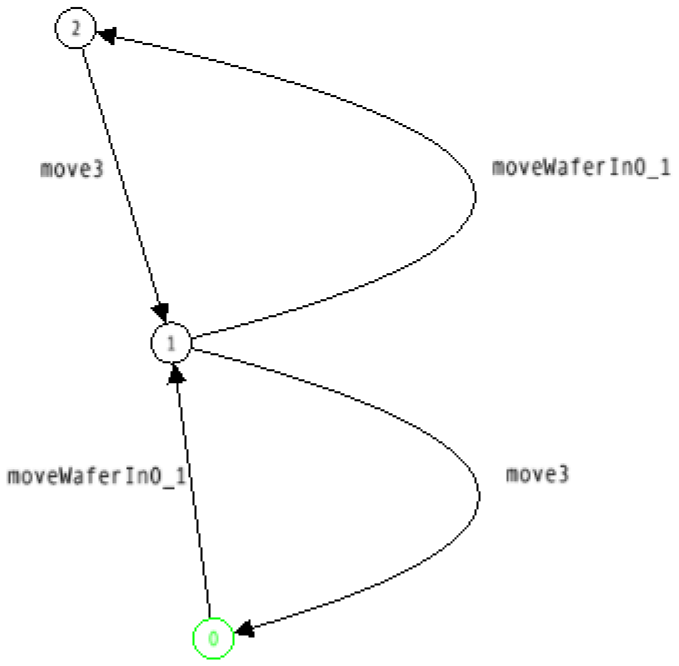


Figure 8 - The verification of requirement 7

The ltsgraph is obtained by hiding all the actions except those needed for this requirement and then finding the weak trace equivalent using ltsconvert.

Since the system has a capacity of 2 unprocessed wafers before taking any wafer into the high vacuum chamber, the system can at maximum do two moveWaferIn0_1 actions after that the system will have to do a move3 action, moving the unprocessed wafer into high vacuum chamber.

Simulation using xsim

We also verified some of our requirements using the xsim simulation tool available in the mCRL2 toolset.

As for the previous verification technique, we included an extensive description and all the details of this evaluation in a separate document, **Annex B**.

Model checking

For the model checking using mu-calculus formulas verification technique we took a step forward from the translation of the requirements in Phase 3. We translated each requirement into mu calculus formulas as seen below . We then used the LPS2PBES and PBES2BOOL tools in the mcrl2 toolset to verify the mu-calculus property.

By evaluating the formulas which will follow below, we came up with the results presented in Table 3. In the table, the generation time represents the time taken to compute the BES from the PBES; the solving time represents the time taken to solve the BES; number of Boolean variables, number of equations and the rank of the equations are self-descriptive. The mcf file's running output can be found in **annex C- PBES2BOOL Output.pdf**.

Take note that the requirements were evaluated with a Intel Core 2 Duo CPU @2GHz, 1GB RAM running Ubuntu.

Requirement	Generation time [milliseconds]	Solving time [milliseconds]	Number of Boolean variables	Number of equations	Maximum rank of the equations	Results
#1	2801	40	10073	10498	8	True
#2	5398	31	19122	19294	1	True
#3	2874	38	10090	10622	6	True
#4	1704	9	6037	6094	1	True
#5	1170	4	4039	4125	1	True
#7	546	7	2056	2849	2	True

Table 3 - Generated metrics by the mu-calculus verification

For the different formulation of the requirements we use the following notations: x.a represents the global property, x.b the translated property as in Phase 3 and x.c represents the mu calculus formula for this requirement.

1.a System will maintain vacuum in both high and low chambers.

1.b Every **openDoor(i)** action is eventually followed by a **closeDoor(i)** regarding the same door, after which **maintainVacuum** action will be performed in the particular vacuum chamber.

As mentioned before, the i parameter stands for the numbers 4, 6, 7 and 8. The numbers 4, 6 and 7 indicate the door numbers for the doors of the low vacuum chamber and 8 indicates the door number for high vacuum chamber.

1.c In order to validate all the properties that come out from this formulation, we combined them into the following mu-calculus formula:

```
 ([true*.(openDoor8)]  mu X.[closeDoor8.maintainVacuum(ThreshHigh)]X)
&&
([true*.(openDoor4)]  mu X.[closeDoor4.maintainVacuum(Threshlow)]X)
&&
([true*.(openDoor6)]  mu X.[closeDoor6.maintainVacuum(Threshlow)]X)
&&
([true*.(openDoor7)]  mu X.[closeDoor7.maintainVacuum(Threshlow)]X)
```

2.a Only unprocessed wafers can serve as inputs to a higher level (Sluice1 -> Low , Low -> High) and only processed wafers can serve as inputs for the exiting sluice (Low → Sluice2)

Two consecutives **move3** actions cannot happen unless a **move** action did not happen between them.

Two consecutives **moveWaferIn0_1** actions cannot happen unless **move2** a move action did not happen between them.

Two consecutives **move1** actions cannot happen unless **moveWaferOut2_0** action did not happen between them.

Two consecutives **move2** actions cannot happen unless **move3** action did not happen between them.

Two consecutives **move** actions cannot happen unless **move1** did not happen between them.

```
2.c
([true*.move3.!move*.move3]false)
&&
([true*.moveWaferIn0_1.!move2*.moveWaferIn0_1]false)
&&
([true*.move1.!moveWaferOut2_0*.move1]false)
&&
([true*.move2  . (!move3)*  . move2]false)
&&
([true*.move  . (!move1)*  . move]false)
```

Only unprocessed wafers can serve as input to the high vacuum chamber and only processed wafers can serve as inputs for the Sluice2.

3.a When wafer processed, route it out of the machine

3.b Each **EndProcessing** action is eventually is followed by a **move** action which is eventually followed by a **move1** action and then eventually by a **moveWaferOut2_0** action.

3.c – Again, we split this long formula into 3 shorter properties that need to hold at the same time.

```
([true*.EndProcessing]mu X.[!move]X)
&&
([true*.move]mu  X.[!move1]X)
```

```
&&
([true*.move1]mu  X.[!moveWaferOut2_0]X)
```

4.a Both doors of one sluice cannot be opened unintentionally at the same time

4.b Each **openDoor1** action cannot be followed by an **openDoor3** action unless a **closeDoor1** has happened in between.

Each **openDoor5** action cannot be followed by an **openDoor2** action unless a **closeDoor5** has happened in between.

4.c We combined the two formulas :

```
([true*.openDoor5 . !closeDoor5* . openDoor2]false)
&&
([true*.openDoor1 . !closeDoor1* . openDoor3]false)
```

5.a Wafers cannot be placed in pre-occupied positions, wafer cannot be moved out of a door if there is a wafer already waiting outside.(not valid for outermost door of the sluices as we are not concerned with what is outside the machine).

5.b Each **move3** action cannot be followed by an **move3** action unless a **move** has happened in between.

This requirement is already covered by the second requirement.

```
[true*.move3.  !move*  . move3]false
```

6.a The system should have a list of allowed malfunctions and raise alarm for it, for e.g. even if 1 sluice is not working, reroute, and continue working with other sluice and raise alarm as well. The value provided by the sensors needs be checked for a valid range (plausibility)

7.a The system will eventually take an unprocessed wafer into the high vacuum chamber.

7.b A **moveWaferIn0_1** action will eventually be followed by a **move3** action.

7.c

```
[true*.(moveWaferIn0_1)]  mu X. [move3]X
```

Conclusions

For this project we modeled the new generation of the ASML wafer stepper, as indicated in the original assignment. Our work was split in six phases. In the first phase we set up a list of global requirements that our system needs to respect. Next, we defined the internal and external actions which were needed for the correct functionalities of the system. During our projects, these actions have suffered modifications, up to the final version which we reported in this document. In the third phase of our project we translated the global requirements in terms of the previously defined actions. For the 4th phase we designed the architecture which we were going to use to model the system.

We continued with the modeling of the system using the mCRL2. The code had an early stage, at which each of the three components were built and tested. We experienced deadlocks and we spent a lot of time to fix them. With the help of the abstraction visualization and the simulation method we were able to eliminate the deadlocks completely.

In the final phase of the project we validated the system's requirements using three different methods: abstraction visualization, simulation and verification through model checking with the help of mu-calculus formulas.

Our system's generated an lts file which contains 116 levels, 2560 states and 6128 transition.

The model was found to be deterministic.