Analysis of the Intel Five-Machine Six Step Mini-Fab

Ing. J.P.A. van den Berk

SE 420383

Masters thesis

Supervisor: Prof.dr.ir. J.E. Rooda
Coach: Dr.ir. A.A.J. Lefeber

Eindhoven University of Technology
Department of Mechanical Engineering
Systems Engineering Group

Eindhoven, May 2004

Contents

Summary

Controlling a re-entrant flow line, as commonly used in the semi-conductor industry, is complex. The complexity is illustrated well by the Intel Five-Machine Six Step Mini-Fab case. The case contains all difficulties that need to be dealt with in practice. Equipment and personnel can be unavailable, products need to be transported and stocked and machines have various properties to account for. The target of the case is to reach a throughput of 84 products per week despite the difficulties.

The re-entrant flow line of the Intel case has been modelled in χ, because χ is well suited for modelling discrete-event systems like the line. Techniques for modelling elegantly in χ were used to acquire an elegant model. For the construction of the model, first a basic flow line has been considered, that contains only a product generator, buffers, machines and an exit process. Subsequently, the model is extended with additional complexity until every aspect of the case has been modelled. Firstly, machine input restrictions and a product transporter are added to the model. Next, two operators are added and finally a technician is added to the model. The flow line is controlled by push control and the flow line elements operate FIFO (First In First Out). After the model has been designed, it is validated. It behaves as it was designed to behave and represents the case correctly. After the validation of the model, simulations can be run. With the current control the simulation results show that the required throughput can not be met. The maximal throughput, neglecting the buffer capacity constraints, equals 57.5 products per week. To increase the throughput, the control of the flow line needs to be improved.

Before the control of the flow line is improved, the control problem of the Intel case is analyzed to determine if the required throughput can be met. The analysis is firstly performed by hand. Capacity analysis of the flow line elements shows that the elements have enough capacity. However, the personnel availability issues are not combined with the machine availability issues in the capacity analysis. To combine these issues, an educated schedule of one shift without machine break down is constructed manually. The schedule shows that the control problem without break down is feasible. Next, machine break down is added to the schedule. To account for the stochastics of the break down, the estimated required extra capacity of the machines is also added. To include the break down behavior correctly, multiple shifts have to be scheduled. Therefore, an integer linear programming model of the case is designed. Because the

model is too complex to be solved with the available optimization tools, even after simplification, the model is relaxed by neglecting the integer constraints. Under this assumption production for one week can be scheduled in 0.97 weeks. Since the model is relaxed, the resulting schedule is not feasible. However, all machines except the bottleneck machine have some surplus capacity and the schedule may be prolonged by 0.03 weeks. Therefore, it is assumed that a feasible schedule of the production can be constructed and that the control problem of the Intel case is feasible.

Finally, the control of the flow line is improved. When machine break down is neglected, the control problem becomes deterministic and a repetitive schedule can be implemented. The required throughput can then be reached, as described in the previous paragraph. When machine break down is not neglected, a repetitive schedule can not be used to control the flow line, due to stochastics. Instead, heuristic control is implemented in small steps to acquire elegant control. The control remains as simple as possible, so it can be easily understood and adjusted. Firstly, the control of the product generator and the buffers is improved to increase the throughput. Secondly, the operating and off time priorities of personnel are improved. To make the control easier to analyze, machine break down is neglected. The maximal throughput equals 80.4 products per week, while the buffer capacity constraints are neglected. Finally, the control should be refined and control for after a machine break down occurs should be implemented. Due to lack of time in the project, this has not been done yet.

Samenvatting

Het aansturen van een teruggekoppelde productielijn, zoals vaak voor komt in de semiconductor industrie, is complex. De complexiteit wordt duidelijk weergegeven door de Intel Five-Machine Six Step Mini-Fab casus. De casus bevat alle problemen die men in de praktijk tegenkomt. Productie gereedschap en personeel zijn niet altijd aanwezig, producten dienen te worden getransporteerd en opgeslagen en machines hebben diverse eigenschappen waar rekening mee gehouden moet worden. Het doel van de casus is een doorzet van 84 producten per week te bereiken, ondanks de problemen.

De teruggekoppelde productielijn van de Intel casus is gemodelleerd in χ, omdat χ uitermate geschikt is voor het modelleren van systemen die op discrete tijdstippen van toestand veranderen, zoals de lijn. Technieken voor elegant modelleren in χ zijn toegepast om een elegant model te verkrijgen. Voor de opbouw van het model is eerst een eenvoudige productielijn beschouwd, die slechts een product-generator, buffers, machines en aan het einde een opslag proces bevat. Het model is achtereenvolgens uitgebreid met additionele complexiteit, totdat elk aspect van de casus gemodelleerd is. In eerste instantie zijn restricties op de toevoer van producten aan machines alsmede een product-transportsysteem toegevoegd aan het model. Vervolgens zijn twee operators toegevoegd en als laatste is een monteur toegevoegd aan het model. De productielijn wordt aangestuurd door producten in de lijn te duwen. De elementen van de lijn werken op volgorde van aankomst. Nadat het model ontworpen is, wordt het gevalideerd. Het model gedraagt zich zoals het zich dient te gedragen en vertegenwoordigt de casus correct. Na de validatie van het model kunnen simulaties worden gedraaid. De simulatie resultaten laten zien dat met de huidige aansturing van de productie lijn de gevraagde doorzet niet kan worden behaald. Indien de restricties op de capaciteit van buffers verwaarloosd worden, bedraagt de maximale doorzet 57.5 producten per week. Om de doorzet te verhogen, moet de aansturing van de productie lijn verbeterd worden.

Voordat de aansturing van de productielijn verbeterd wordt, wordt het aanstuur probleem van de Intel casus geanalyseerd om te bepalen of de gevraagde doorzet gehaald kan worden. De analyse wordt in eerste instantie met de hand uitgevoerd. Capaciteitsanalyse van de elementen van de productielijn laat zien, dat alle elementen voldoende capaciteit hebben. De beschikbaarheid kwesties van het personeel zijn echter niet gecombineerd met de beschikbaarheid kwesties van de machines in de capaciteitsanalyse. Om deze kwesties wel te combineren, is handmatig een goed doordachte planning gemaakt

van een halve werkdag, waarin het faalgedrag van machines wordt verwaarloosd. De planning laat zien dat het aanstuurprobleem zonder faalgedrag van machines oplosbaar is. Vervolgens wordt het faalgedrag toegevoegd aan de planning. Om de stochastiek van het faalgedrag in de planning mee te nemen, wordt ook de geschatte benodigde extra capaciteit van de machines toegevoegd. Meerdere werkdagen zijn gepland, om het faalgedrag correct toe te voegen. Daarom is een integer lineair programmeringsmodel van de casus ontworpen. Omdat het model zelfs na simplificatie te complex is om opgelost te worden met de beschikbare optimalisatie gereedschappen, is het model gerelaxeerd door de integer restricties te verwaarlozen. Onder deze aanname kan de productie van een week gepland worden in 0,97 weken. Aangezien het model gerelaxeerd is, is de resulterende planning niet uitvoerbaar. Alle machines behalve de bottleneck machine hebben echter enige capaciteit over en de planning mag verlengd worden met 0,03 weken. Daarom wordt aangenomen dat een uitvoerbare planning van de productie gemaakt kan worden en dat het aanstuur probleem van de Intel casus oplosbaar is.

Tot slot wordt de aansturing van de productielijn verbeterd. Wanneer het faalgedrag van machines verwaarloosd wordt, wordt het aansturingsprobleem deterministisch en kan een vaste planning telkens herhaald worden. De gevraagde doorzet kan dan gehaald worden, zoals in de vorige paragraaf beschreven is. Wanneer het faalgedrag van machines niet verwaarloosd wordt, kan de aansturing van de productielijn niet plaatsvinden door telkens een vaste planning te herhalen, omdat de stochastiek van de lijn dit niet toelaat. In plaats daarvan wordt heuristische aansturing geïmplementeerd in kleine stappen, zodat een elegante aansturing verkregen wordt. De aansturing wordt zo eenvoudig mogelijk gehouden, zodat hij gemakkelijk te begrijpen en aan te passen is. Op de eerste plaats is de aansturing van de product-generator en de buffers verbeterd om de doorzet te verhogen. Op de tweede plaats zijn de prioriteiten van het personeel voor bediening en afwezigheid verbeterd. Om de aansturing gemakkelijker te kunnen analyseren, wordt het faalgedrag van machines verwaarloosd. De maximale doorzet bedraagt 80,4 producten per week, terwijl de restricties op de capaciteit van buffers verwaarloosd worden. Tot slot dient de aansturing verfijnd te worden en dient de aansturing voor de situatie dat een machine faalt te worden geïmplementeerd. Door gebrek aan tijd voor het project, heeft deze implementatie nog niet plaats gevonden.

Preface

After almost three years of broadening my insights on Mechanical Engineering in general and more specific on Systems Engineering at the Eindhoven University of Technology, I have finish my education with this masters thesis. I am confident that I am ready to wander of into the real world.

Firstly, I would like to take this opportunity to thank my parents, who have made it possible for me to continue my education at the university, after having graduated at Fontys Hogescholen Eindhoven. Next, my gratitude goes out to my coach during this masters project, Dr.ir. A.A.J. Lefeber from the Control and Optimization research group. Finally, I thank Prof.dr.ir. J.E. Rooda for his supervision of the project and advice on discrete-event modelling.

Johan van den Berk,
Eindhoven, May 2004.

Chapter 1

Introduction

Controlling a re-entrant flow line, as commonly used in the semi-conductor industry, is complex. To illustrate the complexity of the control problem, Dr. Kempf from Intel composed the Intel Five-Machine Six Step Mini-Fab [2] case. As the title suggests, this case concerns a production facility in which products are processed on five machines in six steps. The products are wafers, which are silicon plates produced in the semi-conductor industry. They contain hundreds of identical chips used in electronic accessories like computers and mobile phones. The case clearly illustrates the difficulties that need to be dealt with in practice. Equipment and personnel can be unavailable, products need to be transported and stocked and machines have various properties to account for. The target of the case is to reach a required amount of throughput with the flow line despite the difficulties.

In this masters project the re-entrant flow line of the Intel case is simulated, analyzed and controlled. First, an elegant discrete-event model with FIFO (First In First Out) control of the flow line is designed to simulate the factory. The χ formalism is used to design the model, because χ is well suited for modelling discrete-event systems like many production environments. Before improving the control of the flow line, the control problem of the Intel case is analyzed to determine if the required throughput can be met. Initially the stochastics of the flow line are neglected in this analysis. The case is analyzed manually by capacity analysis and by constructing an educated schedule. Next, computer aided analysis is performed by designing and optimizing a linear programming model of the case. And finally, the influence of stochastics on the control problem is analyzed. After the analysis of the problem, the control of the flow line is improved in the discrete-event model to increase the throughput of the line. The implemented heuristic control is derived from analyzing the behavior of the flow line. The control is now ready to be implemented in a real flow line.

The simulation, analysis and control of the Intel case are described in this report, after the case is presented in Chapter 2. The discrete-event model with FIFO control is presented in Chapter 3. First, the designing guidelines are presented. Next, the

designing of the model is described. Finally, the model is verified and the simulation results of the flow line are discussed. Chapter 4 describes the analysis of the control problem. The manual analysis is reported in the first section, followed by the computer aided analysis in the second section and the stochastic analysis in the final section. The control of the re-entrant flow line is improved in Chapter 5. Finally, the conclusions and recommendations are presented in Chapters 6 and 7.

Chapter 2

Case description

Before presenting the research of the project, the Intel Five-Machine Six Step Mini-Fab case is described in this chapter. The case is a simple model of a factory. The description starts with the basic flow line and adds complexity to the line in steps. In Section 2.1 the basic flow line is presented. It contains the factory areas, the products, the production sequence and the basic production equipment. Section 2.2 adds restrictions on machine input and product transportation to the basic flow line. Section 2.3 adds operators to the flow line. They load and unload the machines and perform setup on them. Finally, the flow line is completed by adding a technician to it in Section 2.4. The technician performs scheduled and unscheduled maintenance on the machines.

2.1 Basic flow line

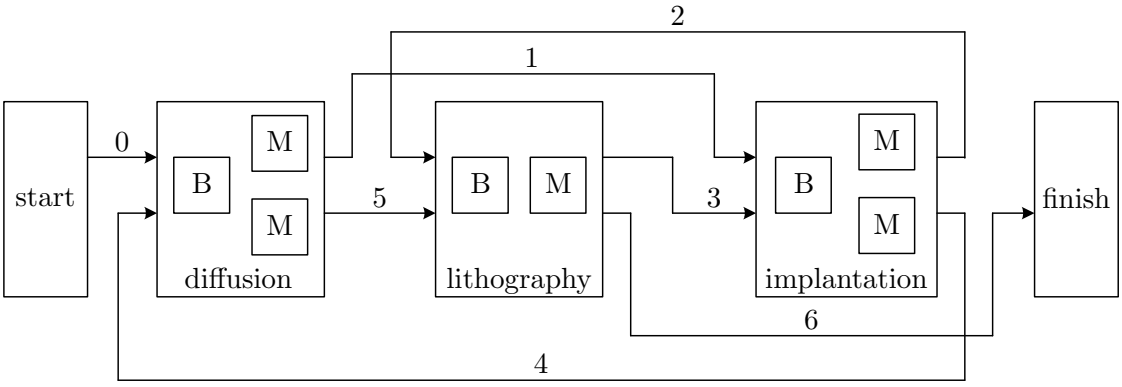


Figure 2.1: *Basic flow line*

In this section the basic flow line of the Intel case is described. The line is shown in Figure 2.1. It operates non-stop and contains five areas. The first area is the start

area. Before the products of the flow line are processed, they are stocked in this area. The next three areas are the workstation areas diffusion, lithography and implantation. They contain the diffusion, the lithography and the ion implantation process of the wafer fabrication. The final area is the finish area. The finished products are stocked in this area.

Three types of products flow through the areas. The first type is the test product. It is used to monitor the accuracy of the production process. The other two types (type A and type B) are commercial products. They are to be sold to customers. Per week 3 test products, 51 type A products and 30 type B products are taken from the stock in the start area and are released into the flow line. The released products follow the predefined production sequence visualized in Figure 2.1. Six production steps are performed on them in three workstations. Each workstation is entered twice. The products are processed in the sequence diffusion, implantation, lithography, implantation, diffusion and lithography. Finally, the finished products are stocked in the finish area.

The production equipment inside the workstations is shown in Figure 2.1. Each workstation contains one buffer (B) for stocking the incoming and outgoing products of the workstation. The capacity of the buffer in the diffusion workstation equals 18 products. In the lithography and the implantation workstation the capacity of the buffer equals 12 products. Apart from a buffer, the diffusion and the implantation workstation contain two machines (M) and the lithography workstation contains one machine. A machine receives a product from the workstation buffer and processes it. Next, it returns the product to the buffer. All machines perform a low and a high production step on a product respectively the first and the second time the product enters the workstation. The production times of the low and high step of the diffusion process are respectively 225 min and 255 min. The production times of the lithography process are 55 min and 10 min and the production times of the ion implantation process are 30 min and 50 min. Equipment preemption is excluded from the case, thus once a machine starts a job, it finishes it before starting another one. Apart from preemption also rework is excluded from the case. This concludes the basic elements of the flow line.

2.2 Flow line with machine input restrictions and product transportation

In this section restrictions on machine input and product transportation are added to the basic flow line of the previous section. Machine input has two restrictions. Firstly, test products are processed once by both machines in multi-machine workstations. So when a test product enters the diffusion or the implantation workstation for the second time it can only be processed on the machine it was not processed on the first time. Thus, the entire flow line is monitored. Secondly, diffusion machines require input of batches. A batch contains three products of equal production step. It also contains

less than two test products. Furthermore a high step batch does not contain a mix of commercial products.

Next, product transportation between areas is added to the flow line. The transportation is performed by the product transporter. The transporters capacity equals one product. At the start of a transportation cycle, it moves empty to the pickup area. Next, the product is loaded from a stocker or a buffer, depending on the area, into the transporter and is transported to the desired area. When it arrives, the product is unloaded from the transporter into a stocker or a buffer, again depending on the area, and the cycle is finished. The transporter moves in 4 min empty or full to an adjacent area and loads or unloads a product in 1 min. This concludes the product transportation.

2.3 Flow line with operators

In this section two operators (operator 0 and operator 1) are added to the flow line with machine input restrictions and product transportation. The operators have three tasks. Firstly, they load the machines. The machine input is retrieved from the workstations buffer and is loaded into the machine. The loading times of the diffusion, the lithography and the ion implantation process are respectively 20 min, 10 min and 15 min. Secondly, the operators unload the machines, after the machines have processed their input. The machine output is stocked in the workstations buffer. The unloading times of the diffusion, the lithography and the ion implantation process are respectively 40 min, 10 min and 15 min. Finally, an operator performs setup on the lithography machine when the machines input changes. Setup times for a change in product type, product step or both are respectively 5 min, 10 min or 12 min. The setup is performed immediately prior to the loading of the machine. During setup the machine is empty.

Operators not only have tasks, but also have restrictions. Firstly, operator 0 serves the diffusion and the implantation workstation and operator 1 serves the lithography and the implantation workstation. Secondly, each operator has two breaks of 60 min and one meeting of 60 min per shift of 720 min. Their off time is unrestricted to synchronization. Thirdly, operators operate non-preemptive, so once they start a task, they finish it before starting another one. Finally, operators require 1 min to move to an adjacent workstation. The number of operators that are simultaneously in transport is unrestricted. All operator restrictions have now been discussed and the operators have been added to the flow line.

2.4 Flow line with operators and a technician

Next to the operators that have been added to the flow line in the previous section, the factory employs a technician to perform maintenance on the machines. The technician performs two types of maintenance. The first type is 'scheduled maintenance' and is

required by every machine. A machine is empty when it starts and remains empty during it. The scheduled maintenance for the machines in the diffusion, the lithography and the implantation workstation takes respectively 75 min, 30 min and 120 min. A window exists in which the maintenance should be started. If the maintenance is not started in the window, then the current machine cycle is finished, but the machine will reject new input until the technician performs the maintenance. The window opens for the diffusion machines at the beginning of each day or 720 min after the previous scheduled maintenance, whichever is later. It closes at the end of the day. For the lithography machine and the ion implantation machines the window opens at the beginning of each shift or 360 min after the previous scheduled maintenance, whichever is later. It closes at the end of the shift.

The second type of maintenance that the technician performs is 'unscheduled maintenance' and is required by the ion implantation machines, because the machines break down during processing. The current product inside a down machine is wasted and the machine rejects new input until the machine has been repaired. The repair time and the time between failure are stochastically distributed and are respectively 420 min $\pm$ 60 min and 3000 min $\pm$ 1560 min. Scheduled and unscheduled maintenance are required independently of each other.

Similar to the operators, the technician has restrictions. Firstly, he has two breaks of 45 min and one meeting of 30 min per shift. Secondly, he operates non-preemptive. Finally, the technician requires 1 min to move to an adjacent workstation. This completes the description of the factory. In the next chapters the research of the project will be presented, starting with the χ model of the case.

Chapter 3

χ model

In the previous chapter the Intel case has been described. In this chapter the χ model of the case is presented. Information on the χ formalism can be found in [7] and [1]. An introduction to modelling a flow line in χ can be found in [4]. The model is build starting with a simple version that contains the basic production processes and subsequently extending it with additional complexity until every aspect of the case has been included in the model. FIFO (First In First Out) control is present at this stage of the project. Before describing the χ modelling, Section 3.1 presents the used modelling techniques for designing elegant models. Sections 3.2 through 3.5 describe the modelling of the case according to the structure of the case description of the previous chapter. The analysis of the model is presented in Section 3.6.

3.1 Modelling elegantly

Before the modelling of the case is described, the used modelling techniques are presented. They are not rules to assure an elegant χ model, but are guidelines for designing elegant models. Two general techniques for modelling are used. Firstly, record the choices made in earlier stages of the design to aid making correct decisions in the later stages. Secondly, design a basic model and increase the complexity of it in small steps until the model is complete. Not only does this approach result in an elegant model, but it also simplifies the verification of the model. Next to general modelling techniques, also techniques that focus on the model itself are used. Firstly, move the complexity of the model to functions. They are more transparent than processes, because they do not permit synchronization statements, communication statements or channels. Secondly, avoid non-determinism in the model to make it predictable. Next, avoid deadlocking and live-locking. A process that is interacting with another process causes deadlock if the other process remains blocked in a synchronization or communication action that will no longer succeed. A process causes live-lock if it is never blocked by a synchronization or communication action. Furthermore, increase the readability of the model

by designing clear, compact and insightful constructions and by using consistent indentation and nomenclature in the model. Next, utilize library functions to increase the compactness of the model. Finally, gather the modelling parameters in a separated section of the model. In this way the parameters and thus the simulation scenario can be changed easily. All used modelling techniques for modelling elegantly in χ have now been described. Before presenting the χ model, the modelling of the case is described in the next four sections.

3.2 Basic flow line

In the previous section the modelling techniques have been discussed. In this section they will be used to design the basic flow line. Not only the modelling techniques, but also state diagrams of the case elements are used during modelling to get a correct and elegant model of the case. The diagrams are shown in [2].

The design of the basic flow line starts with determining the time unit for simulation time. The unit is set equal to one minute, so all time-related model parameters are integers. The next step in the design is modelling the factory operating time. Because the factory operates non-stop, no forced production idle time is modelled. The production start time is 0 min and it continues endlessly.

The products that are processed during production are represented in the model by a record tuple containing the strictly necessary information. The first element of the tuple is the product identification number. The number is required to identify the products in the flow line. The second element is the production step of the product, which is required to define the production state of the product. The last element of the tuple is the release time of the product and equals the time that a product is released into the flow line. The release time is used to calculate the products individual flow time that is required to determine the mean flow time of the line.

The products are processed in the factory areas by the basic production equipment, as shown in Figure 3.1. The factory areas are not modelled as processes, but are represented by the production equipment inside the areas. The start area is represented by a product generator that releases products into the flow line. Two options exist to model the generator. The first option is to model one process that creates products. It uses a function to determine the product type. The process releases the products into the flow line with constant time intervals. The second option is to model two types of generator processes, the signal generator (Gs) and the collector process (Gc). The signal generator represents a product type. It determines the release times of the products of its type by signalling the collector with constant time intervals. The length of the intervals depends on the number of products per type that are released per week. Three signal generators are required to model all product types. The collector receives signals from the signal generators and therefore knows when to release a product of which type

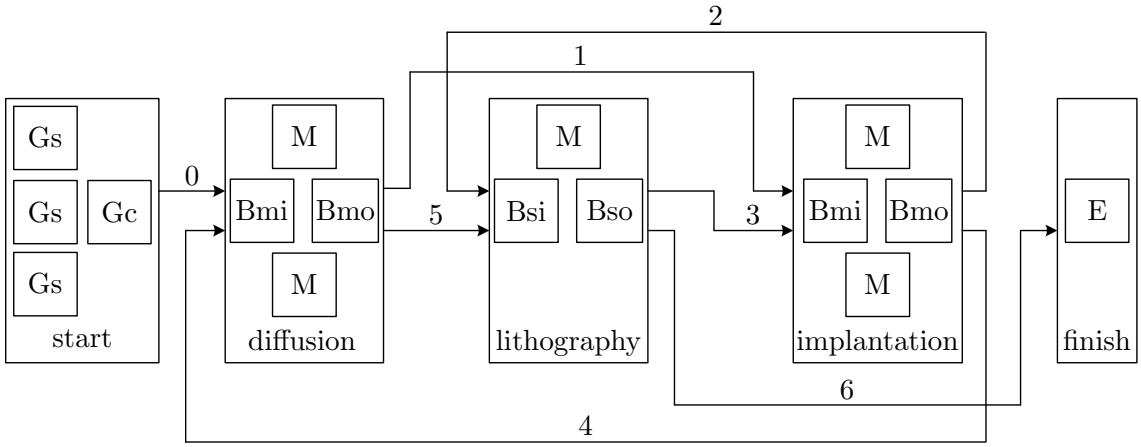


Figure 3.1: *Basic production equipment in* χ

into the flow line. The function that determines the product type in the first option brings unnecessary complexity into the model and therefore the second option is chosen.

After the generator has been modelled, the workstation areas are designed. The diffusion and the implantation area are represented by two buffers (Bmi and Bmo) and two machines (M). To exclude blocking from the model, the buffers are modelled to have infinite capacity. The finite buffer capacity should be modelled by improved control, which is not included in this model. The products that enter the workstations, arrive at the incoming buffer (Bmi) and are stocked. When a machine is idle, it requests the incoming buffer for a product. Without this construction the buffer would not know which machine requires a product. As a consequence the buffer would have to try to send the product to both machines. If both machines want to receive a product at the time of sending, then it is not known to which machine the product is sent. This would therefore include much non-determinism in the model. After a machine has placed a request for input, the buffer sends a product from its contents to the machine. Because at this stage of the project FIFO control is modelled, the buffer operates FIFO (First In First Out). When a machine receives a product, it has to determine the processing time. All machines are instances of the same process, but have different processing times. These times are put into an array tuple. The machine determines the correct processing time from the tuple using the production step of the product it has received. Another option to determine the processing time would have been to include processing time parameters in the machine process. Because of the extra parameters, this options increases the complexity of the model unnecessarily. Now the processing time has been determined the machine waits for the duration of it. Next, the production step of the product is increased by one and the machine sends the product to the outgoing buffer (Bmo). Modelling an incoming and an outgoing buffer per workstation instead of one multi functional buffer clarifies the flow of products through the line. The outgoing buffer receives the product and sends it directly to the next area. It has two product

sending channels, as is visualized in Figure 3.1, and uses the production step of the product to determine the correct channel to send over.

The lithography workstation area is represented by two buffers and one machine. The structure of the area is similar to the other workstation areas. The only difference is formed by the incoming (Bsi) and the outgoing (Bso) buffer of this area. Because they serve only one machine, they require only half of the channels that are used by the diffusion and implantation buffers for connection to the machines in the workstation. Therefore different buffer processes are modelled. Another option would have been to use the multi-machine workstation buffers in this area. But then half of the buffer channels that should be connected to a machine, is connected only to the buffer process. This option is therefore less elegant.

The finish area is the final area of the flow line. The area is represented by the exit process (E). This process continuously accepts products and can be used to determine the throughput and the mean flow time of the flow line. The products leave the flow line at the exit process. This concludes the areas and their production equipment.

After describing the areas, the channels through which the products are passed between the areas are presented. Figure 3.1 shows the channels. The collector process of the generator sends the products to the incoming buffer of the diffusion workstation. The outgoing buffer of each workstation sends the product to the incoming buffer of the next production area. Therefore all incoming and outgoing buffers have two connection channels. The correct channel to send the products over is determined by observing the production step of the products. When all the processing of a product has been finished, the outgoing buffer of the lithography workstation sends the product to the exit process. The modelling of the basic flow line has now been presented.

3.3 Flow line with machine input restrictions and product transportation

According to the structure of the case description in Chapter 2, restrictions on machine input and product transportation are added to the basic flow line in this section. Two machine input restrictions are modelled. Firstly, test products are processed once by both machines in the multi-machine workstations. To identify test products, a type number is inserted as second element in the product representation tuple. The restriction can now be modelled in three ways. Firstly, a controller can be designed that governs the machine routing of test products. Secondly, all incoming buffers in multi-machine workstations can record the machine history of the test products and send them to the correct machine. Finally, the machine history of the test products can be added to the product representation tuple, so the incoming buffers can determine the correct machine to send the products to. The first option adds much complexity to the model, because an extra process and extra communication channels are necessary. The second option makes the buffers unnecessarily complex, while they should remain

basic processes. The final option is chosen for its simplicity. Not only test products, but all products record their machine history to keep the product representation tuples uniform. The history is inserted as fourth element in the product representation tuple. The history is updated simultaneously with the processing step of the product after the product has been processed on a machine. The history update is required only after a product has been processed in a multi-machine workstation for the first time. If the product is processed in the lithography workstation or in any workstation for the second time, then the history update is not required. Since the update is not harmful and the prevention of it adds unnecessary actions to the model, the history update is performed after every processing step. The machine history of a product is recorded in an array tuple with three elements. One element is used for each workstation. The elements contain the identification number of the machine on which the product was last processed per workstation. Now the machine history is recorded, an incoming multi-machine workstation buffer uses it to send the test products to the correct machine.

The second machine input restriction is applied to diffusion machines. They require batches of products for input. The incoming buffer of the diffusion workstation remains basic by generating the batches in functions. Three options exist for selecting batches from a buffers contents. The first option is to search the contents directly for one of the allowed product combinations. If the combination is not present in the contents, then the next combination is searched for. This process is repeated until a batch is found or no combinations are left to search for. The second option is to split the contents of the buffer into separate lists for each product type and processing step. Next, a function determines an allowed combination of products from the lists, if such combination is present. The final option is to use a select expression to construct a list of all allowed product combinations from the contents of the buffer. This expression uses a function to determine the correctness of a batch. After the list has been constructed, a selection function selects a batch from it. The final option is chosen, because the programming of it results in the most compact formulation. After a batch has been determined, it is sent to one of the diffusion machines. To communicate products uniformly, all processes now send and receive products in a list. This can either be a list of three products for the diffusion machines or a singleton list for all other processes. Due to the batching restriction, the incoming buffers of the diffusion and the implantation workstations determine machine input of a different amount of products. The buffer process remains universal by including a parameter in the process that selects the correct function for generating the machine input. This concludes the restrictions on machine input.

Next, product transportation between areas is added to the basic flow line with machine input restrictions. The transportation can be added in three ways. Firstly, a transporter can be modelled that can always receive and administer requests from buffers for product transportation. At the start of a transportation cycle the product that will be transported is determined from the administered requests. Secondly, a transporter can be designed that selects at the start of its transportation cycle a product to transport randomly from all buffers that are trying to send a product to the transporter. Next, it performs its cycle sequentially. Finally, a transporter that is dispatched

by a transporter dispatcher process can be designed. The dispatcher administers the requests from buffers for product transportation. Again the transporter operates sequentially. The first design requires a complicated transportation process, because the transporter must always be able to receive requests from buffers. The second design introduces much non-determinism in the model. The final design is the most elegant one, because it contains only basic processes and excludes most non-determinism from this part of the model. Now the basic design has been chosen, the details are modelled. The transporter dispatcher receives requests for transportation of products from the generator collector and from the outgoing buffers. Furthermore, it dispatches the transporter to one of the requesting processes at the start of the transportation cycle. The transporter moves to the area of the requesting process. After the transporter has arrived, the process passes the product that needs to be transported to it and the transporter determines the drop area of the product. To this end the production sequence of the products is predefined in an array tuple of area identification numbers. The production step of the product is used to determine the drop area from this tuple. Finally, the transporter moves to the drop area and unloads the product into the incoming buffer of the area or into the exit process, dependent on the drop zone. Hereafter a new transportation cycle starts. Now, the machine input restrictions and the product transportation have been added to the model of the basic flow line.

3.4 Flow line with operators

In this section the operators are added to the flow line with machine input restrictions and product transportation. Three options exist to model the operators. Firstly, each operator can be modelled as a process that receives requests for operators from the incoming buffers and the machines. Furthermore it receives and sends products during loading and unloading and determines its off time. Secondly, the requests can be received by a personnel dispatcher. The dispatcher also dispatches the operators, which are individual processes, and determines their off time. Finally, operators can be modelled not as processes, but as pieces of information. The information contains the strictly necessary elements to model the operator. A dispatcher again receives the requests for operators. When an operator is dispatched by the personnel dispatcher to a process in a workstation or to an off time process, the information is sent to the process. When an operator is loading or unloading a machine, the information is passed either from the buffer to the machine or vice versa together with the products that are loaded or unloaded. The first option requires a complicated communication structure between the individual operators and the processes requesting the operators, especially for the implantation workstation because both operators can serve there. Furthermore, it requires complicated operator processes. The second option still requires complicated operator processes. The third option is the most compact and insightful one and also requires only basic processes. It is therefore chosen.

Now the basic design has been chosen, the details can be modelled. The operators are

represented by an array tuple containing their identification number and their position. When the operators are idle, the information is present in the dispatcher process. Next, the dispatcher receives a request for an operator from an incoming buffer or a machine and determines the correct operator to dispatch to the process by using a function. Then the operator is sent via the personnel transporter process, which determines and applies the operators transportation time, to the requesting process. After operating in the workstation the operator is returned to the personnel dispatcher. Apart from dispatching the operators to the workstations, the dispatcher also initiates the operator off time by sending the operators to the off time process. The dispatcher knows when it may send an operator to the process by administering the allowed off time. After a predefined time has passed, the off time counter is raised by one and the operator can take his off time. Meetings and breaks are modelled similarly, but have individual counters so they can be given different priorities. A number that represents the type of off time is therefore sent together with the operator to the off time process. After the off time, the operator is returned to the dispatcher. With FIFO control of the flow line the off time has a higher priority than the dispatching to workstations to assure all breaks are held and meetings are attended.

When an operator is dispatched to a workstation to load a machine, the operator receiving process is an incoming buffer. The buffer sends the operator and the machine input directly to the machine. If the machine is the lithography machine, then the setup time is determined with a function by using the type and processing step of the product and the setup is applied. The machine remains a universal process by adding the setup to it in a selection statement guarded by a workstation identification parameter. Next, all machines apply the loading time of the product similarly to the application of the processing time. So a machine first determines the loading time from an array tuple of loading times by using the production step of the product and then applies the loading. After the loading, the machine returns the operator to the dispatcher. When an operator is dispatched to a workstation to unload a machine, the operator receiving process is the machine. It applies the unloading time similarly to the application of the loading time and then sends the operator and the machine output to the outgoing buffer. The buffer directly returns the operator to the dispatcher. Now the operators have been added to the model.

3.5 Flow line with operators and a technician

In this section the technician is added to the model of the flow line. The technician is modelled similarly to the operators. His identification number and position are sent by the personnel dispatcher to the workstation areas via the personnel transporter and to the off time process. The technician has two tasks. The first task is to perform scheduled maintenance. If the maintenance window is modelled in the machine process, this process will become very complex. Therefore all windows are modelled in the dispatcher process. This process administers apart from the windows also the number

of scheduled maintenances that have to be performed. When maintenance has to be performed on an machine and the machine and the technician are idle, the personnel dispatcher sends the technician to the machine. To determine if a machine is idle, the machine status is administered by the dispatcher. Furthermore, the dispatcher prevents operators from loading a machine that is receiving maintenance. After the machine has applied the maintenance time similarly to the loading time in the previous section, the technician is returned to the dispatcher. If a machine has not received maintenance on time, the dispatcher prevents operators from loading the machine until the maintenance has been performed.

The second task of the technician is to perform unscheduled maintenance. The ion implantation machines break down during processing. The break down is initiated by the finishing of an unscheduled down timer. Two options exist to prevent a machine from going down while it is not processing. The first option is to flag if the timer is finished while the machine is not processing. The break down then occurs immediately after the machine starts processing. The second option is to freeze the unscheduled down timer while the machine is not processing. The second option is chosen, because it represents the factory more accurately. Because the timer is paused when the machine is not processing, the start value of the timer does not equal the time between failure. Instead the value equals the time between failure adjusted by a parameter to model the correct number of downs. When a down occurs, the personnel dispatcher receives a request from the down machine for the technician. After the machine receives the technician, the machine applies the repair time and returns the technician to the dispatcher. Restrictions on unscheduled down time distributions are omitted in the case. To get an evenly spread distribution of down times, the uniform distribution is used in the model. Machine failure is added to the machine process similarly to machine setup, so the failure is added in a selection statement guarded by a workstation identification parameter. The process therefore remains universal. This concludes the technician and the χ modelling of the case elements. Now the design of the model has been presented, the χ model with push control and with and flow line elements that operate FIFO is shown in Appendix A together with the description of the model.

3.6 Analysis of the χ model

In the previous sections the χ model has been designed. In this section the model is analyzed. The entire case has been designed, except the constraints on the buffer capacities. These constraints require implementation of improved control to avoid blocking. They are omitted because the improved control is excluded from this model. Therefore, the analysis of the flow line at this stage of the project will be done without applying the buffer capacity constraints. Before the Intel case is modelled correctly, the parameter that adjusts the time between failure of the ion implantation machines needs to be determined. This is necessary to model the right number of unscheduled downs and is done every time the model is changed. Appendix B presents the adjustment proce-

dure. The value of the parameter is determined by counting the number of unscheduled downs, while varying the parameter until the number equals the desired value. Now the model has been adjusted, it can be validated. But first the non-determinism in the model is determined. The non-determinism occurs in the model when more than one process starts attempting to communicate with the same process at the same time. All communication will take place instantly, but the sequence of communication can not be determined in advance. The non-determinism occurs in four sections of the model. Firstly, it occurs when two signal generators try to signal the collector at the same time. Secondly, it occurs when two processes request for product transportation at the same time. Thirdly, it occurs when two machines request for a product at the same time. Finally, it occurs when two processes request for personnel at the same time.

Before simulations are run, the model is validated. During validation the model behavior is compared to the flow line behavior prescribed in the case. The simulation results can only be trusted after validation. The validation techniques used in this project are 'transient calculation', 'steady state calculation', 'visualization using Gantt charts' and 'function analysis'. In transient calculation a few products are inserted into the flow line and the simulation output is compared to manual calculation. The outcome should be identical. In steady state calculation the model output in steady state is compared to manual calculations. Again the outcome should be identical. In visualization using Gantt charts the behavior of the flow line is visualized in Gantt charts and the occurring events are analyzed. Finally, the functions in the model are validated by analyzing the output of their input domain. The output should be correct over the entire domain. Appendix C shows examples of the used validation techniques. The model behaves as it was designed to behave and represents the case correctly.

After validation of the model, simulations can be run. In this project the most interesting behavior of the flow line is the steady state behavior. The first one hundred weeks of production are not used for the analysis of this behavior, so the influence of the transient state of the flow line can be neglected. The steady state behavior of the flow line with required input is presented in Appendix D. From this appendix results that the bottleneck machine of the flow line is the lithography machine. Furthermore, the capacity of the product transporter is sufficient for this flow line and a throughput (δ) is reached of 47.4 products per week or 56.4% of the required δ. The mean flow time (φ) of the products is increasing, because the work in process (w) is increasing. Finally, the control of the flow line is analyzed. The processes have much idle time and the machines have to wait for loading and unloading. Furthermore, the lithography machine has much setup time and all machines have a poor reaction to the unscheduled down of an ion implantation machine. It can be concluded that the control of the flow line is poor.

Four important control topics need improvement. Firstly, the personnel of the flow line operates FCFS (First Come First Served), whereas the priority of personnel should be directed at assisting the bottleneck machine. Secondly, personnel off time is scheduled independently of the personnel need of the machines, whereas it should be scheduled

when personnel is least needed. Thirdly, the setup time of the lithography machine should be minimized by releasing the right type of products into the flow line and by changing the control of incoming buffers. The buffers operate at the moment FIFO (First In First Out), but they should send the optimal available input to the succeeding machine. Finally, unscheduled down events should be handled by improved control.

The implementation of improved control in the flow line, has to increase the reached δ of the line by 43.6% to meet the required δ. However, the amount of work performed by the line has to increase less than 43.6% to meet the required δ, because some work is wasted on the increasing amount of w. The productivity of the flow line is defined as the total amount of work performed by the line divided by the required amount of work and is calculated in Appendix E. The productivity equals 81.5% and thus the required increase of productivity equals 18.5%.

Now the maximal throughput (δ_{max}) that can be reached with FIFO control shall be determined. It equals δ when w is only just non-increasing in steady state, because then no work is wasted on the semi-finished products that form the increasing w. To determine δ_{max}, the input of the flow line is decreased until w does not increase during simulation. The ratio of the product types is kept identical to the ratio in the case. δ_{max} and the accompanying φ are determined in Appendix F. δ_{max} equals 57.5 products per week or 68.5% of the required δ. φ equals 4580 min when 57.5 products are released per week. The productivity of the flow line with required input is greater than δ_{max}, because the flow line with required input allows the non-bottleneck machines to perform more work than the bottleneck machine. This work is wasted on the increasing w. Whereas δ_{max} is limited to the maximal work of the bottleneck machine.

Now the final analysis of the flow line will be performed. Appendix G shows that Little's law can be applied to the flow line with non-increasing w. This concludes the analysis of the flow line together with the χ modelling of the case. In the next chapter the feasibility of the control problem is determined.

Chapter 4

Control problem analysis

The χ model of the Intel case has been designed in the previous chapter. Before improved control will be added to the model in the next chapter, the feasibility of the control problem is determined in this chapter. So, it is determined if the required throughput of 84 products per week can be met. Initially the stochastics of the flow line are neglected in the analysis. In Section 4.1 the manual analysis is presented. Capacity calculations of the flow line elements are performed and a schedule is constructed manually. In Section 4.2 the computer aided analysis is described. The analysis is performed by optimizing a linear programming model of the case. Finally, the influence of the stochastics of the flow line is considered in Section 4.3.

4.1 Manual analysis

To provide a first impression of the feasibility of the control problem of the Intel case, the problem is analyzed manually. The manual analysis consists of two parts. Firstly, the capacity of the flow line elements is analyzed and secondly, an educated schedule is constructed manually. For the analysis of the capacity, the required and the available capacity of the flow line elements are calculated. Appendix H shows that the required capacity is calculated for the exact number of products that needs to be processed, accounting for wasted products in the implantation workstation. The buffers are assumed to have sufficient capacities when improved control is implemented. The required capacity of the machines, the transporter and the personnel is less than their available capacity. Therefore, the available capacity of all elements is sufficient for the control problem. However, only one personnel availability issue has been combined with the machine availability issues, because it is the only personnel issue, that influences machine availability for certain. The lithography machine will have at least three idle periods of 5 min per shift, because operator 2 has three off time periods per shift, that are 5 min longer than the maximum processing time on the machine. An example of an

availability issue that has not been included in the availability calculations is, that operators can not serve more than one machine at the same time. Therefore, in a feasible schedule the machines will have idle time that is not included in the calculations.

In the second part of the manual analysis all availability issues have been combined and an educated schedule is constructed manually. Constructing a schedule of one week is very time consuming. Therefore, instead of scheduling 84 products in one week, 6 products are scheduled in one shift. In this schedule the stochastics of the flow line are neglected, but unscheduled maintenance could be included in the model as scheduled maintenance. The maintenance time would be the mean repair time. Because only one shift is scheduled, the maintenance time would have to be scaled. The scaled maintenance represents the unscheduled maintenance behavior poorly and is therefore not included in the schedule. Because no products are wasted by the ion implantation machines, some surplus production is scheduled. The correct number of products that have to be produced per shift has been determined in Appendix H. The surplus production equals 2.6% of the correct production. The surplus production will not be taken into account during this analysis. The production will be scheduled cyclicly, which means that the schedule returns at the end to its initial state. It can therefore be repeated without adjustments. If the schedule of one shift is repeated 14 times, a schedule of one week is constructed. Therefore, the analysis of a schedule of one shift is indeed sufficient for the analysis of the control problem. To simplify the construction of the schedule, some assumptions are made. Firstly, the capacity analysis of Appendix H shows, that the capacity of the product transporter is sufficient for this flow line. Also, the transporter operates independently of other resources in the line and the buffers in the line handle the stochastics in the product transportation requests. Without loss of generality it can be assumed, that the improved control that will be implemented in the next chapter assures the incoming buffers contain enough products to feed the machine at any time. Therefore, the product transportation can be neglected in the schedule and products are always available. Secondly, the improved control is assumed to assure that the incoming buffers of the flow line contain correct machine input. Therefore, restrictions on machine input can be neglected in the schedule. The assumptions result in a schedule of the loading, the processing and the unloading of six low and six high production steps per workstation. Furthermore, the setup of the lithography machine is included in the schedule. Appendix H shows that the average setup time equals 24 min per shift. The setup is scheduled as two individual setups of 12 min upon production step change. Finally, the scheduled maintenance of the machines and the personnel off time are included in the schedule. All jobs are scheduled on the machine bars and the personnel bars of a Gantt chart. During scheduling, the objective was to construct a cyclic schedule which fits in a shift of 720 min. Figure 4.1 shows the legend of the manually constructed schedule and Figure 4.2 shows the schedule.

Figure 4.2 shows all processes have little idle time, except the ion implantation machines and the technician. This results from neglecting machine break down in the schedule. The length of the cyclic schedule equals 700 min and fits in a shift. Therefore, the control problem without machine break down is feasible. The lithography machine

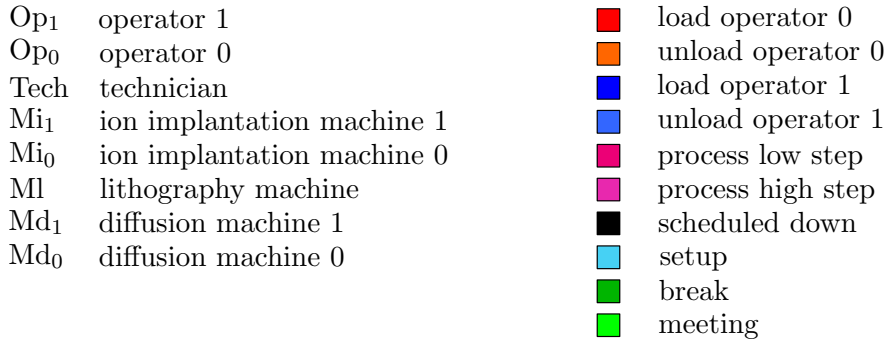


Figure 4.1: *Legend of the manually constructed schedules*

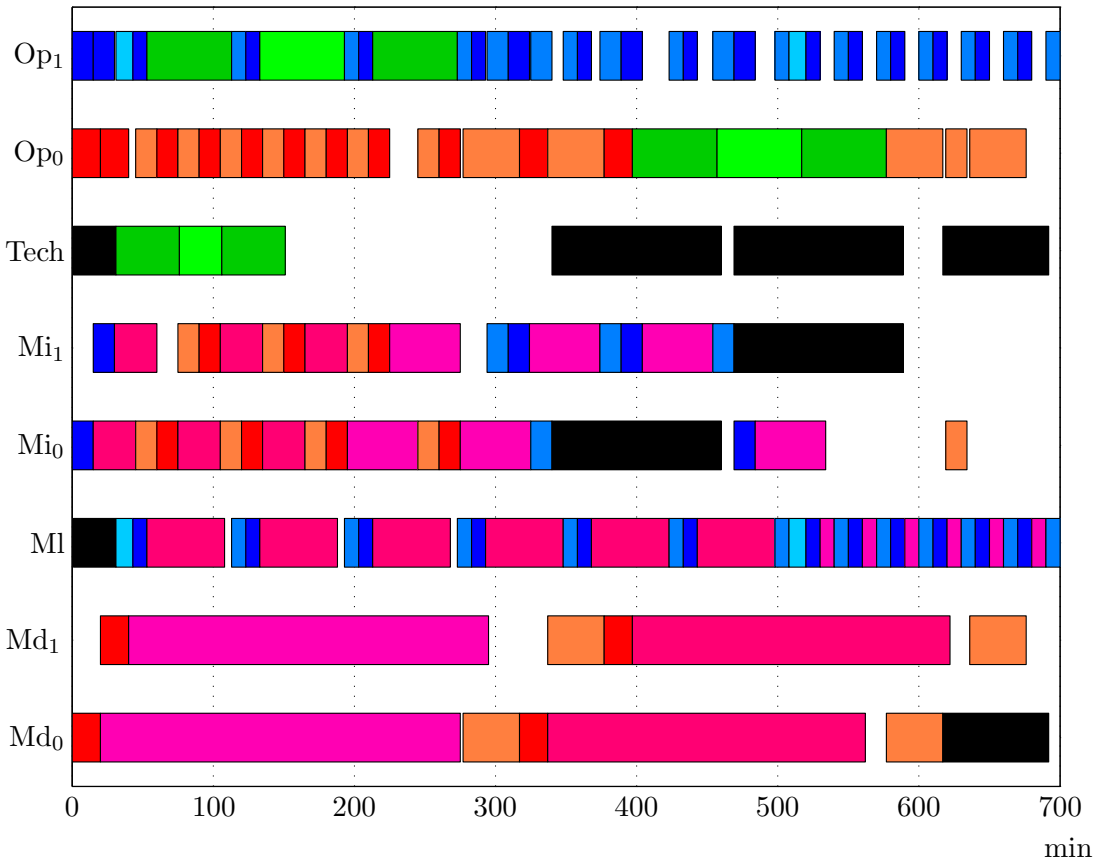


Figure 4.2: *Manually constructed schedule*

is fully utilized in this schedule and is the bottleneck of the flow line. Therefore, the length of the schedule is minimal. As was mentioned in the first paragraph of this section, the lithography machine has three idle periods of 5 min per shift, because operator 1 has three off times that are 5 min longer than the maximum processing time on the machine. Finally, the product transportation and the buffer levels are analyzed

in Appendix I to confirm, that transporter and buffer capacities are indeed sufficient for this schedule. An example transportation scenario is used for this analysis. The transporter and buffer capacities are indeed sufficient for the schedule of Figure 4.2.

4.2 Computer aided analysis

In the previous section it has been determined by manual analysis that the control problem without machine break down is feasible. Now, it will not be as easy to determine the feasibility of the control problem with deterministic (planned) machine failure by constructing a schedule manually. Also, multiple shifts have to be scheduled, to include the break down behavior correctly. Therefore, a linear programming model of the flow line, that is ready for optimization, is designed in this section. The model will provide a lower bound on the length of the schedule of one weeks production. The model has to be linear, because the size of the Intel case results in a model that is too large, even with simplifications, for other solving techniques than linear ones.

In the linear model the production of 84 products per week is modelled and machine break down is included in the model. But no products are wasted when a break down occurs. So, like in the second part of the manual analysis, some surplus production is scheduled. Section 4.1 shows this surplus production equals 2.6% of the correct production. Again, the surplus production is not taken into account during the analysis.

To reduce the number of variables and constraints in the model, while representing the crucial parts of the case, the following modelling assumptions are made. Firstly, in the previous section it has been determined, that the capacity of the product transporter is sufficient for the flow line. Therefore, the transportation of products is not considered in the model. The previous section also showed, that the buffer capacities will be sufficient when the control of the flow line is improved and therefore the buffers do not have to be included in the model. It was assumed in the previous section, that improved control assures that the incoming buffers of the flow line contain correct machine input on time. Therefore, the processing steps can be scheduled without release dates and precedence relations for the processing sequence of the products. Only the total number of jobs scheduled for each production step needs to be correct. Without modelling the transporter and the precedence relations, the modelling of the start and the exit area is not necessary. Because the correct machine input is handled by improved control, machine input restriction do not have to be modelled. Furthermore, it is assumed that personnel transportation can be neglected, because it is assumed that improved control assures the bottleneck machine receives personnel on time. The other machines are less crucial to the length of the schedule and the transportation times are substantially smaller than the times of the tasks of personnel and the off times. Finally, all scheduled maintenance can be combined with an unscheduled maintenance without violating the scheduled maintenance window restrictions. Therefore, the window restrictions are neglected initially. If they are violated in the resulting schedule, they can be added to

the model.

With the presented assumptions two models can be used to represent the case. The first model [5] uses variables to assign jobs to positions on a machine and to assign start times to the positions of the machine. For example, job 7 is scheduled at position 10 of a machine and position 10 starts after 200 min. The second model [6] is a time-indexed model. It uses variables to assign start times of jobs to discrete time intervals on a machine. For example, job 7 starts on a machine after 200 min. Both models are focussed on the scheduling of jobs on machines and require additional constraints to prevent personnel from performing more than one job at a time. Because all personnel operate on multiple machines, it is necessary to know which job is processed at what time. In the first model the processing time of a job can only be coupled to the identification of the job via the processing position on a machine. The coupling therefore requires many extra variables and constraints. The time-indexed model is better suited to represent the case, because the coupling is performed directly by the variables. Therefore, a time-indexed model is designed.

Now the type of the model has been determined, an attempt is made to further reduce the number of variables and constraints by simplification techniques, while still representing the crucial parts of the case. The reduction is necessary to allow solving of the model. Firstly, in the modelling assumptions it was explained that product and personnel transportation can be neglected. All jobs except the setup jobs that have been determined in the previous section, have a duration that are a multiple of five. The setup jobs are given in this section a duration of 10 min, so a time discretization step can be taken of 5 min instead of 1 min. Secondly, instead of scheduling three jobs per machine run, one for loading, one for processing and one for unloading, only two jobs are scheduled. Because the loading of a product is always directly followed by the processing of the product, the loading and processing can be combined into one job. This combined job has a duration for the operators of only the loading time and for the machines of the total of the loading time and the processing time. The unloading job has a duration of the unloading time for both the operators and the machines. Next, instead of scheduling individual jobs for each production step of each product one time, general jobs can be scheduled. Each general job represents a production step in the flow line and is scheduled once for every product that flows through the line. Furthermore, instead of modelling five individual machines, the identical machines of the line are modelled as one machine with doubled capacity. As a consequence, every job has one machine on which it can be scheduled. Next, the operators are modelled without employment area restrictions. This simplification prevents, as calculated in Appendix H, per shift 15 min idle time of the lithography machine. Because if one operator is having off time, the other one remains available to serve the machine. This is accounted for in the length of the schedule. The advantages of this simplification for the diffusion machines are negligible, because one operator spends almost all of his time serving at the lithography workstation. For the implantation workstation the simplification provides no advantages. Because both operators are now identical they can be modelled as one operator with doubled capacity, similar to the identical

machines. As a consequence, all jobs are loaded and unloaded by one operator. Finally, the setup time of the lithography machine is included in the appropriate loading jobs by prolonging them.

After the reductions of the model have been discussed, the principle modelling elements are presented. Two types of variables are used in the model. The first type represents the main variables of the model x_{jk}. The value of the variables equals 1 if job j is started at time k and 0 otherwise. The second type is the minimization variable z. It equals the maximum completion time of all jobs. Next to variables, the model also contains constraints. The first type of constraints, used to assure that all general jobs j with process time p_j are processed for the required amount of times b_j, before the maximal length of the schedule T has passed, is

$$\sum_{k=0}^{T-p_j} x_{jk} = b_j \qquad \forall j.$$

The second type of constraints, used to prevent machines and personnel from scheduling more jobs at a time than their capacity b_S allows, is

$$\sum_{j \in S} \sum_{l=k+1-p_j}^{k} x_{jl} \leq b_S \qquad \forall k, S. \tag{4.1}$$

From all the jobs of the subset of jobs S that are processed on the same machine or handled by the same personnel, there can only be b_S jobs in process or being handled at time k. The duration of the job p_j can be different for a machine and an operator, as described in the previous paragraph. The third type of constraints, used to introduce a relaxed precedence relation for the processing and unloading of the products in the model, is

$$\sum_{k=0}^{T-p_j} (k + p_j) \cdot x_{jk} \leq \sum_{k=0}^{T-p_{j+1}} k \cdot x_{j+1,k} \qquad \forall j.$$

The precedence relation is relaxed, because the relation is applied to general jobs instead of individual jobs. So, on average the processing jobs j are completed earlier than the unloading jobs $j+1$ are started. The fourth type of constraints is

$$\sum_{k=0}^{T-p_{j+1}} k \cdot x_{j+1,k} \leq \sum_{k=0}^{T-p_j} (k + p_j) \cdot x_{jk} + \Delta \qquad \forall j.$$

It resembles the third type and prevents that first all processing jobs are scheduled and next all unloading jobs. The constant Δ allows a small time interval between the completion of the processing jobs j and the start of the unloading jobs $j+1$, because the operator will not always be directly available for unloading. The fifth type of constraints, used to equate the variable z to the maximum completion time of all jobs, is

$$z \geq (k + p_j) \cdot x_{jk} \qquad \forall j, k.$$

Now, z represents the length of the schedule and will be minimized. The final type of constraints, used to restrict the main variables x_{jk} to integer values and to present them a lower and an upper boundary, is

$$x_{jk} \in \{0, 1\} \qquad \forall j, k.$$

The complete integer linear programming model that is used for optimization is shown in Appendix J.

After the design of the mathematical model, attempts are made to solve the model with branch and bound techniques and with the program *LP-solve*. The attempts have failed, because the number of variables (40,000) and constraints (50,000) and also the number of variables in the constraints (700,000) is too great. Next, the model has been reduced to schedule one shift instead of one week. The time to repair the ion implantation machines is therefore scaled. In this way, the break down behavior is not modelled well, as described in the previous section. After the reduction remain 3,000 variables, 4,000 constraints and 50,000 variables in the constraints. The resulting model is still too complex to solve integer with the available tools.

A new model has been designed to further reduce the number of variables and constraints. In this model only one job is scheduled per machine run. The constraints for the operator (4.1) are now replaced by

$$\sum_{j \in S} \left(\sum_{l=k+1-p_j}^{k+1-q_j} x_{jl} + \sum_{l=k+1-r_j}^{k} x_{jl} \right) \leq 2 \quad \forall k. \tag{4.2}$$

The set of jobs S in (4.2) includes all jobs performed by the operator. The run time of job j, thus the total loading, processing and unloading time, is represented by p_j and q_j is the total processing and unloading time of the job. Finally, the unloading time of the job is represented by r_j. The operator is forced to perform unloading directly after processing and it will therefore be more difficult to schedule the jobs in the given time period than with the previous set of constraints. The lithography machine could present a problem, because in the Intel case the machine is served by only one operator and the operator off time duration is greater than the maximal processing time. Therefore, the operator can not take off time during processing on the machine. This forces much idle time on the lithography machine. The identical operators that are modelled according to the model reductions of this section are the solution to the problem. One operator can take off time, while the other operator unloads the lithography machine on time. Now, the number of variables is reduced to 2,000 and the number of variables in the constraints becomes 35,000. The number of constraints remains equal. The problem is still too complex to solve integer and therefore, this model is not used anymore. The first model of this section is used for the remainder of the solving process, so again one week is scheduled. The model is relaxed, thus the integer constraints are neglected, to allow optimization. The minimization variable z is not sufficient anymore to determine the minimal length of the schedule, because the main variables do not necessarily have

integer values. The constraints for z are therefore neglected and the minimization is performed by hand. But first, the model is validated by comparing the number of scheduled general jobs, the scheduled machine and personnel capacity and the difference between the loading and unloading time of the general jobs to the required values. The validation shows that the model behaves correctly.

Now, the optimization will be described. The mathematical model is build in *Matlab* and is solved using the *linprog* function from the optimization toolbox. The object function is not provided in the function call and therefore, any feasible schedule is a solution to model. By reducing the maximal finish time of any job until the problem becomes infeasible, the minimal length of the schedule is found. The minimal length equals 0.94 weeks. The lithography machine is fully utilized in this schedule and is therefore the bottleneck of the flow line. The minimal length of the schedule should be prolonged with 15 min per shift, because in any feasible schedule the lithography machine has 15 min idle time per shift, while the serving operator has off time. Another 4 min are added to the length of the schedule to account for the rounding of the setup time, as explained in Section 4.2. The resulting length of the schedule equals 0.97 weeks. In the resulting schedule the lithography machine is again the bottleneck of the flow line and is fully utilized. The length of the schedule is therefore indeed minimal. The schedule is relaxed and therefore not feasible, but all machines except the lithography machine have some surplus capacity. It is assumed that the surplus capacity is sufficient to allow an integer solution. Because the model is solved without the integer constraints, the scheduled maintenance jobs are spread all over the scheduled maintenance variables. Therefore, all maintenance window restrictions improve the model negligibly and remain excluded from the model as was done initially in this section. According to this analysis, the control problem with deterministic (planned) machine break down is assumed to be feasible.

4.3 Stochastics analysis

In the previous section it has been determined, that the control problem with deterministic (planned) machine break down is assumed to be feasible. Now, the stochastics of the flow line will be added to the analysis. So, the ion implantation machines break down with stochastically distributed repair time and time between failure. The stochastics do not influence the required capacity of the flow line, as calculated in Appendix H, but do influence the available capacity of the machines. An ion implantation machine breaks down without warning and therefore, a choice needs to be made upon break down to directly repair the broken down machine or to firstly perform scheduled maintenance on other machines. If repair starts directly, the scheduled maintenance of the machines can not always be performed on time and therefore, all machines will have less available capacity on average, even the bottleneck machine. It is therefore preferred, that scheduled maintenance is performed before the repair starts. Now the broken down machine loses extra available capacity, but the other machine will loose less capacity.

The improved control, that will be implemented in the next chapter, determines which machines require scheduled maintenance before the repair starts. The control can be designed to be account for the buffer levels. The improved control can furthermore be designed to perform all scheduled maintenance as early as possible in the shift, so the available capacity of the machines is least effected by the unscheduled down. The diffusion machines in general are not effected much by the unscheduled down, because they only receive scheduled maintenance once per day. If the ion implantation machines have enough spare capacity, it can be concluded, that the control problem with stochastic machine break down is feasible. An educated guess is made of the extra loss in availability of the broken down machine. The guess is based on the scheduled maintenance time of the lithography and the ion implantation machines and accounts for the chance that scheduled maintenance has already been performed during the shift in which a machine breaks down. The loss in availability equals 120 min per break down. A prolonged break down is modelled in the linear programming model of Appendix J. The resulting minimal length of the schedule of one weeks production equals again 0.97 weeks. It will be less easy to change the relaxed schedule in a feasible one than it was in the previous section, but the schedule can be prolonged by 0.03 weeks and also the 2.6% surplus production that is scheduled can be omitted in the actual schedule. The control problem of the case is assumed to be feasible, according to this analysis. To analyze the sensitivity of the outcome of the analysis for the guessed loss in availability of the ion implantation machines, the maximal allowed loss in availability upon break down is determined. The maximal loss equals 225 min and therefore, the outcome of the analysis is insensitive to the made guess. Now the control of the flow line of the case will be improved in the next chapter.

Chapter 5

Improved control of the flow line

In the previous chapter it has been determined, that the control problem of the Intel case is likely to be feasible. It will however be difficult to implement control in the flow line that results in the required throughput, if such control exists. Currently in the χ model the most simple control, which is push control, is implemented. Furthermore, the products are released into the flow line without an educated control of the product type. Each type has its own constant inter departure time. Next, the transporter, buffers and personnel operate FIFO (First In First Out). This should be interpreted for the diffusion buffer, that it passes the first available batch. Furthermore, personnel take off time as soon as they are allowed to. Finally, unscheduled down events are treated like ordinary loading or unloading events without specific control. These were also the control topics that need to be improved according to Section 3.6. In Section 5.1 the design of the improved control is described. Next, in Section 5.2 the designed control is implemented and finally the simulation results of the flow line with improved control are analyzed in Section 5.3.

5.1 Design of improved control

In this section the design of improved control of the flow line is described. The control initially remains push control. The control problem becomes deterministic, when machine break down is neglected. The improved control for the deterministic flow line can be designed as a repetitive schedule. For example, the manually constructed schedule from Section 4.1 can be implemented. The required throughput will be met with this control and the buffer capacity constraints will not be violated. When machine break down is not neglected, the stochastics of the flow line prevent the implementation of a repetitive schedule. Instead heuristic sequencing rules can be applied. The rules are derived from analyzing the behavior of the flow line and they are mostly directed to aiding the bottleneck of the flow line, which is the lithography machine. This machine should be utilized maximally. Next, the ion implantation machines should be utilized as

much as possible, because they loose the most available capacity due to the stochastics of the flow line. Finally, the diffusion machines are served with least priority, because they are least affected by the stochastics of the line and have more surplus capacity than the lithography machine.

The control of the flow line is improved in small steps to acquire elegant control and to make it easier to validate the model. The first control topic to be improved, is the type of the products that are released into the flow line. First all commercial products of type A are released and next all products of type B. The test products are released in between, to monitor the quality of the flow line correctly. Due to the scheduled down of the ion implantation machines, the final batch of the high production step of each series of commercial products can perhaps not be completed by the diffusion buffer. Therefore, the incomplete batch will have to wait in the buffer until the new series of products has been processed and the old series is started again. So, a few products will have very long flow times.

The next control topic is the policy of the incoming buffers. The buffers try to present the machines series of six products of equal production step. In this way, cyclic behavior is implemented into the flow line. Furthermore, the buffer of the lithography workstation processes test products last in a series of equal step, because type and step setup can then be combined. Now, the lithography machine has least setup time.

The third control topic is the control of personnel. Operator 1 is the most important operator, because he serves the lithography machine, which is the bottleneck of the flow line. Therefore, his control is determined firstly. His primary task is to serve the lithography machine. But when the machine processes a product for the low production step, operator 1 has time to serve the ion implantation machines or to take off time. He serves the machines when he can start serving within the next 8 min or when he has no off time left and otherwise he takes off time. When he starts serving within the next 8 min, he has time to perform three load or unload tasks before he has to return to the lithography workstation. He has to return on time to prevent extra idle time of the lithography machine. When scheduled maintenance is performed on the lithography machine, operator 1 can also serve at the implantation workstation. Again the operator returns on time to prevent idle time of the lithography machine. Next, the control of operator 0 is designed. He serves mainly the implantation workstation when operator 1 does not serve the workstation. Furthermore, operator 0 serves the diffusion machines when he can not start serving an ion implantation machine directly or operator 1 is present at the implantation workstation. Operator 0 takes off time when he can not serve an ion implantation machine or a diffusion machine within the next 20 min. Both operators prioritize loading tasks above unloading tasks inside a workstation to increase the throughput. Finally, the control of the technician is designed. He performs scheduled maintenance on the diffusion machines and the lithography machine as soon as possible and on the ion implantation machines during a high processing step series on the lithography machine. The technician takes off time when he can not perform maintenance during the next 30 min.

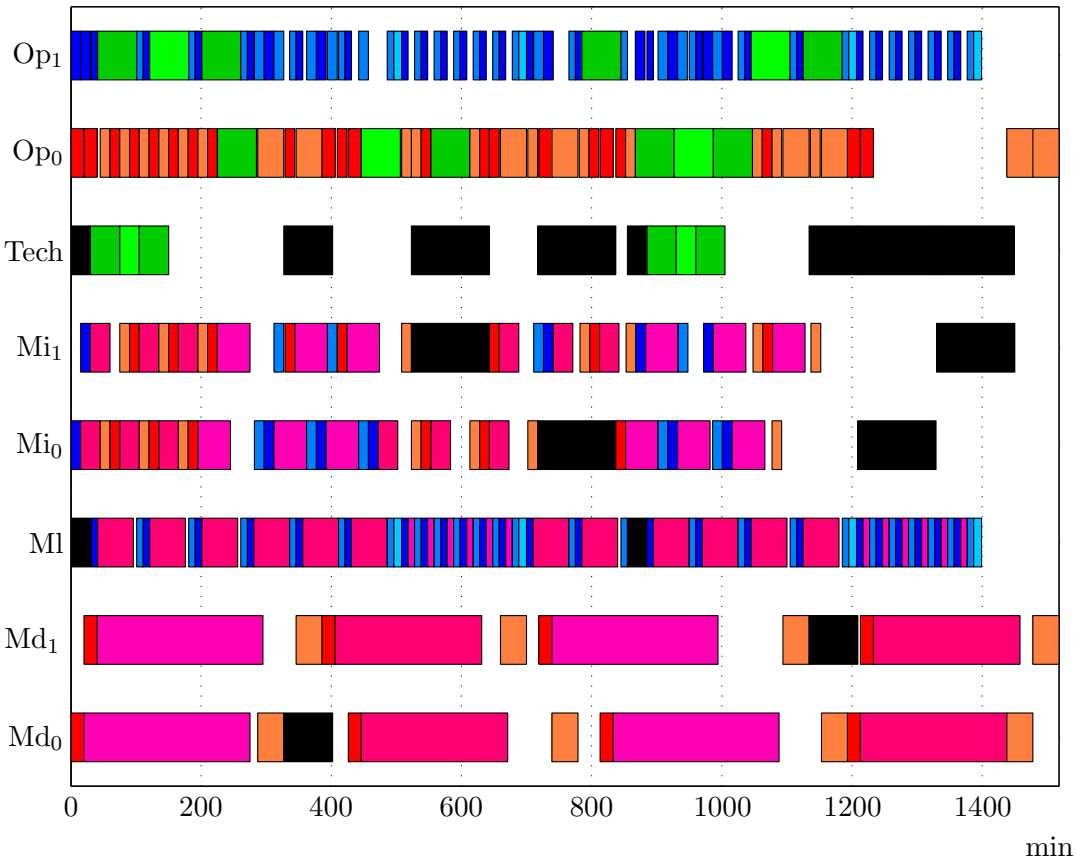


Figure 5.1: *Manually constructed schedule using improved control strategy*

After the design of the third topic, a schedule similar to the schedule of Section 4.1 is constructed manually to analyze the behavior of the control. Machine break down is neglected in the schedule and the production for two shifts is scheduled. Figure 4.1 shows the legend of the manually constructed schedule and Figure 5.1 shows the schedule. The lithography machine is not the bottleneck of the flow line anymore and is finished producing after 1399 min. Furthermore, The ion implantation machines and the technician show much idle time, because machine break down is neglected in the schedule. The schedule also shows, that diffusion machine 1 is the bottleneck of the flow line and is finished after 1518 min, which is also the cyclic length of the schedule. Therefore, the schedule does not fit in two shifts. However, due to the stochastics of the flow line, the personnel will make different operating decisions in each shift. Therefore, the idle time of the machines will vary per shift. Furthermore, like in Section 4.2, 2.6% surplus production has been is scheduled. So, the required production without machine break down can perhaps be produced with the designed control.

The fourth control topic is the control after an unscheduled down event occurs. When an ion implantation machine breaks down, it is not repaired directly, because the

other machines of the flow line could require scheduled maintenance before the repair is finished. Scheduled maintenance is performed on the diffusion machines and the ion implantation machine that did not break down, if the machines require maintenance during the next six hours after the repair starts. The machines will not be loaded until the scheduled maintenance has been performed. The maintenance is performed on the lithography machine if the machine requires maintenance during the next eight hours after the repair starts. This machine will be loaded for a new run if the technician does not perform the maintenance directly. Finally, when no scheduled maintenance has to be performed anymore, unscheduled maintenance is performed. The lithography machine and the diffusion machines receive scheduled maintenance as soon as possible, so it is likely they do not require scheduled maintenance before the unscheduled maintenance is performed. All times in the design of improved control can be adjusted if simulation results encourage adjustment. All times except the times involving operator 1 are determined by an educated guess.

The final control topic is the control of the work in process (w). When the flow line is controlled by push control, w can become very large and the line can become congested. When the buffer capacity constraints are violated to reach the required throughput with push control, pull or conwip (constant work in process) control could be the solution to the problem. Pull control should result, with equal throughput, in a smaller w, because the products are pulled out of the flow line. A sort of pull control for this re-entrant flow line could be designed by prioritizing the high production step. With conwip control, the work in process is kept constant. So, every time a product leaves the flow line, a new product is released into the line.

5.2 Implementation of improved control

In this section the improved control is implemented in the χ model. The implemented control topics of the previous section are shown in the χ model that is presented in Appendix K. It is reminded, that the parameter that adjusts the time between failure of the ion implantation machines needs to be adjusted every time the control or the input of the flow line is changed, as was mentioned in Section 3.6. The first improved control topic is the type of the products that are released into the flow line. The generator of the flow line without improved control is adjusted. The three signal generators are replaced by two signal generators, one for test products and one for commercial products. A counter in the collector process determines the type of commercial product that is released in the flow line whenever the signal generator for commercial products signals the collector.

The second control topic is the policy of the incoming buffers. The diffusion and ion implantation buffers record the production step of the current machine input series and the number of products in the current series. Furthermore, the input generating functions are adjusted to return machine input of the current production step series

if possible, until the maximal length of the series has been reached. Then, the other production step series is started. The lithography buffer records next to the current production step also the current product type. The buffer determines in an input generating function the most preferable input for the lithography machine, so the setup time is minimized, while the production is cyclic.

The third control topic is the control of personnel. To make the improved control easier to analyze, machine break down is neglected. Due to lack of time in the project, the control topic has not been implemented exactly as described in the previous section. Therefore, the implemented control will be described firstly. Operator 1 serves mainly the lithography machine. When the lithography machine performs a low production step, operator 1 may take off time. He serves at the implantation workstation, if there are no requests placed for service at the lithography workstation. Operator 0 serves mainly at the diffusion workstation, because the implantation workstation has more spare capacity than the diffusion workstation due to the neglecting of machine break down. Operator 0 serves the implantation workstation, when there are no requests placed for service at the diffusion workstation. He may take off time when both diffusion machines are busy for the next 20 min and there are no requests placed for service at the implantation workstation. The technician performs scheduled maintenance according to the previous section. He may take off time when he can not perform maintenance on the diffusion machines and the lithography machine during the next 30 min. In the implementation, the personnel dispatcher process is firstly adjusted to record the processing step of all machines and the time a machine finishes processing or scheduled maintenance. The dispatcher also records for all machines the earliest time the current machine run will be finished. With this information the dispatching function can be adjusted to prioritize the requests for personnel and the off time of personnel. The dispatching function is also adjusted to prioritize loading tasks above unloading tasks. Finally, the initiation conditions for scheduled maintenance are adjusted to represent the designed control of the previous section. Attempts to implement control as it was designed in the previous section, have shown that a lot of exception handling is needed to get good results. The attempts have also shown, that it is important to implement simple control, because it is easy to understand and to adjust if simulation results present reason for adjustment.

The fourth control topic is the control after an unscheduled down event occurs. This control has not been implemented yet, again due to lack of time in the project. The final model of the flow line with improved push control and without machine break down is presented in Appendix K.

Finally, pull and conwip control are implemented to analyze their effect on the maximal throughput and the buffer levels of the flow line. The designed pull control of the previous section is implemented by adjusting the machine input selection functions of the incoming buffers. Now, the buffers always try to send a product with the high production step to machines. Next, the designed conwip control is implemented. Every time a product leaves the flow line, a new product is released into the line. This control

requires three communication channels between the collector process of the generator, which releases the products into the line, and the exit process and ion implantation machine processes, by which a product leaves the flow line. The signal generators are now superfluous and the product type is determined by the collector process in a function. This concludes the implementation of improved control.

5.3 Analysis of improved control

The simulation results of the flow line with improved control are analyzed in this section. When the flow line is controlled by push control and the flow line elements operate FIFO (First In First Out), the maximal throughput (δ_{max}) equals 57.5 products per week and the mean flow time (φ) equals 4580 min, as described in Section 3.6. The accompanying work in process (w) equals 26.1 products. After each implementation of control, the model is validated. The validation is performed according to the validation in Section 3.6 and the flow line behaves correctly after each implementation of improved control.

After implementation of the first control topic, the type of the products that are released into the flow line, simulations are run to evaluate the control. Like in Section 3.6, the steady state behavior of the flow line is analyzed. The first one hundred weeks of production are not used for the analysis, so the influence of the transient state of the flow line can be neglected. Without applying the buffer capacity constraints, δ_{max} equals 58.1 products per week. Only a small increase in δ_{max} results from this improvement of control, because the input for the bottleneck of the flow line, the lithography machine, switches much between low and high production step and therefore, the setup time for the lithography machine is still great. The average φ has decreased, because the high production step batches of the diffusion workstation are completed earlier. φ equals 4510 min. w equals 26.0 products.

Improvement of the second control topic, the policy of the incoming buffers, increases δ_{max}, because it decreases the setup time of the bottleneck machine much. Without applying the buffer capacity constraints, δ_{max} equals 60.4 products per week. φ decreases to 4340 min, because the setup time has decreased. w remains fairly constant and equals 26.2 products.

Improvement of the third control topic, the control of the personnel of the flow line, is done in two steps. Firstly, the designed control of the off time of operator 1 is implemented. Furthermore, the production sequence of the products is adjusted, so the products only enter the lithography workstation twice and then go to the exit. Now, it is analyzed if the off time control is sufficiently improved to allow the bottleneck of the flow line, the lithography machine to process the required input. Simulations show that the control is sufficient for the lithography machine to process all the input. Figure 5.2 shows a Gantt chart of the lithography machine and operator 1. Figure C.1 shows the legend of chart.

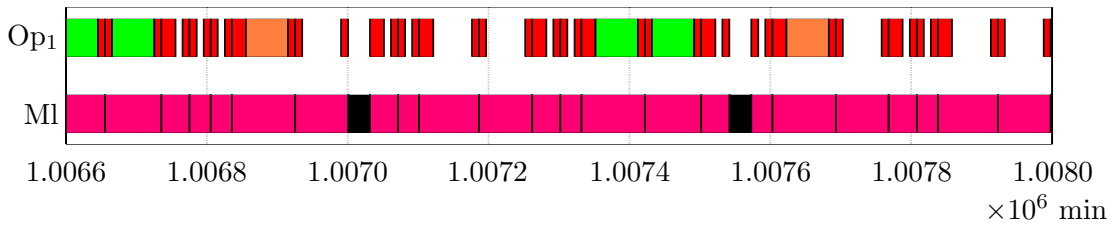


Figure 5.2: *Gantt chart of the lithography machine and operator 1 during the last two shifts of one hundred weeks of production*

Figure 5.2 shows that the lithography machine changes its production step much. This occurs because the machine has only few products in its incoming buffer. Furthermore, the lithography machine has no idle time. However, in the complete model some products will be wasted and when the lithography buffer contains more products, it will change the production step of the machine input less often. Figure 5.2 also shows that operator 1 has four periods per shift in which he can serve at the implantation workstation. Three periods coincide with the processing of the products of low production step and one period coincides with the scheduled maintenance of the lithography machine.

The second step in the improvement of the control of personnel, involves the entire control as it was described in the previous section. Now, the break down behavior of the ion implantation machines is neglected. δ_{max} equals 80.4 products per week. φ and w respectively equal 3204 min and 25.6 products. Simulations show that the buffer capacity constraints of the lithography and the implantation buffer are violated at different times. Furthermore, the number of off times is analyzed and all off time is taken. Not necessary three periods of off time are taken per shift. When 81 products are released per week the contents of the lithography and the implantation buffers keeps increasing. The buffers contain approximately an equal amount of products. Therefore, both the lithography and the implantation workstation are the bottleneck of the flow line. Figure 5.3 shows a Gantt chart of the activity of the machines, the personnel and the transporter during the last four shifts of one hundred weeks of processing with flow line input of 80.4 products per week. The legend of the chart is presented in Figure C.1. The Gantt chart clearly shows, that the processes have little idle time, except the ion implantation machines and the technician, because machine break down is neglected. The flow line behaves well with the implemented control. It appears from this chart, that the lithography machine is fully utilized, but other Gantt charts have shown idle time of the machine. The chart also shows, that all scheduled maintenance is performed on time. Finally, personnel take enough off time per shift, but the off time is not necessarily evenly spread over the duration of the shift. All processes show idle time in the Gantt charts and therefore, the throughput of the flow line can probably be improved by implementing more suitable control in the flow line.

The previous paragraph showed, that the buffer capacity constraints are violated when 80.4 products are released into the flow line per week and machine failure is neglected. When machine failure is not neglected, the buffer capacity constraints will

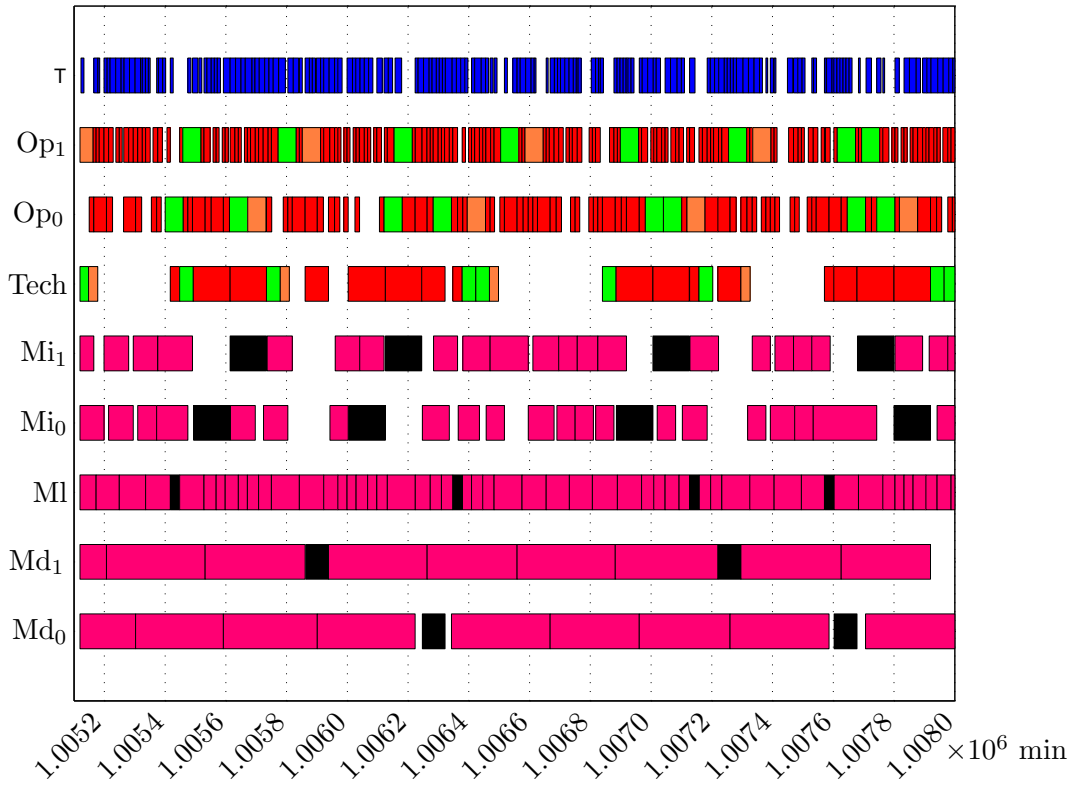


Figure 5.3: *Gantt chart of the production of the last four shift of one hundred weeks of processing with improved push control*

be violated even more and push control will probably not be sufficient for the flow line. Therefore, pull and conwip control, as described in the previous section, are analyzed. The push control of the model of the previous paragraph is firstly replaced by pull control for the re-entrant flow line. δ_{max} equals 79.4 products per week, which is less than with the push control. The input for lithography machine changes often of production step with this control and therefore, the machine has a lot of setup time and has become solely the bottleneck of the flow line. φ and w respectively equal 3258 min and 25.5 products. Therefore, for the flow line of the Intel case, this type of pull control is not preferred above push control.

The push control is secondly replaced by conwip control. The control requires a w value and then determines automatically when to release a product into the flow line. If w equals 25 products, then δ_{max} is reached and equals 79.4 products. The accompanying φ equals 3160 min and is, due to the smaller w level, smaller than with improved push or pull control. Simulations show that both the lithography and the implantation workstation are the bottleneck of the flow line, as was the case with improved push control, because the buffers contain on average a high amount of products. Both buffers contain up to 14 products at different times and thus, the buffer capacities are insufficient for this w level. δ_{max} is a little smaller than with improved push control, probably because

the behavior of the flow line is less cyclic due to the irregular release of products into the flow line.

It can be concluded from this analysis, that for the flow line without machine break down and with the currently implemented control, the maximal throughput is reached by push control and not by pull or conwip control, when buffer capacity constraints are neglected. With the current improved control, the required throughput can not be reached, but not all designed control has yet been implemented. Therefore, the required throughput can perhaps be met with the designed control. This concludes the improvement of the control of the flow line.

Chapter 6

Conclusions

The complexity of controlling a re-entrant flow line is illustrated well by the Intel Five-Machine Six Step Mini-Fab case. It contains all difficulties that need to be dealt with in practice and the target of the case is to reach a throughput of 84 products per week despite the difficulties. The re-entrant flow line of the Intel case has been modelled in χ, because χ is well suited for modelling discrete-event systems like the line. Techniques for modelling elegantly in χ have been used to acquire an elegant model. For the construction of the model, first a basic flow line has been considered, that contains only a product generator, buffers, machines and an exit process. The generator uses signal generators to conveniently determine what type of product to release into the flow line. Next, to avoid non-determinism in a workstation in which a single buffer serves multiple machines, a product request structure is modelled. When a machine becomes idle it requests the buffer for input. Furthermore, the generation of machine input is moved to functions to keep the buffer processes basic. In the next model, machine input restrictions are added to the flow line in the functions that generate the input and also product transportation is added. The transportation is modelled by a basic transporter process that is dispatched by a transporter dispatcher. In the third model two operators are added to load and unload the machines and to perform setup on one of the machines. The operators have the same dispatching structure as the transporter. However, the operators are not modelled as processes, but as operator state information. The information is passed either to the flow line processes where the operators are performing a task or to an off time process when the operators are having off time. In the final model a technician is added to the flow line to perform scheduled and unscheduled maintenance. The technician is modelled similarly to the operators. The personnel dispatcher determines when scheduled maintenance is performed on the machines. The machines request for the technician when they break down. The flow line is controlled by push control and the flow line elements operate FIFO (First In First Out).

After the model has been designed, it is validated. The model behaves as it was designed to behave and represents the case correctly. After the validation of the model,

simulations can be run. The simulation results show that the required throughput can not be met with the current control of the flow line. The maximal throughput, ignoring the buffer capacity constraints, equals 57.5 products per week. The bottleneck of the flow line is the lithography machine. The simulations also show, that four important control topics need improvement. Firstly, the priority of personnel should be directed at assisting the bottleneck machine. Secondly, the personnel off time should be scheduled when the personnel is least needed. thirdly, the setup time of the machine that requires setup should be minimized by implementing an improved product release strategy in the generator and by improving the machine input selection procedure of the buffers. Finally, unscheduled down events should be handled by improved control.

Before the control of the flow line is improved in the model, the control problem of the Intel case has been analyzed to determine if the required throughput can be met. Initially the stochastics of the flow line is neglected in this analysis. Firstly, the case is analyzed manually. Capacity analysis shows all flow line elements have sufficient capacity. However, most personnel availability issues were not included in that analysis and therefore, in a realizable schedule the machines will have idle time that is not accounted for in the analysis. To include these issues, an educated schedule of the production for one shift without machine break down is constructed manually. The schedule fits in one shift, so the control problem without break down is feasible. Next, deterministic machine break down is added to the schedule. It will not be easy to manually construct a schedule that fits in the given time period. Also, to include the break down behavior correctly in the model, multiple shifts have to be scheduled. Therefore, an integer linear programming model of the case is designed. Because the model is too complex to be solved with the available optimization tools, even after simplification, the model is relaxed. That is, the integer constraints are neglected. The model can now be solved. The production for one week is scheduled in 0.97 weeks. The lithography machine is again the bottleneck of the flow line. All machines except the bottleneck machine have some surplus capacity. It is assumed the surplus capacity is sufficient to allow an integer solution. Therefore, the control problem with deterministic (planned) machine break down is assumed to be feasible. Next, the influence of stochastics on the control problem is analyzed. The stochastics is present in the break down behavior of the machines. An educated guess of the required extra capacity of the machines during a break down is made. This extra capacity is modelled and the model is solved again. The production for one week is still scheduled in 0.97 weeks and the lithography machine is the bottleneck of the flow line. Again the resulting schedule is not feasible due to the relaxation of the model. It will now be harder to make the schedule feasible, but all machines except the bottleneck machine still have some surplus capacity and the schedule can be prolonged by 0.03 weeks. Therefore, the control problem of the Intel case is assumed to be feasible.

Finally, the control of the flow line is improved. When machine break down is neglected, the control problem becomes deterministic and a repetitive schedule can be implemented. The required throughput can then be reached, as described in the previous paragraph. When machine break down is not neglected, a repetitive schedule

can not be used to control the flow line, due to stochastics. Instead, heuristic control is implemented in small steps to acquire elegant control. The control remains as simple as possible, so it can be easily understood and adjusted. Firstly, the product generator releases commercial products in series of equal type to minimize the setup time of the machine that requires setup. The generator releases the test products in between. Furthermore, the policy of the buffers is improved. They now present the machines with series of low and high production step products to acquire cyclic behavior and again to minimize the setup time. Next, the operating and off time priorities of personnel are improved. Personnel priority is directed mainly at the bottleneck machine of the flow line, the lithography machine, and off time is taken when personnel is least needed. Firstly, only the lithography machine is included in the χ model to determined whether the bottleneck machine has enough capacity to meet the required throughput with the designed control. From simulations results that the capacity is sufficient. Secondly, all machines are included in the model. To make the control easier to analyze, machine break down is neglected. Now, the lithography machine will loose some throughput, because the serving operator also assists at the implantation workstation. The maximal throughput equals 80.4 products per week, while the buffer capacity constraints are neglected. The lithography and the implantation workstation have equal throughput and both are the bottleneck of the flow line. Finally, the control should be refined and control for machine break down should be implemented. Due to lack of time in the project, this has not been done yet. Next to push control, also a sort of pull and conwip control have been implemented. Both pull and conwip control performed worse than push control for the flow line with current improved control.

Chapter 7

Recommendations

Five recommendations result from this research project. Firstly, in Chapter 4 an integer linear programming model of the Intel case has been designed to determine the feasibility of the control problem of the case. However, the model could not be solved and therefore the relaxed model was solved. The integer model could be adjusted, so the number of variables and constraints decreases and the model can be solved with the available optimization tools. Another option is to use other optimization tools. By solving the integer linear programming model instead of the relaxed model, the feasibility of the control problem of the Intel case is determined with more accuracy. It will also give a more accurate lower bound on the length of the schedule.

Secondly, due to lack of time in the project, the control has not been improved enough to reach the required throughput. The improvement should therefore be resumed. Initially, machine break down remains neglected. The designed control of Section 5.1 that has not been implemented yet, can be implemented and validated. Next, the maximal throughput is determined. If the required throughput can not be met, the control should be improved by analyzing the flow line behavior until the required throughput can be met. Then, the designed control after an unscheduled down event occurs can be implemented and machine break down is not neglected in the model. Again, the control should be validated and the maximal throughput should be determined. If the required throughput can not be met, the control should be improved until the throughput is met.

Thirdly, during implementation of the improved control, a problem occurred. The personnel can not be send in advance to the workstation it will operate in next. Therefore, all machines, including the lithography machine will have extra idle time, in which the personnel is transported. If the required throughput can not be met with the implementation of improved control, the model can be adjusted. The personnel can be sent to the personnel transporter and then back to the personnel dispatcher, from where it is dispatched to the destination process.

Fourthly, if the case is a model of an existing factory, then the control of the χ model

could be implemented in the real flow line. The flow line can then be monitored and the change in throughput of the line, hopefully an increase of throughput, can then be determined. Also, the control can be analyzed in practice. If the control is not satisfactory, it can be adjusted firstly in the model. The model can then be used to predict the effect of change in control on the throughput, flow time and work in process of the flow line. If the effect is desirable it can be implemented in the real flow line.

Finally, the χ model of the flow line contains all difficulties that need to be dealt with in practice and could therefore be used as test case for other research projects. For example, it could be used for research on control of re-entrant flow lines or for testing of scheduling algorithms. Although the model is fairly complex, it can also be used for educational projects.

Bibliography

[1] A.T. Hofkamp and J.E. Rooda. *χ reference manual*. Academic Service, 2002.

[2] K. Kempf. *Intel Five-Machine Six Step Mini-Fab Description*. Intel/ASU Report, 1994. `http://www.eas.asu.edu/~aar/research/intel/papers/fabspec.html`.

[3] J.P.C. Kleijnen. *Verification and validation of simulation models*. Katholieke Universiteit Brabant, 1992.

[4] J.E. Rooda and J. Vervoort. *Analysis of Manufacturing Systems*. Eindhoven University of Technology, 2003.

[5] M. Sevaux and P. Thomin. *Heuristics and metaheuristics for a parallel machine scheduling problem: a computational evaluation*. University of Valenciennes, 2001.

[6] J.P. Sousa and L.A. Wolsey. *A time-indexed formulation of non-preemptive single machine scheduling problems*. Mathematical Programming 54, 1992.

[7] J. Vervoort and J.E. Rooda. *Learning χ 0.8*. Academic Service, 2003.

Appendix A

χ model

In this appendix the χ model with FIFO (First In First Out) control is presented in four sections. In the first section the general elements of the model are described. In the next two sections the framework of the model is presented. It consists of two levels. First the upper level is described followed by the lower level. The χ model is shown in the final section.

General elements

The general elements of the χ model are put at the top and at the bottom of the model. The top contains the elements that are available throughout the model. These elements are divided into three sections. The first section contains the libraries that need to be imported. After importing a library, the functions that are defined in the library can be used in the model. In the second section the model parameters are defined as constants. They can now be used throughout the model and are easily adjusted, because they are gathered at the top of the model. The final section contains the aliased variable types. The aliases are used in the model to increase the readability of the model. One of the aliases is the lot. This term is used to represent a product as it flows through the line. The bottom of the model contains the initiation command, which initiates the entire flow line. This concludes the general elements of the model.

Upper level

After the general elements have been described, the framework of the χ model is presented. This section presents the upper level of the framework and the next section the lower level. The upper level contains the entire flow line and is represented by cluster Intel. The cluster consists of five types of clusters and two types of processes. The first cluster is the generator. It signals a lot transporter when a lot is released into the flow line and passes the lot when the lot transporter has arrived. The second cluster

is the lot transporter. It receives requests for lot transportation from the generator and the workstation buffers. Furthermore, it receives lots from the generator and the buffers, when the transporter has arrived. Finally, it sends the lots to the buffers and the exit, depending on the production step of the lot. The multi-machine workstation buffer is the third cluster. Firstly, it receives lots from the lot transporter and receives requests for input from the workstations machines. Next, when the buffer has both a request and correct machine input, it sends a request for an operator to the personnel cluster. When the operator has been received, he is sent together with the input to the requesting machine. After a machine has been unloaded, it sends its output together with the operator that performed the unloading to the buffer. Next, the buffer returns the operator to the personnel cluster and requests for lot transportation. Finally, when the transporter has arrived, it receives a lot from the buffer. The fourth cluster is the single-machine workstation buffer. It operates similar to the multi-machine workstation buffer, but it serves only one machine. The final cluster is the personnel cluster. It receives requests for personnel from the buffers and the machines. Furthermore, personnel is received from and dispatched to them by the personnel cluster.

The first process of the upper level is the machine. It requests the workstation buffer for input and receives the input together with an operator. If the machine is the lithography machine, then the setup is applied. Next, the loading time of the machine is applied. After the loading, the operator is returned to the personnel cluster and the machine input is processed. If the machine is an ion implantation machine, then the unscheduled down timer is started. If a break down occurs during processing, then a technician is requested from the personnel cluster. After the technician has been received, unscheduled maintenance is applied and he is returned to the personnel cluster. If the break down does not occur or the machine is not an ion implantation machine, then an operator is requested from the personnel cluster, after the machine input has been processed. When the operator has been received, the machine output is unloaded and updated. Next, the output is sent together with the operator to the workstation buffer. Finally, if the machine receives a technician from the personnel cluster instead of input from a buffer, then scheduled maintenance is performed on the machine and the technician is returned to the cluster. The second and final process is the exit. Its function is to receive the finished lots from the lot transporter.

After the description of the clusters and the processes, the flow of lots through the upper level of the framework is presented. The lots flow from the generator via the lot transporter to the first workstation buffer. This buffer sends them to one of the machines in the workstation. The machine returns them to the buffer after they are processed. Next, the lot transporter transports the lots to the next processing area. After the processing of the lots has been finished, the lot transporter transports them to the exit. This concludes the upper level of the framework of the model.

Lower level

The lower level of the framework of the model is represented by the five clusters inside the upper level, which has been described in the previous section. The first cluster is the generator. It contains two types of processes. The first type is the signal generator, which has the task of signalling a collector process with predetermined time intervals. The second type is the collector, which receives signals from the signal generators. After the collector has received a signal, it constructs a lot and puts it in a buffer. Furthermore, it requests for lot transportation and when the transporter arrives, the collector sends the lot to the transporter. The generator consists of three signal generators and one collector.

The second cluster is the lot transporter. It consists of two processes. The first process is the lot transporter dispatcher. It receives requests for transportation from the generator and the workstation buffers and dispatches the lot transporter. The second process is the lot transporter. It requests a pickup position from the dispatcher and moves to the position. When it has arrived, it receives a lot and determines the drop position. Finally, it moves to the drop position and drops the lot.

The third cluster is the multi-machine workstation buffer. It contains two processes. The first process is the incoming multi-machine workstation buffer. The incoming buffer starts with determining correct machine input from the lots that are present in the buffer together with the placed requests for machine input. To this end, the functions *dispd* and *dispi* both return the first element of the list of all correct input of respectively the diffusion and the ion implantation machines. Next, the buffer tries to receive lots and also requests for input from machines. If correct machine input is available, then the buffer requests an operator from the personnel cluster. Finally, after the operator has arrived, the input and the operator are sent to the requesting machine. The second process is the outgoing multi-machine workstation buffer. Firstly, it receives machine output together with an operator. Next, it returns the operator and requests for lot transportation. Finally, when the lot transporter has arrived, it receives a lot from the buffer.

The fourth cluster is the single-machine workstation buffer. It contains the two processes incoming single-machine workstation buffer and outgoing single-machine workstation buffer. They operate similar to the multi-machine workstation buffers. Only two differences exist. Firstly, the former buffers serve only one machine and secondly the incoming single-machine workstation buffer does not require a function to determine correct machine input, because every single lot is correct input.

The final cluster is the personnel cluster. It consists of three processes. The first process is the personnel dispatcher. It starts with determining via functions the personnel options for off time and operating. Next, it determines the earliest scheduled down window and counter events of all the machine. Furthermore, it determines the earliest off time counter event of all personnel. After this has been determined, the dispatcher tries to receive requests for personnel and also the personnel itself from the

workstations. When it has received personnel, the dispatcher updates the machine and the scheduled down status of the flow line, because the personnel has changed the state of the line with its action. Next, the dispatcher tries to receives personnel from an off time process and tries to dispatch personnel to the process. It also tries to dispatch personnel to the workstations via a personnel transporter. When the personnel is sent to the transporter, the necessary adjustments are made to represent the state of the flow line correctly. Finally, the scheduled maintenance window and counter events and also the off time counter events are handled. The second process in the personnel cluster is the personnel transporter. It receives personnel from the personnel dispatcher and applies its transportation time. Furthermore, it sends the personnel to the destination process, that is determined by the dispatcher. The final process is the off time process. It receives personnel from the personnel dispatcher and applies the off time. After the off time, the personnel is returned to the dispatcher. This concludes the lower level of the framework of the model and now the entire model has been described.

χ model with FIFO control

In this section the χ model with FIFO control is shown. Information on the χ formalism can be found in [7] and [1]. To increase the readability of the model, consistent nomenclature is used. The channel names are formed by the letters of the alphabet starting from a. The variable names also show consistency. Firstly, the i is used as a counter and the j is used to construct bundles. Furthermore, a list of variables ends with s and the variables that represent a time quantity start with t. Finally, a product is represented by x, y or z. The readability of the model is also increased by visualizing the clusters. Figure A.1 shows the legend of the visualizations of the clusters and Figure A.2 visualizes the upper level of the χ model, which is represented by cluster Intel. This cluster contains all the clusters and the processes that have been discussed in the section of the upper level. Next to the clusters and the processes, also the communication channels are shown. Figures A.3 through A.7 show the same elements for the lower level of the framework. The figures show respectively the generator, the lot transporter, the multi-machine workstation buffer, the single-machine workstation buffer and the personnel cluster. They are followed by the χ model of the model, which concludes this appendix.

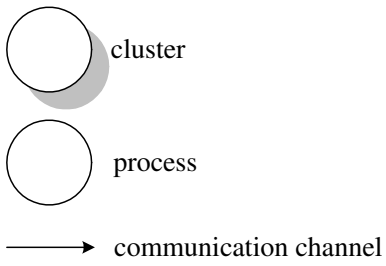
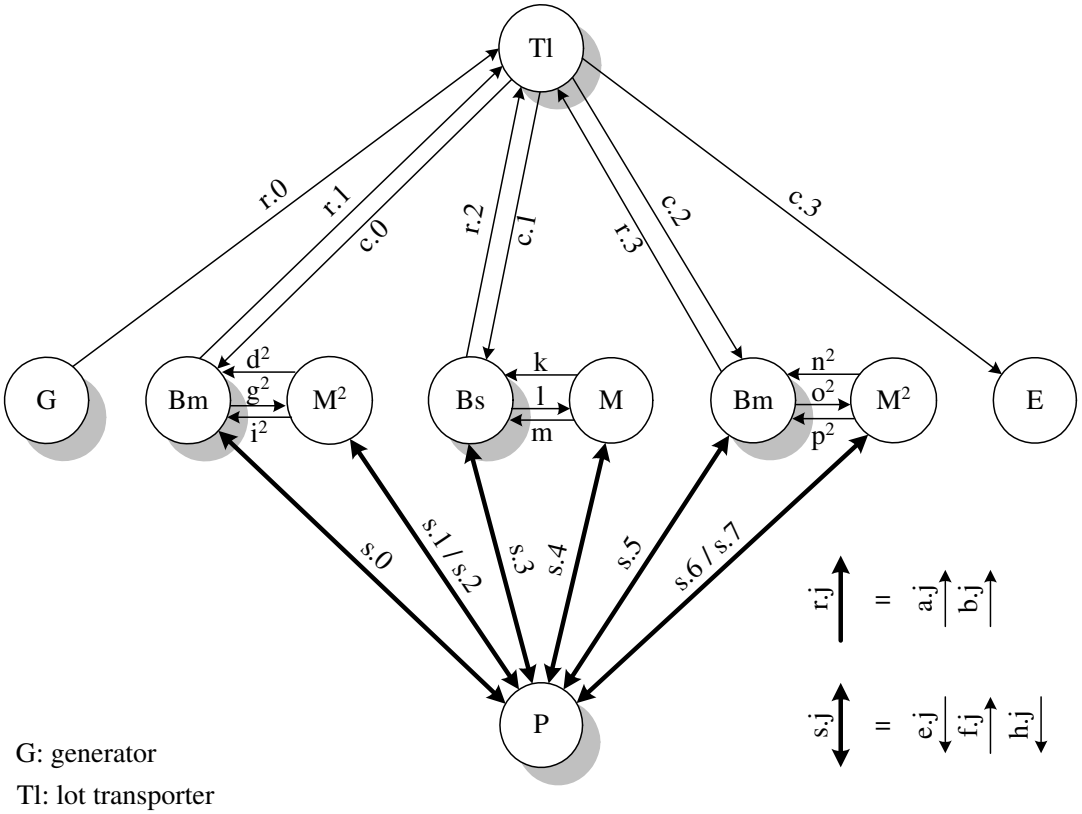
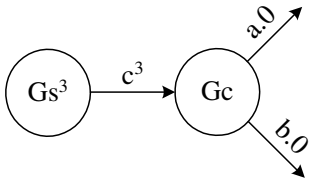


Figure A.1: *Cluster visualization legend*



G: generator
Tl: lot transporter
Bm: multi-machine workstation buffer
Bs: single-machine workstation buffer
M: machine
P: personnel
E: exit

Figure A.2: *Cluster Intel*



Gs: signal generator
Gc: collector

Figure A.3: *Cluster generator*

Tpld: lot transporter dispatcher
Tpl: lot transporter

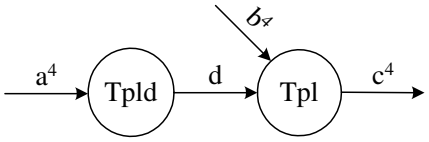
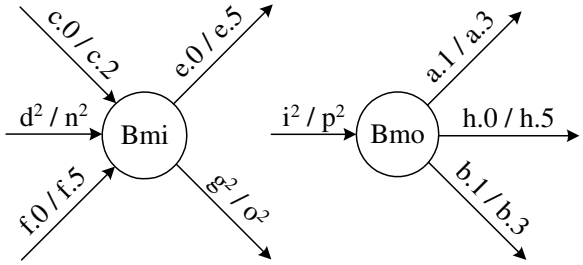
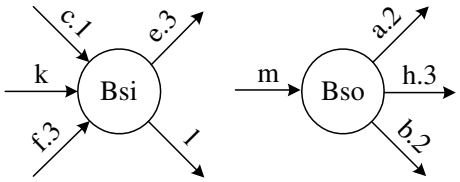


Figure A.4: *Cluster lot transporter*



Bmi: incoming multi-machine workstation buffer
Bmo: outgoing multi-machine workstation buffer

Figure A.5: *Cluster multi-machine workstation buffer*



Bsi: incoming single-machine workstation buffer
Bso: outgoing single-machine workstation buffer

Figure A.6: *Cluster single-machine workstation buffer*

Pd: personnel dispatcher
Pt: personnel transporter
Po: off-time

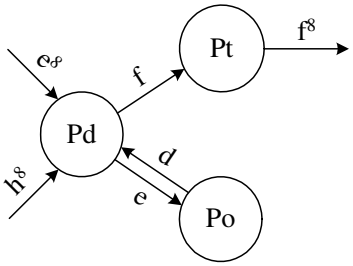


Figure A.7: *Cluster personnel*

```
from std import *
from random import *

const ih:   nat^3   = <|2,2,2|>                       // ini history
    , imst: nat^5   = <|0,0,0,0,0|>                   // ini machine status
    , iot:  nat^3^2 = <|<|0,0,0|>,<|0,0,0|>|>         // ini off-times
    , ip:   nat^2*  = [<|0,0|>,<|0,1|>,<|0,2|>]       // ini state personnel (pos,type)
    , isdc: nat^5   = <|0,0,0,0,0|>                   // ini sched down counter
    , isdw: nat^5   = <|2,2,2,2,2|>                   // ini scheduled down window
    , isp:  nat^2   = <|2,0|>                         // ini setup (sp,tp)
    , itl:  nat     = 0                               // ini pos. transporter lots
    , itsd: real^5  = <|0.0,0.0,0.0,0.0,0.0|>         // ini sched down couter raise time
    , mca:  nat^8   = <|0,0,0,1,1,2,2,2|>             // channel to op area mapping
    , mfp:  real    = 0.455                           // mach fraction processing
    , mcm:  nat^8   = <|0,0,1,2,2,3,3,4|>             // channel to machine mapping
    , mmc:  nat^5   = <|1,2,4,6,7|>                   // machine to channel mapping
    , mr:   nat^7   = <|1,3,2,3,1,2,4|>               // lot routing mapping
    , pim:  bool^8  = <|false,true,true,false,true,false,true,true|> // process is machine
    , tasd: real^5  = <|1440.0,1440.0,720.0,720.0,720.0|> // average time betw sched down
    , tbo:  nat^2   = <|360,720|>                     // time between break,meeting
    , tbud: real^2  = <|1440.0,4560.0|>               // time between unsch down (lb,ub)
    , to:   nat^3^2 = <|<|45,60,60|>,<|30,60,60|>|>   // off times (break,meeting)
    , te:   nat^6   = <|225,30,55,50,255,10|>         // process times
    , ti:   real^3  = <|10080/3,10080/51,10080/30|>   // inter departure times
    , itno: nat^3^2 = <|<|180,150,210|>,<|360,330,390|>|> // next off times
    , tol:  nat^3   = <|20,10,15|>                    // op load times
    , tou:  nat^3   = <|40,10,15|>                    // op unload times
    , tsd:  nat^3   = <|75,30,120|>                   // scheduled down times
    , tsp:  nat^4   = <|0,5,10,12|>                   // setup times (none,tp,sp,both)
    , ttlu: nat     = 2                               // transp loading + unloading time
    , ttr:  nat^2   = <|4,1|>                         // transport times (transp,pers)
    , tud:  real^2  = <|340.0,480.0|>                 // unscheduled down time (lb,ub)

type lot  = id.nat#tp.nat#sp.nat#hs.nat^3#st.real
            // identification#type (0,1,2)#step (0..6)#history#starttime
   , n1ls = nat#lot*
   , n2ls = nat#nat#lot*
   , n3ls = nat#nat#nat#lot*
   , n4ls = nat#nat#nat#nat#lot*

proc Gs(a: !void, n: nat) = |[ *[ true -> a!; delta ti.n ] ]|

proc Gc(a: (?void)^3, b: !void, c: !lot*) =
|[ xs: lot*, i: nat
 | xs:= []; i:= 0
 ; *[ j: nat <- 0..3: true; a.j? -> xs:= xs ++ [<i,j,0,ih,time>]; i:= i + 1; b!
    | len(xs) > 0; c![hd(xs)]    -> xs:= tl(xs)
    ]
]|

clus G(a: !void, b: !lot*) = |[ c: (-void)^3 | j: nat <- 0..3: Gs(c.j,j) || Gc(c,a,b) ]|

proc Tpld(a: (?void)^4, b: !nat) =
|[ xs: nat*
 | xs:= []
 ; *[ j: nat <- 0..4: true; a.j? -> xs:= xs ++ [j]
    | len(xs) > 0; b!hd(xs)      -> xs:= tl(xs)
    ]
]|

func tt(ft: nat^2, n,k: nat) -> int = |[ ret n * abs(+ft.0 - ft.1) + k ]|
```

```
proc Tpl(a: ?nat, b: (?lot*)^4, c: (!lot*)^4) =
|[ ft: nat^2, xs: lot*
 | ft.0:= itl
 ; *[ true
      -> a?ft.1; delta tt(ft,ttr.0,0); b.(ft.1)?xs; ft:= <| ft.1, mr.(hd(xs).sp) |>
       ; delta tt(ft,ttr.0,ttlu); c.(ft.1 - 1)!xs; ft.0:= ft.1
    ]
]|

clus Tl(a: (?void)^4, b: (?lot*)^4, c: (!lot*)^4) =
|[ d: -nat | Tpld(a,d) || Tpl(d,b,c) ]|

func sel(xs: n1ls*) -> n1ls =
|[ [ len(xs) = 0 -> ret <0,[]> | len(xs) > 0 -> ret hd(xs) ] ]|

func feasbat(r: nat, x,y,z: lot) -> bool =
|[ testlots: lot*, a,b,c: bool, n: nat
 | testlots:= [ p | p: lot <- [x,y,z], p.tp = 0 ]; n:= len(testlots)
 ; a:= x.tp = y.tp; b:= x.tp = z.tp; c:= y.tp = z.tp
 ; [ n >= 2              -> ret false
   | n < 2 and x.sp < 3  -> ret true
   | n = 0 and x.sp >= 3 -> ret a and b
   | n = 1 and x.sp >= 3 -> ret (a or b or c) and hd(testlots).hs.0 /= r
   ]
]|

func dispd(rs: nat*, xs: lot*) -> n1ls =
|[ ret sel([ <r,[x,y,z]>
          | r: nat <- rs
          , x: lot <- xs, y: lot <- xs, x.id < y.id, x.sp = y.sp
          , z: lot <- xs, y.id < z.id, y.sp = z.sp
          , feasbat(r,x,y,z)
          ])
]|

func dispi(rs: nat*, xs: lot*) -> n1ls =
|[ ret sel([ <r,[x]> | r: nat <- rs, x: lot <- xs, x.tp /= 0 or x.hs.2 /= r ]) ]|

proc Bmi(a: ?lot*, b: (?void)^2, c: !n2ls, d: ?n2ls
        , e: (!n1ls)^2, disp: (nat*,lot*) -> n1ls) =
|[ xs,ys,zs: lot*, p,r: nat, rs: nat*, rzss: (n1ls)*
 | xs:= []; rs:= []; rzss:= []
 ; *[ true
      -> <r,zs>:= disp(rs,xs)
       ; [ true; a?ys                -> xs:= xs ++ ys
         | j: nat <- 0..2: true; b.j? -> rs:= rs ++ [j]
         | len(zs) > 0; c!<1,r,zs>
           -> rzss:= rzss ++ [<r,zs>]; rs:= rs -- [r]; xs:= xs -- zs
         | true; d?<p,r,zs>           -> rzss:= rzss -- [<r,zs>]; e.r!<p,zs>
         ]
    ]
]|

proc Bmo(a: (?n1ls)^2, b: !nat^2, c: !void, d: !lot*) =
|[ xs,ys: lot*, i,p: nat
 | xs:= []
 ; *[ j: nat <- 0..2: true; a.j?<p,ys>
      -> xs:= xs ++ ys; b!<|p,j|>; i:= len(ys); *[ i > 0 -> c!; i:= i - 1 ]
    | len(xs) > 0; d![hd(xs)]
      -> xs:= tl(xs)
    ]
]|
```

```
clus Bm(a: ?lot*, b: (?void)^2, c: !n2ls, d: ?n2ls, e: (!n1ls)^2, f: (?n1ls)^2, g: !nat^2
       , h: !void, i: !lot*, disp: (nat*,lot*) -> n1ls) =
|[ Bmi(a,b,c,d,e,disp) || Bmo(f,g,h,i) ]|

proc Bsi(a: ?lot*, b: ?void, c: !n2ls, d: ?n2ls, e: !n1ls) =
|[ xs,ys: lot*, rq: bool, p,q: nat
 | xs:= []; rq:= false
 ; *[ true; a?ys                        -> xs:= xs ++ ys
    | true; b?                          -> rq:= true
    | rq and xs /= []; c!<1,0,[hd(xs)]> -> rq:= false
    | true; d?<p,q,ys>                  -> xs:= xs -- ys; e!<p,ys>
    ]
]|

proc Bso(a: ?n1ls, b: !nat^2, c: !void, d: !lot*) =
|[ xs,ys: lot*, p: nat
 | xs:= []
 ; *[ true; a?<p,ys>          -> xs:= xs ++ ys; b!<|p,0|>; c!
    | len(xs) > 0; d![hd(xs)] -> xs:= tl(xs)
    ]
]|

clus Bs(a: ?lot*, b: ?void, c: !n2ls, d: ?n2ls, e: !n1ls, f: ?n1ls, g: !nat^2, h: !void
       , i: !lot*) =
|[ Bsi(a,b,c,d,e) || Bso(f,g,h,i) ]|

func su(psp,ptp,sp,tp: nat) -> nat =
|[ [ sp = psp and tp = ptp   -> ret tsp.0
   | sp = psp and tp /= ptp  -> ret tsp.1
   | sp /= psp and tp = ptp  -> ret tsp.2
   | sp /= psp and tp /= ptp -> ret tsp.3
   ]
]|

func updm(x: lot, wi,mi: nat) -> lot = |[ x.sp:= x.sp + 1; x.hs.wi:= mi; ret x ]|

proc M(a: !void, b: ?n1ls, c: !nat^2, d: !n2ls
      , e: ?n2ls, f: !n1ls, wi,mi: nat) =
|[ dtbud,dtud: -> real, tnud: real, p,q,psp,ptp,sp,tp: nat, ud: bool, qs,xs: lot*
 | dtbud:= uniform(mfp*tbud.0,mfp*tbud.1); dtud:= uniform(tud.0,tud.1)
 ; tnud:= sample dtbud; <|psp,ptp|>:= isp; ud:= false; a!
 ; *[ true; b?<p,xs>
      -> [ wi = 1
           -> sp:= hd(xs).sp; tp:= hd(xs).tp; delta su(psp,ptp,sp,tp); psp:= sp; ptp:= tp
         | wi /= 1 -> skip
         ]
       ; delta tol.wi; c!<|p,0|>
       ; [ wi = 2
           -> tnud:= tnud + time
            ; [ true; delta te.(hd(xs).sp)
                -> tnud:= tnud - time
              | true; delta tnud - time
                -> ud:= true; d!<0,0,[]>; e?<p,q,qs>; delta sample dtud
                 ; c!<|p,0|>; tnud:= sample dtbud
              ]
         | wi /= 2 -> delta te.(hd(xs).sp)
         ]
       ; [ ud     -> ud:= false
         | not ud -> d!<1,0,[]>; e?<p,q,qs>; delta tou.wi
                   ; f!<p,[ updm(x,wi,mi) | x: lot <- xs ]>
         ]
```

```
       ; a!
    | true; e?<p,q,qs>
      -> delta tsd.wi; c!<|p,0|>
    ]
]|

func feaswp(w,j,p: nat) -> bool =
|[ ret w = p and (w = 0 or w = 1 and (j < 3 or j > 4)) or w = 2 and p = 1 and j > 2 ]|

func dispp(ot: nat^3^2, rs: (n3ls)*, wps: nat^2*, mst,sdw: nat^5)
          -> (nat#nat^2)*#((n3ls)#nat^2)*#nat^5 =
|[ ops: (nat#nat^2)*, dps: ((n3ls)#nat^2)*, dp: (n3ls)#nat^2, k,m: nat
 | ops:= [ <n,w> | w: nat^2 <- wps, n: nat <- [0,1], ot.n.(w.1) > 0 ]
 ; dps:= [ <<j,p,q,xs>,w> | w: nat^2 <- wps, ot.0.(w.1) = 0 and ot.1.(w.1) = 0
                          , <j,p,q,xs>: n3ls <- rs, feaswp(w.1,j,p)
                          , sdw.(mcm.j+q) > 0 or pim.j ] ++
         [ <<mmc.n,0,0,[]>,w> | ot.0.0 = 0 and ot.1.0 = 0, w: nat^2 <- wps, w.1 = 0
                              , n: nat <- [0,1,2,3,4], mst.n = 0, sdw.n /= 1 ]
 ; [ len(dps) = 0
     -> skip
   | len(dps) > 0
     -> dp:= hd(dps); k:= dp.0.1; m:= mcm.(dp.0.0)
      ; [ k = 0 and mst.m = 0 -> sdw.m:= 0 | k /= 0 or mst.m /= 0 -> skip ]
   ]
   ; ret <ops,dps,sdw>
]|

func pns(x,y: real#nat) -> real#nat = |[ [ x.0 <= y.0 -> ret x | x.0 > y.0 -> ret y ] ]|

func nsdwe(sdw: nat^5, twc,two: real^5) -> (real#nat)* =
|[ nsdwns: (real#nat)*
 | nsdwns:= [ <twc.n,n> | n: nat <- [0,1,2,3,4], sdw.n = 2 ] ++
            [ <two.n,n> | n: nat <- [0,1,2,3,4], sdw.n = 1, twc.n - tasd.n <= two.n ] ++
            [ <twc.n-tasd.n,n> | n: nat <- [0,1,2,3,4], sdw.n = 1, twc.n - tasd.n > two.n ]
 ; [ len(nsdwns) = 0 -> ret [] | len(nsdwns) > 0 -> ret [fold(tl(nsdwns),pns,hd(nsdwns))] ]
]|

func nsde(tnsd: real^5) -> real#nat#real^5 =
|[ sdns: (real#nat)*, tsdc: real, u: nat
 | sdns:= [ <tnsd.n,n> | n: nat <- [0,1,2,3,4] ]
 ; <tsdc,u>:= fold(tl(sdns),pns,hd(sdns)); tnsd.u:= tnsd.u + tasd.u; ret <tsdc,u,tnsd>
]|

func pno(x,y: nat^3) -> nat^3 = |[ [ x.0 <= y.0 -> ret x | x.0 > y.0 -> ret y ] ]|

func no(tno: nat^3^2) -> nat#nat#nat#nat^3^2 =
|[ mns: nat^3*, tot,u,v: nat
 | mns:= [ <|tno.m.n,m,n|> | m: nat <- [0,1], n: nat <- [0,1,2] ]
 ; <|tot,u,v|>:= fold(tl(mns),pno,hd(mns)); tno.u.v:= tno.u.v + tbo.u; ret <tot,u,v,tno>
]|

func updp(j,p,q: nat, mst,sdc,sdw: nat^5, twc,two: real^5, t: real)
         -> nat^5#nat^5#nat^5#real^5#real^5 =
|[ n: nat, trem: real
 | n:= mcm.j + q
 ; [ p = 0 and mst.n = 1
     -> sdw.n:= 1; sdc.n:= sdc.n - 1; two.n:= t + (tasd.n)/2; mst.n:= 0
      ; trem:= t + tasd.n - rmod(t,tasd.n)
      ; [ sdc.n = 0 -> twc.n:= trem + tasd.n
        | sdc.n = 1 -> twc.n:= max(two.n,trem)
        | sdc.n > 1 -> twc.n:= two.n
        ]
```

```
   | p > 0 or mst.n /= 1
     -> mst.n:= mst.n mod 2
   ]
 ; ret <mst,sdc,sdw,twc,two>
]|

proc Pd(a: (?n2ls)^8, b: (?nat^2)^8, c: ?nat^2*, d: !nat#nat^2, e: !n4ls) =
|[ dps: ((n3ls)#nat^2)*, rs: (n3ls)*, wps,ws: nat^2*, w: nat^2, mst,nsdw,sdc,sdw: nat^5
 , ops: (nat#nat^2)*, op: nat#nat^2, r: n3ls, m,n,p,q,u,tot: nat, xs: lot*
 , nsdwes: (real#nat)*, ntnsd,tnsd,twc,two: real^5, ot,ntno,tno: nat^3^2, tsdc: real
 | rs:= []; wps:= ip; mst:= imst; twc:= tasd; sdc:= isdc; sdw:= isdw; ot:= iot
 ; tno:= itno; tnsd:= itsd
 ; *[ true
      -> <ops,dps,nsdw>:= dispp(ot,rs,wps,mst,sdw); nsdwes:= nsdwe(sdw,twc,two)
       ; <tsdc,u,ntnsd>:= nsde(tnsd); <tot,m,n,ntno>:= no(tno)
       ; [ j: nat <- 0..8: true; a.j?<p,q,xs>
           -> rs:= rs ++ [<j,p,q,xs>]
         | j: nat <- 0..8: true; b.j?<|p,q|>
           -> wps:= wps ++ [<|j,p|>]
            ; <mst,sdc,sdw,twc,two>:= updp(j,p,q,mst,sdc,sdw,twc,two,time)
         | true; c?ws
           -> wps:= wps ++ ws
         | len(ops) > 0; d!hd(ops)
           -> op:= hd(ops); wps:= wps -- [op.1]; m:= op.0; n:= op.1.1; ot.m.n:= ot.m.n - 1
         | len(dps) > 0; e!<hd(dps).1.0,hd(dps).0.0,hd(dps).1.1,hd(dps).0.2,hd(dps).0.3>
           -> <r,w>:= hd(dps); rs:= rs -- [r]; wps:= wps -- [w]; n:= mcm.(r.0) + r.2
            ; mst.n:= mst.n + 1; sdw:= nsdw
         | len(nsdwes) > 0; delta hd(nsdwes).0 - time
           -> n:= hd(nsdwes).1; sdw.n:= (sdw.n + 1) mod 3
         | true; delta tsdc - time
           -> sdc.u:= sdc.u + 1; tnsd:= ntnsd
         | true; delta tot - time
           -> ot.m.n:= ot.m.n + 1; tno:= ntno
         ]
    ]
]|

func ptpp(x,y: real#nat#nat#nat#lot*) -> bool = |[ ret x.0 <= y.0 ]|

proc Pt(a: ?n4ls, b: (!n2ls)^8) =
|[ xs: (real#nat#nat#nat#lot*)*, p,q,u,v: nat, ys:lot*, x: real#nat#nat#nat#lot*
 | xs:= []
 ; *[ true; a?<u,v,p,q,ys>
      -> xs:= insert(xs,<time+tt(<|mca.u,mca.v|>,ttr.1,0),v,p,q,ys>,ptpp)
    | len(xs) > 0; delta hd(xs).0 - time
      -> x:= hd(xs); b.(x.1)!<x.2,x.3,x.4>; xs:= tl(xs)
    ]
]|

func ppo(x,y: real#nat^2) -> bool = |[ ret x.0 <= y.0 ]|

proc Po(a: ?nat#nat^2, b: !nat^2*) =
|[ os: (real#nat^2)*, op: nat^2, m: nat
 | os:= []
 ; *[ true; a?<m,op>                    -> os:= insert(os,<time+to.m.(op.1),op>,ppo)
    | len(os) > 0; delta hd(os).0 - time -> b![hd(os).1]; os:= tl(os)
    ]
]|

clus P(a: (?n2ls)^8, b: (?nat^2)^8, c: (!n2ls)^8) =
|[ d: -nat^2*, e: -nat#nat^2, f: -n4ls
 | Pd(a,b,d,e,f) || Pt(f,c) || Po(e,d)
```

```
]|

proc E(a: ?lot*) = |[ xs: lot* | *[ true -> a?xs ] ]|

clus Intel()=
|[ a: (-void)^4, b,c: (-lot*)^4, d,n: (-void)^2, e,f: (-n2ls)^8, g,i,o,p: (-n1ls)^2
 , h: (-nat^2)^8, l,m: -n1ls, k: -void
 | G(a.0,b.0)
|| Tl(a,b,c)
|| Bm(c.0,d,e.0,f.0,g,i,h.0,a.1,b.1,dispd)
|| j: nat <- 0..2: M(d.j,g.j,h.(j+1),e.(j+1),f.(j+1),i.j,0,j)
|| Bs(c.1,k,e.3,f.3,l,m,h.3,a.2,b.2)
|| M(k,l,h.4,e.4,f.4,m,1,0)
|| Bm(c.2,n,e.5,f.5,o,p,h.5,a.3,b.3,dispi)
|| j: nat <- 0..2: M(n.j,o.j,h.(j+6),e.(j+6),f.(j+6),p.j,2,j)
|| P(e,h,f)
|| E(c.3)
]|

xper = |[Intel()]|
```

Appendix B

Adjusting the time between failure

To model the Intel case correctly, the parameter that adjusts the time between failure of the ion implantation machines needs to be determined. This is necessary to model the right number of unscheduled downs. The parameter needs to be determined every time the model is changed. This appendix presents the adjustment procedure.

The parameter that needs to be adjusted in the model is the *mfp* parameter. It represents the average fraction of time that the ion implantation machines are processing. Its lower boundary equals 0 and is reached when the machines are continuously idle. Its upper boundary equals 0.57 and is reached when the machines have no idle time. The upper boundary equals less than 1, because the machines have to be loaded and unloaded. The value of the parameter is determined by counting the number of unscheduled downs of both ion implantation machines during the last one hundred weeks of the simulation of two hundred weeks of processing. After the first one hundred weeks the transient behavior of the flow line can be neglected. The value of the parameter is varied until the number of downs equals the prescribed amount of 400 downs per one hundred weeks in steady state. Due to stochastics the number of downs varies per simulation run. The number is therefore counted in six runs and the six values are averaged to get the correct number of downs. For example, the correct value for the *mfp* parameter equals 0.455 for the flow line with required input and with FIFO (First In First Out) control.

Appendix C

χ validation examples

When the χ model with FIFO (First In First Out) control has been designed, it has to be validated before simulation results can be trusted. The four validation techniques [3] used in this project are 'transient calculation', 'steady state calculation', 'visualization using Gantt charts' and 'function analysis'. Of each used validation technique an example is presented in this appendix.

Transient calculation

In transient calculation a few products are released into the flow line and the simulation output is compared to manual calculation. The outcome should be identical. The first six products of the required input of the Intel case are released into the flow line. The type restrictions on the diffusion machine input are neglected, so all batches can be composed. The activity of the machines, the personnel and the transporter during the flow of the products through the line is output by simulation and is calculated manually. Of both results a Gantt chart is constructed. The legend of the charts is presented in Figure C.1 and the charts are visualized in Figure C.2. The Gantt charts are identical and therefore the model behaves correctly in this validation.

T	transporter		transporter busy
Op_1	operator 1		personnel busy
Op_0	operator 0		personnel break
Tech	technician		personnel meeting
Mi_1	ion implantation machine 1		machine busy
Mi_0	ion implantation machine 0		scheduled down
Ml	lithography machine		
Md_1	diffusion machine 1		
Md_0	diffusion machine 0		

Figure C.1: *Legend of the Gantt charts of the flow line*

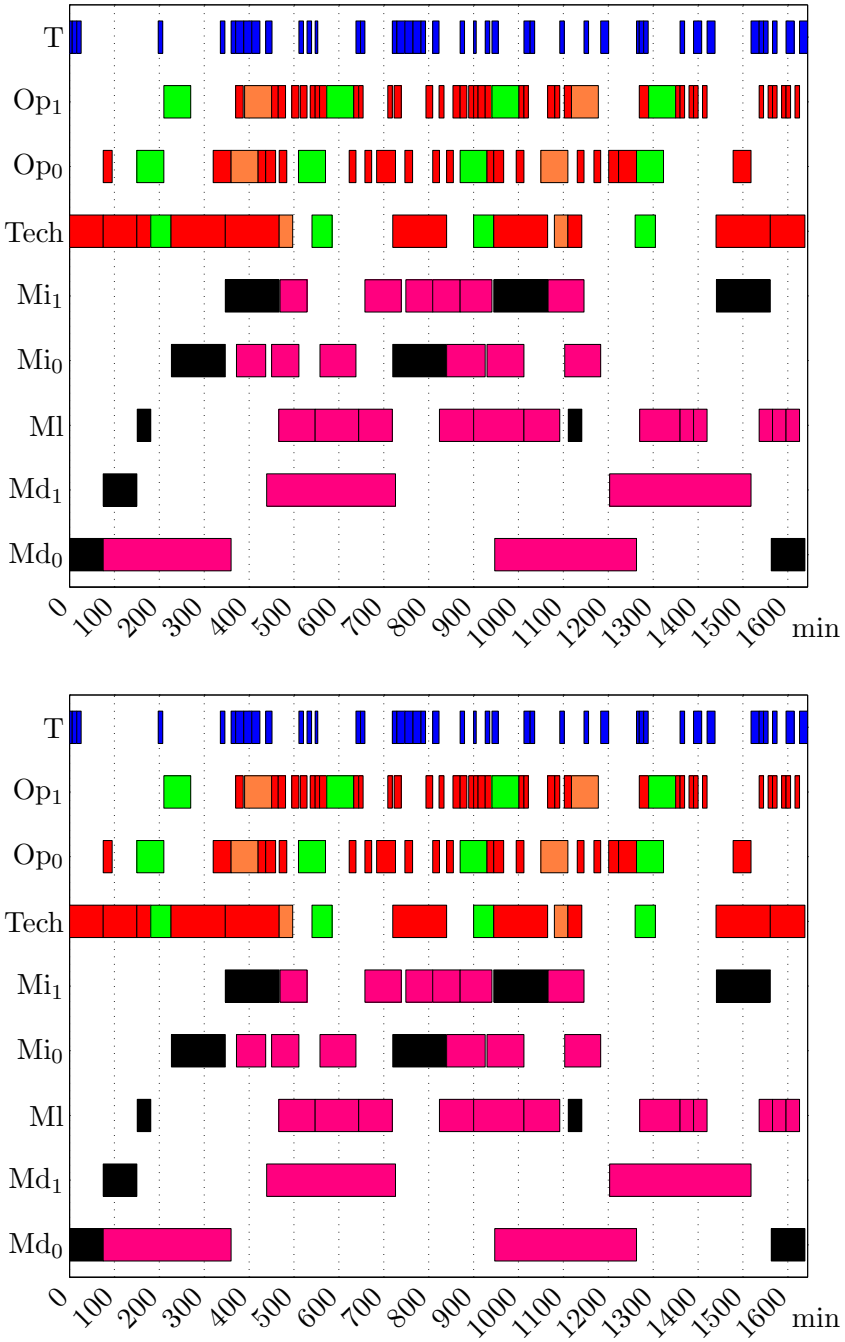


Figure C.2: *Gantt charts made by simulation (top) and by manual calculation (bottom) of the flow of six products through the line*

Steady state calculation

In steady state calculation the model output in steady state is compared to manual calculations. The outcome should be identical. In this example the compared output is

the number of scheduled maintenances of the lithography machine during one hundred weeks of production. the number of maintenances is not influenced by the transient behavior of the flow line, because the windows are independent of the state of the flow line. Therefore, the counting of the maintenances can start directly. The model is changed to output the total number of scheduled maintenances when the lithography machine receives maintenance. Next, simulations are run six times for one hundred weeks to account for stochastics. The average result of the simulations is 1399.8 maintenances. The manual calculation results in 1400 maintenances per one hundred weeks. In one simulation the maintenance of the final shift was too late and therefore a small difference in the outcome resulted. The difference is completely explicable and in this case allowed. Therefore, the model behaves correctly in this validation.

Visualization using Gantt charts

In visualization using Gantt charts the behavior of the flow line is visualized in Gantt charts and the occurring events are analyzed. In this example the number of scheduled maintenances of all machines and the number of breaks and meetings of all personnel per shift are validated by analyzing constructed Gantt charts of the activity of the machines and the personnel of the flow line.

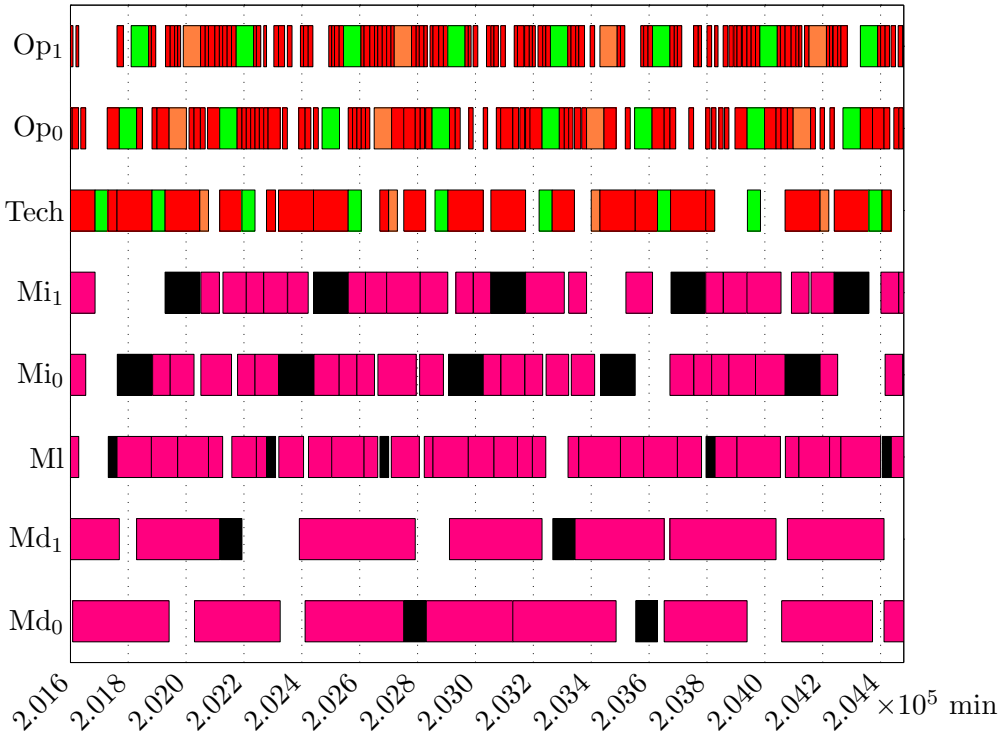


Figure C.3: *Gantt chart of the first four shifts of the flow line after twenty weeks of operating*

The legend of the charts is presented in Figure C.1. Figure C.3 presents the Gantt

chart of the first four shifts of the flow line after twenty weeks of operating. As the first shift begins, ion implantation machine 1 is receiving unscheduled maintenance. The lithography machine and the ion implantation machines therefore require one scheduled maintenance from the previous shift. They can not accept new products until the maintenance is performed. Therefore, scheduled maintenance is performed on each of them five times during the shown four shifts. The diffusion machines are not influenced by the unscheduled maintenance of the ion implantation machine, because they require scheduled maintenance only once per day instead of once per shift. Therefore, they each receive maintenance two times during the four shifts. Thus, the visualization shows the correct number of scheduled maintenances. Now the breaks and meetings of the personnel are evaluated. The technician takes nine breaks, which is one break more than is expected in four shifts. The extra break remained from the previous shift, because the technician was performing unscheduled maintenance and therefore could not take the break. The technician also attends four meetings, so he behaves as expected. Both operators each take eight breaks and attend four meetings, as they should. Thus, in the visualization using Gantt charts the model behaves correctly.

Function analysis

The final validation technique used in this project is the function analysis. The functions in the model are validated by analyzing the output of their input domain. The output should be correct over the entire domain. In this example the function *dispi* is modified for the validation to output all the allowed ion implantation machine input. The function now reads

```
func dispi(rs: nat*, xs: lot*) -> n1ls =
|[ ret [ <r,[x]> | r: nat <- rs, x: lot <- xs, x.tp /= 0 or x.hs.2 /= r ] ]|.
```

The function receives for input subsequently four times a list of machines that are requesting a product together with a list of products. The list of requesting machines varies with every function call to cover the input domain. The lists are respectively [], [0], [1] and [0,1]. The list of products is constant during all function calls and includes all combinations of product type and ion implantation machine history. The list reads

```
[<0,0,5,<|2,2,2|>,0.0>,<1,1,5,<|2,2,2|>,0.0>,<2,2,5,<|2,2,2|>,0.0>
,<3,0,5,<|2,2,0|>,0.0>,<4,1,5,<|2,2,0|>,0.0>,<5,2,5,<|2,2,0|>,0.0>
,<6,0,5,<|2,2,1|>,0.0>,<7,1,5,<|2,2,1|>,0.0>,<8,2,5,<|2,2,1|>,0.0>]
```

The entire input domain of the function is covered. The output of the four function calls is analyzed to validate the handling of the product type and the machine history of the products by this function. The output consists of respectively an empty list, all products except number 3, all products except number 6 and finally, the previous two

outputs combined. All output is explicable. The empty list is output after the first function call, because no machine has requested for a product. From respectively the second and the third function input, the test products that have been processed on ion implantation machine 0 respectively machine 1 are not allowed for input. Therefore, only product 3 respectively product 6 is not allowed for machine input. The final input combines the previous two inputs and therefore the outputs are also combined. Thus, the function behaves correctly. This concludes the validation examples. All validation of the model shows it behaves as it was designed to behave and represents the case correctly.

Appendix D

Steady state behavior

This appendix presents the steady state behavior of the flow line with required input. Four aspects of the behavior are analyzed. Firstly, the bottleneck workstation of the flow line is determined from the contents of the incoming buffers of the workstations. The diffusion and the ion implantation buffer contain few products, while the lithography buffer contains an increasing amount of products. Therefore, the lithography workstation is the slowest workstation of the flow line and is the bottleneck. Secondly, the capacity of the transporter is analyzed by administering the amount of requests that are placed for product transportation. The amount remains small throughout the simulation and therefore the transporter capacity is sufficient. Thirdly, the throughput (δ) and the mean flow time (φ) of the flow line are determined. δ is defined as the total number of products per week that reach the finish area and that leave the flow line due to unscheduled down of the ion implantation machines. The products are counted during the last one hundred weeks of simulations of two hundred weeks. To account for stochastics, the simulations are run six times. The average δ equals 47.4 products per week or 56.4% of the required δ. φ is defined as the average time a product spends in the flow line including wasted products. φ is increasing during simulation with the required flow line input, because the work in process (w) is increasing. Finally, the control of the flow line is analyzed. To this end, the activity of the machines, the personnel and the transporter during the flow of the products through the line is output. Gantt charts are constructed from the output and are analyzed. The legend of the charts is presented in Figure C.1. An example of the charts is shown in Figure D.1.

The Gantt chart visualizes the last two shifts of one hundred weeks of processing. It clearly shows that the processes have much idle time, while the required δ is not met. This results from the FIFO control of the flow line. It also shows by prolonged operating times that the machines have to wait for unloading, because the operators are unavailable. Next, it shows by analyzing the lithography machine bar and the operator 2 bar, that the lithography machine has much setup time. Finally, it shows that the lithography machine has a poor reaction to the unscheduled down of ion implantation machine 0. The poor reaction leads to much idle time of the lithography machine.

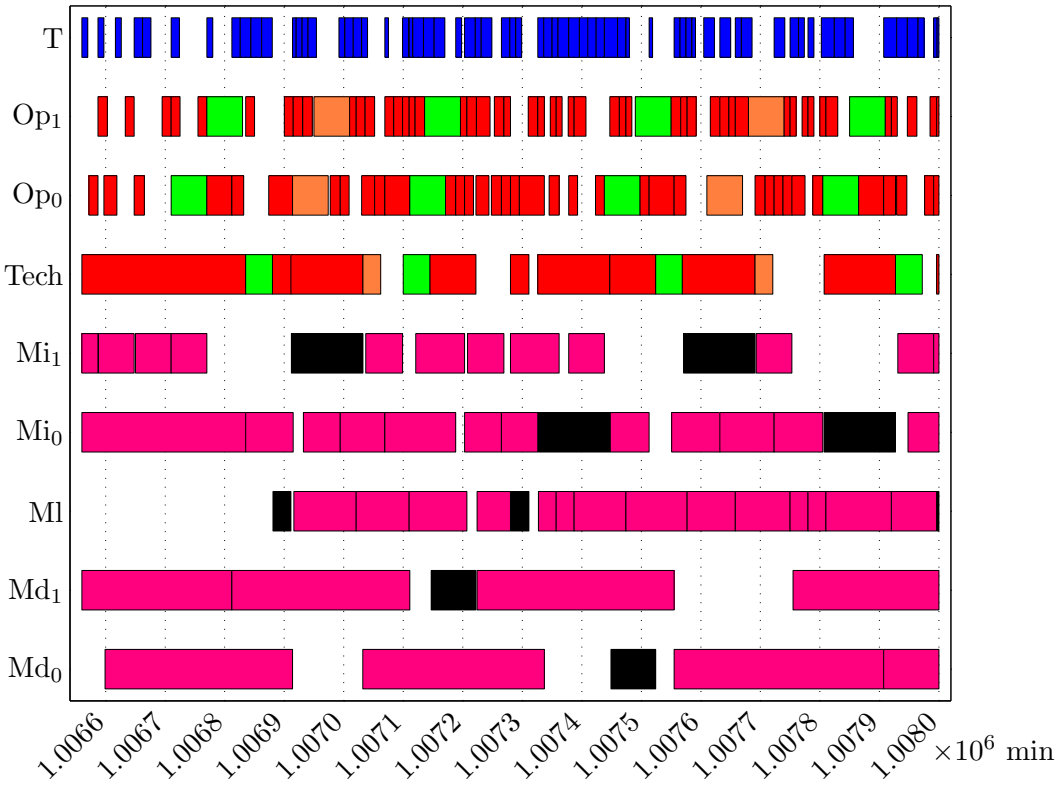


Figure D.1: *Gantt chart of the production of the last two shift of one hundred weeks of processing*

Other Gantt charts show that all machines have a poor reaction to the unscheduled down of an ion implantation machine. It can be concluded from the Gantt chart that the control of the flow line is poor. This concludes the presentation of the steady state behavior of the flow line.

Appendix E

Flow line productivity

In this appendix the productivity of the flow line with FIFO (First In First Out) control is calculated. The productivity is defined as the total amount of work performed by the line divided by the required amount of work. For a flow line with increasing work in process (w) the productivity is greater than the reached throughput (δ), because the machines perform work that is wasted on the increasing w. This work does not show in δ. The productivity is therefore a more suitable measure to quantify the amount of work that is done by a flow line with increasing w than the throughput.

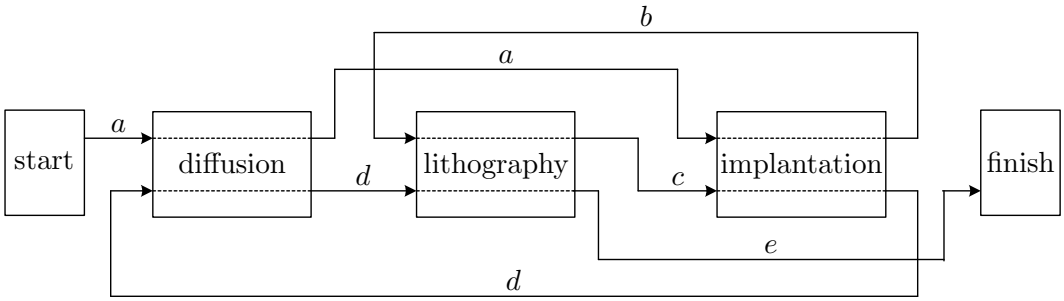


Figure E.1: *Amount of work performed by the flow line*

Figure E.1 visualizes the amount of work that is done by the flow line with required input. The variables a through e represent the amounts of products per week that flow from one area to another. To determine these amounts, the flow line is analyzed. From the start area 84 products are released into the flow line per week. The diffusion workstation processes all its input, because simulation shows that the diffusion buffer contains at most a few products. Therefore, 84 products arrive per week for the first time at the implantation workstation. Next, the implantation workstation processes all its input and wastes during processing part x of the products. The chance for unscheduled down to occur to products of both production steps is proportional to their processing times and therefore part $\frac{3}{8} \cdot x$ of the products is wasted. Then the products enter the

bottleneck workstation of the flow line, the lithography workstation, which processes only part y of its input. Because the processing sequence is FIFO (First In First Out), part y of both the low and the high production step input will be processed. Next, the products enter the implantation workstation for the second time. Again all input is processed, but this time part $\frac{5}{8} \cdot x$ of the products is wasted. Then the products are all processed for the second time by the diffusion workstation and finally they arrive at the lithography workstation for the second time. Again part y of the products is processed. Now (E.1) through (E.5) can be composed,

$$a = 84, \tag{E.1}$$
$$b = 84 \cdot (1 - \tfrac{3}{8} \cdot x), \tag{E.2}$$
$$c = 84 \cdot (1 - \tfrac{3}{8} \cdot x) \cdot y, \tag{E.3}$$
$$d = 84 \cdot (1 - \tfrac{3}{8} \cdot x) \cdot y \cdot (1 - \tfrac{5}{8} \cdot x), \tag{E.4}$$
$$e = 84 \cdot (1 - \tfrac{3}{8} \cdot x) \cdot y \cdot (1 - \tfrac{5}{8} \cdot x) \cdot y. \tag{E.5}$$

To acquire values for x and y, the number of finished products per week is counted during six simulations of two hundred production weeks, so the influence of the transient behavior of the flow line can be neglected. The counting is started after one hundred weeks. On average 43.4 finished products leave the flow line per week. Next, a system of two equations and two variables is constructed from the analysis of the production,

$$84 \cdot \frac{3}{8} \cdot x + 84 \cdot (1 - \frac{3}{8} \cdot x) \cdot y \cdot \frac{5}{8} \cdot x = 4, \tag{E.6}$$
$$84 \cdot (1 - \frac{3}{8} \cdot x) \cdot y \cdot (1 - \frac{5}{8} \cdot x) \cdot y = 43.4. \tag{E.7}$$

Firstly, the number of wasted lots is equated to four in (E.6). Secondly, the theoretical number of finished products is equated to the simulation result in (E.7). Solving the system leads to $x = 0.0576$ and $y = 0.739$. From this result can be concluded that the implantation workstation wastes 5.76% of the products that enter the workstation and that the lithography workstation processes 73.9% of its input. This calculation is checked by analyzing the contents of the lithography buffer in six simulations after one hundred weeks of processing. Due to the stochastics of the flow line, the buffer contains between 3620 and 3680 products, of which between 58.3% and 58.7% has not been processed before by the lithography machine. By analytical calculation the contents should be 3670 products, of which 58.4% has not been processed before by the lithography machine. The analytical values lie inside the intervals of the simulation values, as expected.

Now the number of products that are processed per workstation per week is known, the total work that is performed by the workstations can be determined. To this end, the run time of a machine is introduced and equals the total of the loading, the processing and the unloading time of the machine. The run times of the low and high step of the diffusion machines are respectively 285 min and 315 min. The run times of the lithography machine are 75 min and 30 min and the run times of the ion implantation

machines are 60 min and 80 min. Next, the workstation productivity is defined as the total run time of the machines in the workstation divided by the required run time. The total run time equals for the diffusion, the lithography and the implantation workstation respectively 14130 min, 5850 min and 9620 min. To calculate the required run time, x is resolved from (E.6). This time, the y is known in advance. Because all workstations process their input, y equals 1. Solving the equation leads to $x = 0.0482$. Therefore the implantation workstation now wastes 4.82% of the products that enter the workstation. The required run time equals for the diffusion, the lithography and the implantation workstation respectively 16380 min, 8590 min and 11350 min and the workstation productivity equals respectively 86.3%, 68.2% and 84.8%. The lithography machine has the least productivity and is the bottleneck of the line. The flow line productivity equals 81.5%. This concludes the calculation of the productivity of the flow line with FIFO control.

Appendix F

Maximal δ with accompanying φ

In this appendix the maximal throughput (δ_{max}) and the accompanying mean flow time (φ) of the flow line with FIFO (First In First Out) control are determined. δ_{max} will be reached when the work in process (w) is only just non-increasing in steady state, because then no work is wasted on the semifinished products that form the increasing w. In that case δ_{max} equals the flow line input.

flow line input				mfp
total	test	type A	type B	
84	3	51	30	0.457
59	2.11	35.82	21.07	0.375
58	2.07	35.21	20.72	0.372
57.5	2.05	34.9	20.55	0.369
57	2.03	34.61	20.36	0.367
56	2	34	20	0.364
55	1.97	33.39	19.64	0.361

Table F.1: *Simulation based mfp parameter values for various flow line input*

Because the input of the model is changed, the parameter that determines the time between failure of the ion implantation machines needs to be adjusted. The adjustment procedure is presented in Appendix B. Table F.1 shows the simulation based parameter values for various flow line input. The ratio of the product types is kept identical to the ratio in the case. The first column presents the total number of products released into the flow line per week. The second through fourth column present respectively the number of test products, type A products and type B products released into the flow line per week. The final column presents the simulation based parameter values.

After determining the parameter values, w measurements can be performed to determine the input of the flow line for which w is non-increasing in steady state. Since the buffer capacity constraints are not included in this analysis, it is not necessary to

determine the distribution of w over the flow line elements. Therefore, only the total w in the flow line is measured by simulation. In the simulations w is defined by the difference between the total number of products that have been released into the flow line and the total number of products that have left the line via the exit process or via an unscheduled down. To perform steady state measurements, w is measured after one hundred weeks of simulation and to account for stochastics the simulation are run six times for each flow line input. The input of the line is decreased after each set of simulation runs until w does not increase anymore during simulation. Input of 57.5 products results in an average w of 26.1, which is only just non-increasing. Therefore, δ_{max} equals 57.5 products per week or 68.5% of the required δ. This percentage is smaller than the flow line productivity percentage that is determined in Appendix E, because the non-bottleneck machines perform extra production when the flow line receives the required input.

Next to w, also φ can be measured using the determined parameter values for adjusting the time between failure. φ has already been defined in Appendix D as the average time a product spends in the flow line, including the wasted products. φ is measured during the last one hundred weeks of six simulations of two hundred weeks. For the flow line with input of 57.5 products per week, φ equals 4580 min. The simulations show that releasing 57.5 products per week into the flow line results in a strongly varying φ. This is explained by observing the influence of the stochastics of the flow line. The line approaches its maximum processing with FIFO control and therefore, the influence of the stochastics on φ is increased. This concludes δ_{max} and the accompanying φ.

Appendix G

Little's law

As part of the analysis of the flow line, Little's law [4]: $\delta = \frac{w}{\varphi}$, is applied to the line in this Appendix. The law can only be applied to a flow line in steady state with stable work in process (w). To apply this law to the flow line, the line is treated as a black box. The throughput (δ) has already been defined in Appendix D as the total number of products per week that reach the finish area and that leave the flow line due to unscheduled down of the ion implantation machines. This equals the total number of products that leave the black box per week. For stable w, δ is determined analytically and equals the flow line input. w has already been defined in Appendix F as the difference between the total number of products that have been released into the flow line and the total number of products that have left the line via the exit process or via an unscheduled down. So, w equals the total number of products that are in the black box. The mean flow time (φ) of the black box has already been defined in Appendix D as the average time a product spends in the flow line, including the wasted products. The average w and φ are measured for the flow line input of 57.5 products during six simulations of two hundred weeks. The measuring starts after one hundred weeks of production. The measured values are inserted in Little's law and δ is determined. It ranges from 57.4 to 57.5. The maximal deviation equals 0.17%, which shows the results are very accurate. Little's law has also been applied to other flow line inputs of less products per week than 57.5 and the results are just as accurate. Finally, the law has been applied to inputs of more products per week than 57.5 and then it produces an incorrect result, as expected. Little's law can thus be applied to the flow line in steady state.

Appendix H

Capacity analysis

In this appendix the required and the available capacity of the flow line elements, as described in Chapter 2, are calculated to determine if the available capacity of the elements is sufficient. According to the structure of the case description, the machines will be analyzed first. Next, the transporter will be analyzed, followed by the operators and the technician. The required capacity of the buffers is dependent of the implemented improved control. It is assumed that the available capacity of the buffers is sufficient. For simplicity the calculations will be applied to one shift instead of one week. Before they commence, the exact number of products that need to be processed per production step is calculated. Appendix E shows that the ion implantation workstation wastes 4.82% of the products during production with the required δ. The calculations are based on this percentage. The first production step is performed in the diffusion workstation and 6 products are processed per shift. The second step is performed in the implantation workstation. The number of wasted products is dependent on the length of the run time of the production step, as was described in Appendix E. Accounting for the run time of the low production step, the number of wasted products equals $\frac{5}{8} \cdot 0.0482 \cdot 6 = 0.18$. Because the distribution of the time between break down is uniform, the wasted products are on average already half processed before the break down occurs. Therefore, 5.91 products are processed for the second production step. The third step is performed in the lithography workstation. It receives 5.82 products for this step and processes all of them. The fourth step is performed in the implantation workstation. This time, $\frac{3}{8} \cdot 0.0482 \cdot 6 = 0.11$ products are wasted and again on average they will be processed half before they are wasted. Therefore, 5.77 products are processed for this step. The final two steps receive 5.71 products for input and process all products. Now, the number of products that is processed for each production step is known and the capacity analysis can begin.

Machine analysis

The run times of the machines have been determined in Appendix E and equal for the low respectively high production step 285 min and 315 min. The total required capacity of the diffusion machines equals the total run time of the low and high production step accounting for batching, which is $\frac{285}{3} \cdot 6 + \frac{315}{3} \cdot 5.71 = 1170$ min. The total available capacity of both machines considering scheduled maintenance once per day equals $720 \cdot 2 - 2 \cdot \frac{75}{2} = 1365$ min. The capacity index is defined as the required capacity divided by the available capacity and equals 0.86.

Next, to determine the required capacity of the lithography machine, first the minimal setup time per shift is computed. The setup time equals the setup time per week divided by 14. To acquire a cyclic schedule per shift and to keep the buffer contents low, a series of low and of high production step products are processed in each shift. Therefore, 28 step changes are needed per week. Furthermore, the setup time per week will be minimal if first all commercial products of one type, and then all commercial products of the other type are processed. Per switch between the two commercial product type series 5 type setups are required. The first setup changes the type to the type of the new series, because the low production step products of the new series have reached the lithography machine. Then, the type is switched back for the production of the high step series that were in the diffusion workstation at the moment of the first switch. Next, the type is again switched to the new series, because the next low production step products arrive. After that, the type is switched back again, to process the high production step series that were in the implantation buffer at the moment of the first switch. Finally, the type is switched to the new series. To acquire a cyclic schedule, two series of switches between the two commercial product type series are required per week. The test products are released in between the commercial products, because they are used to monitor the quality of the production line. Per week 3 test products are released into the flow line. Per released test product, 2 product type setups are required when the input of the lithography machine changes to test products and back to commercial products for both the low and the high production step. This amounts to 12 type setups and the total number of type setups equals 22. All type changes are assumed to be performed together with a step change, except half of the type changes involving the test products, because the type is changed directly before and after the processing of those products and the step is changed only once per half shift. The setup times are presented in Section 2.3. The minimal setup time equals $6 \cdot 5 + 12 \cdot 10 + 16 \cdot 12 = 342$ min per week or 24min per shift. From now on, all calculations are again performed per shift. Next to the setup time, also the run time of the lithography machine is included in the required capacity. The total run time of the machine is calculated according to the calculation in the previous paragraph and equals 608 min. The total required capacity then equals 632 min. The available capacity is decreased not only by scheduled maintenance, but also by forced idle time. Operator 2 is the only operator that is allowed to serve the lithography machine. When he takes his three off times per shift during the processing of the low production step, the forced idle time of the machine is minimal and equals

15 min. The available capacity now equals $720 - 15 - 30 = 675$ min and the capacity index, as defined in the previous paragraph, equals 0.94.

Finally, the total required capacity of the ion implantation machines equals 816 min. The total available capacity of the machines, considering scheduled and unscheduled maintenance, equals $1440 - 2 \cdot 120 - \frac{4}{14} \cdot 420 = 1080$ min. The capacity index equals 0.76. All machine capacity indices are less than 1, so the available capacity of the machines is sufficient.

Transporter analysis

After the capacity analysis of the machines has been performed, the analysis is directed to the transporter. The required capacity consists firstly of the loading time of the transporter, which equals 41 min. Furthermore, it consists of the transportation time and the unloading time, which respectively equal 233 min and 41 min. Finally, the required capacity consists of the time the transporter is moving idle. This time is chosen equal to the transportation time. This is a worst case scenario, because it will occur that the transporter does not have to move idle to pick up the next product. The total required capacity then equals $41 + 233 + 41 + 233 = 548$ min. The available capacity equals 720 min and therefore the capacity index is 0.76. Thus, the available transporter capacity is sufficient.

Operator analysis

The required capacity for the operators equals the total setup, loading, unloading and transportation time. In this appendix setup time has been determined to be 24 min. The transportation time is assumed to be 20 min and the loading and unloading times can be calculates using loading and unloading times of Section 2.3 and the amounts of products calculated in the first paragraph of this appendix. The loading time equals $20 \cdot \frac{11.71}{3} + 10 \cdot 11.53 + 15 \cdot 11.82 = 371$ min and the unloading time equals $40 \cdot \frac{11.71}{3} + 10 \cdot 11.53 + 15 \cdot 11.53 = 444$ min. The required capacity then equals $24 + 20 + 371 + 444 = 859$ min. The available capacity considering off time equals 1080 min and therefore the capacity index is 0.80. The individual operator capacity can not be determined, because the ion implantation machines can be operated by both operators. However, the combined available capacity is sufficient.

Technician analysis

The final flow line element that has to be analyzed is the technician. His required capacity to perform scheduled and unscheduled maintenance, using the maintenance times that are presented in Section 2.4, equals $2 \cdot \frac{75}{2} + 30 + 2 \cdot 120 + \frac{4}{14} \cdot 420 = 465$ min. His minimal transportation time per shift equals 4 min. The available capacity

considering off time equals 600 min and therefore the index is 0.78. An unscheduled maintenance can be handled together with the scheduled maintenances in two shifts, so the machines have no forced idle time. The required capacity in the two shifts in the worst case scenario equals $2 \cdot 75 + 2 \cdot 30 + 4 \cdot 120 + 480 = 1170$ min and the available capacity equals 1200 min. The capacity index then is 0.98. All the capacity indices are less than 1 and therefore the available capacity of the flow line elements is sufficient for the required δ.

Appendix I

Applied product transportation and buffer level analysis

In this appendix product transportation and buffer levels are analyzed for the schedule of Figure 4.2 to confirm, that transporter and buffer capacities are sufficient for this schedule. An example transportation scenario is used for this analysis. The transporter transports in this scenario 6 products through the entire flow line. The diffusion workstation starts with 6 high production step products in its incoming buffer and with 4 low step and 2 high step products in its outgoing buffer. The lithography workstation starts with an empty incoming buffer and with 2 low step and 4 high step products in its outgoing buffer. The ion implantation workstations starts with 3 low step products and 2 high step products in its incoming buffer and with 1 low step product in its outgoing buffer. Since the schedule is cyclic, the buffers return at the end of the schedule to their initial state. The transporter firstly performs four cycles in which it transports a product from the generator to the diffusion workstation, from diffusion to implantation, from implantation to lithography and from lithography to the exit. Then, it performs two cycles in which it transports a product from the generator to the diffusion workstation, from diffusion to lithography, from lithography to implantation and from implantation to the lithography workstation. Next, it performs four cycles in which it transports a product from the diffusion to the lithography workstation, from lithography to implantation and from implantation to the diffusion workstation. Finally, the transporter transports the remaining products as soon as they become available. Figure I.1 shows a Gantt chart of the transporter (T). The periods that the transporter is busy are colored blue. The figure shows that the transporter has much idle time.

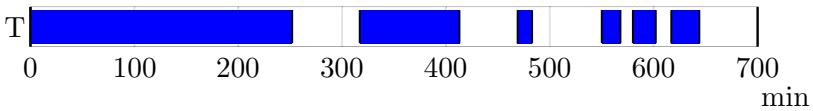


Figure I.1: *Gantt chart of the transporter*

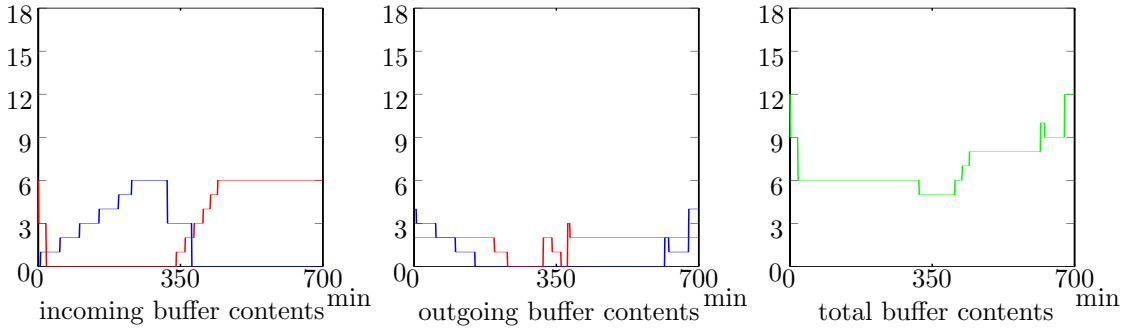


Figure I.2: *Buffer contents of the incoming and outgoing buffer of the diffusion workstation*

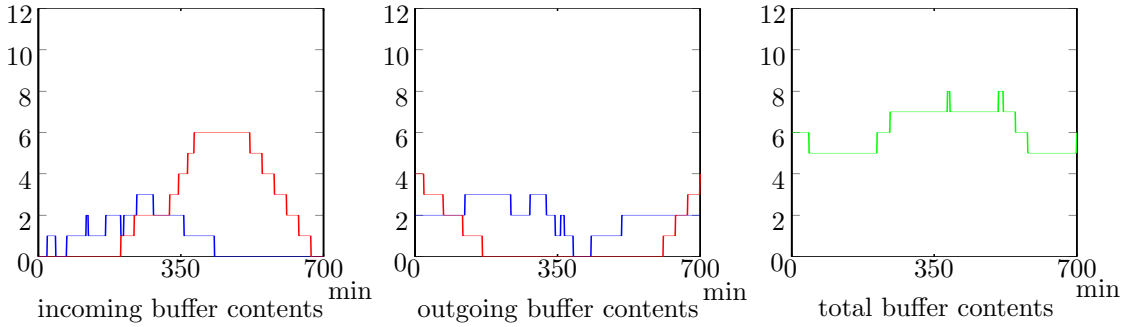


Figure I.3: *Buffer contents of the incoming and outgoing buffer of the lithography workstation*

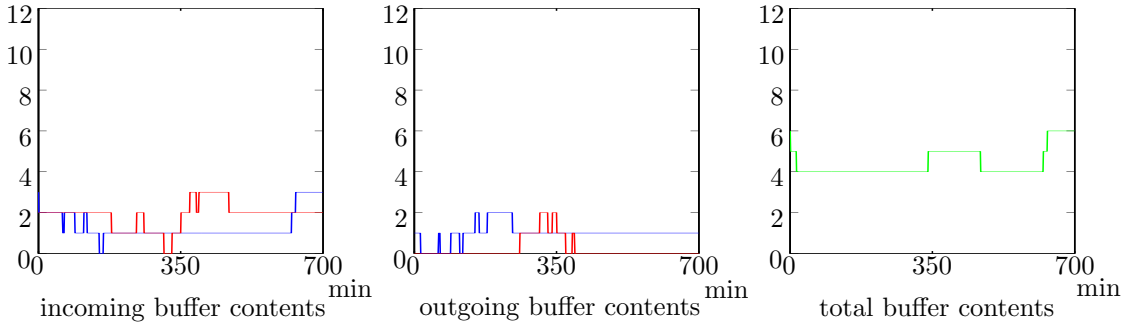


Figure I.4: *Buffer contents of the incoming and outgoing buffer of the implantation workstation*

Figures I.2 through I.4 show the contents of respectively the diffusion, lithography and implantation buffers during the production. The blue lines represent the number of products of the low production step in the buffer. The red lines represent the number of products of the high production step in the buffer. The green lines represent the total number of products in the incoming and outgoing buffers. The figures show that the buffers have enough capacity, because the total buffer level remains for the diffusion buffer below 18 products and for the lithography and the implantation buffer below 12 products. Thus can be concluded, that the transporter and buffer capacities are sufficient for the schedule of Figure 4.2, even with this suboptimal transportation sequence.

Appendix J

Computer aided analysis

To determine the feasibility of the control problem with deterministic machine failure, computer aided analysis is performed. A linear programming model of the flow line is presented in this appendix. The solving of the model will provide a lower bound on the length of the schedule of one weeks production.

In the linear model the production of 84 products per week is modelled, while machine break down is included in the model. The modelling assumptions and reductions and also the principle modelling elements are presented in Section 4.2. In that section it has been determined that a time-indexed model is used to model the case. Furthermore, two jobs are scheduled per machine run, because the loading is always directly followed by the processing and therefore the loading and processing can be combined into one job. The used job identification and duration in the model is presented in Table J.1. The first column presents the identification number of the job in the model. The second column presents the task that is represented by the job. The loading tasks of the machines also include the processing tasks. Personnel off time is scheduled as jobs on an 'off time machine'. Furthermore, the breaks and meetings of the operator, who represents both operators of the case, have equal length and require therefore only one identification number in the model. The third and fourth column present the duration of the jobs for respectively the machines and the personnel. The machines have for the load and process jobs a different job duration than the personnel, because the personnel only performs the loading part of the jobs. The job durations are five times smaller than the durations in the case, because the time discretization step in the model equals 5 min. Now, the model shall be presented.

Two types of variables are used in the model. The first type represents the main variables of the model x_{jk}. The value of the variables equals 1 if job j is started at time k and 0 otherwise. Therefore, an integer linear programming model will result. The second type is the minimization variable z. It equals the maximum completion time of all jobs. The main variables x_{jk} are used to minimize z. Next, the constraints are defined.

id	task	machine	personnel
0	load diffusion low step	49 min	4 min
1	unload diffusion low step	8 min	8 min
2	load ion implantation step low	9 min	3 min
3	unload ion implantation low step	3 min	3 min
4	load lithography low step	13 min	2 min
5	setup and load lithography low step	15 min	4 min
6	unload lithography low step	2 min	2 min
7	load ion implantation high step	13 min	3 min
8	unload ion implantation high step	3 min	3 min
9	load diffusion high step	55 min	4 min
10	unload diffusion high step	8 min	8 min
11	load lithography high step	4 min	2 min
12	setup and load lithography high step	6 min	4 min
13	unload lithography high step	2 min	2 min
14	scheduled maintenance diffusion	15 min	15 min
15	scheduled maintenance lithography	6 min	6 min
16	scheduled maintenance ion implantation	24 min	24 min
17	break technician	9 min	9 min
18	meeting technician	6 min	6 min
19	off time operator	12 min	12 min
20	unscheduled maintenance ion implantation	84 min	84 min

Table J.1: *Job properties*

The first set of constraints, used to schedule the correct number of general jobs j, is

$$\sum_{k=0}^{2016-49} x_{0k} = 28, \tag{J.1}$$

$$\sum_{k=0}^{2016-8} x_{1k} = 28, \tag{J.2}$$

$$\sum_{k=0}^{2016-9} x_{2k} = 84, \tag{J.3}$$

$$\sum_{k=0}^{2016-3} x_{3k} = 84, \tag{J.4}$$

$$\sum_{k=0}^{2016-13} x_{4k} = 70, \tag{J.5}$$

$$\sum_{k=0}^{2016-15} x_{5k} = 14, \tag{J.6}$$

$$\sum_{k=0}^{2016-2} x_{6k} = 84, \tag{J.7}$$

$$\sum_{k=0}^{2016-13} x_{7k} = 84, \tag{J.8}$$

$$\sum_{k=0}^{2016-3} x_{8k} = 84, \tag{J.9}$$

$$\sum_{k=0}^{2016-55} x_{9k} = 28, \tag{J.10}$$

$$\sum_{k=0}^{2016-8} x_{10k} = 28, \tag{J.11}$$

$$\sum_{k=0}^{2016-4} x_{11k} = 70, \tag{J.12}$$

$$\sum_{k=0}^{2016-6} x_{12k} = 14, \tag{J.13}$$

$$\sum_{k=0}^{2016-2} x_{13k} = 84, \tag{J.14}$$

$$\sum_{k=0}^{2016-15} x_{14k} = 14, \tag{J.15}$$

$$\sum_{k=0}^{2016-6} x_{15k} = 14, \tag{J.16}$$

$$\sum_{k=0}^{2016-24} x_{16k} = 28, \tag{J.17}$$

$$\sum_{k=0}^{2016-9} x_{17k} = 28, \tag{J.18}$$

$$\sum_{k=0}^{2016-6} x_{18k} = 14, \tag{J.19}$$

$$\sum_{k=0}^{2016-12} x_{19k} = 84, \tag{J.20}$$

$$\sum_{k=0}^{2016-84} x_{20k} = 4. \tag{J.21}$$

To schedule all jobs in one week, the maximal finish time of any job, accounting for the discretization step of 5 min, equals 2016 min. The maximal start time of any job equals the maximal finish time minus the production time and therefore, k runs in (J.1) through (J.21) from 0 to the maximal start time.

The second set of constraints, used to prevent machines from scheduling more jobs at a time than their capacity allows, is

$$\sum_{l=k+1-49}^{k} x_{0l} + \sum_{l=k+1-8}^{k} (x_{1l} + x_{10l}) + \sum_{l=k+1-55}^{k} x_{9l} + \sum_{l=k+1-15}^{k} x_{14l} \leq 2 \quad \forall k, \tag{J.22}$$

$$\sum_{l=k+1-13}^{k} x_{4l} + \sum_{l=k+1-15}^{k} x_{5l} + \sum_{l=k+1-2}^{k} (x_{6l} + x_{13l}) + \sum_{l=k+1-4}^{k} x_{11l} + \sum_{l=k+1-6}^{k} (x_{12l} + x_{15l}) \leq 1 \quad \forall k, \tag{J.23}$$

$$\sum_{l=k+1-9}^{k} x_{2l} + \sum_{l=k+1-3}^{k} (x_{3l} + x_{8l}) + \sum_{l=k+1-13}^{k} x_{7l} + \sum_{l=k+1-24}^{k} x_{16l} + \sum_{l=k+1-84}^{k} x_{20l} \leq 2 \quad \forall k. \tag{J.24}$$

From all jobs that are processed on the same machine there can be at most one job, (J.23), or two jobs, (J.22) and (J.24), in process at any time k. Equation (J.22) constraints the capacity of the diffusion machines, (J.23) of the lithography machine and (J.24) of the ion implantation machines.

The third set of constraints, used to prevent personnel from performing more jobs at a time than their capacity allows, is

$$\sum_{l=k+1-4}^{k} (x_{0l} + x_{5l} + x_{9l} + x_{12l}) + \sum_{l=k+1-8}^{k} (x_{1l} + x_{10l}) + \sum_{l=k+1-3}^{k} (x_{2l} + x_{3l} + x_{7l} + x_{8l}) + \sum_{l=k+1-2}^{k} (x_{4l} + x_{6l} + x_{11l} + x_{13l}) + \sum_{l=k+1-12}^{k} x_{19l} \leq 2 \quad \forall k, \tag{J.25}$$

$$\sum_{l=k+1-13}^{k} x_{14l} + \sum_{l=k+1-6}^{k} (x_{15l} + x_{18l}) + \sum_{l=k+1-24}^{k} x_{16l} + \sum_{l=k+1-9}^{k} x_{17l} + \sum_{l=k+1-84}^{k} x_{20l} \leq 1 \quad \forall k. \tag{J.26}$$

The constraints are similar to the previous set of constraints. From all the jobs that are performed by the same personnel there can be at most one, (J.26), or two, (J.25), jobs being performed at any time k. Equation (J.25) constraints the capacity of the operator and (J.26) of the technician.

The fourth set of constraints, used to introduce a relaxed precedence relation for the processing and unloading of the products into the model, is

$$\sum_{k=0}^{2016-8} k \cdot x_{1k} \geq \sum_{k=0}^{2016-49} (k+49) \cdot x_{0k}, \tag{J.27}$$

$$\sum_{k=0}^{2016-3} k \cdot x_{3k} \geq \sum_{k=0}^{2016-9} (k+9) \cdot x_{2k}, \tag{J.28}$$

$$\sum_{k=0}^{2016-2} k \cdot x_{6k} \geq \sum_{k=0}^{2016-13} \left((k+13) \cdot x_{4k} + (k+15) \cdot x_{5k}\right), \tag{J.29}$$

$$\sum_{k=0}^{2016-3} k \cdot x_{8k} \geq \sum_{k=0}^{2016-13} (k+13) \cdot x_{7k}, \tag{J.30}$$

$$\sum_{k=0}^{2016-8} k \cdot x_{10k} \geq \sum_{k=0}^{2016-55} (k+55) \cdot x_{9k}, \tag{J.31}$$

$$\sum_{k=0}^{2016-2} k \cdot x_{13k} \geq \sum_{k=0}^{2016-4} \left((k+4) \cdot x_{11k} + (k+6) \cdot x_{12k}\right). \tag{J.32}$$

The precedence relation in (J.27) through (J.32) is relaxed, because it is applied to general jobs instead of individual jobs. This means that on average the processing jobs are completed earlier than the unloading jobs are started.

The fifth set of constraints, used to prevents that first all processing jobs are scheduled and next all unloading jobs, is

$$\sum_{k=0}^{2016-8} k \cdot x_{1k} \leq \sum_{k=0}^{2016-49} (k+49) \cdot x_{0k} + \Delta, \tag{J.33}$$

$$\sum_{k=0}^{2016-3} k \cdot x_{3k} \leq \sum_{k=0}^{2016-9} (k+9) \cdot x_{2k} + \Delta, \tag{J.34}$$

$$\sum_{k=0}^{2016-2} k \cdot x_{6k} \leq \sum_{k=0}^{2016-13} ((k+13) \cdot x_{4k} + (k+15) \cdot x_{5k}) + \Delta, \tag{J.35}$$

$$\sum_{k=0}^{2016-3} k \cdot x_{8k} \leq \sum_{k=0}^{2016-13} (k+13) \cdot x_{7k} + \Delta, \tag{J.36}$$

$$\sum_{k=0}^{2016-8} k \cdot x_{10k} \leq \sum_{k=0}^{2016-55} (k+55) \cdot x_{9k} + \Delta, \tag{J.37}$$

$$\sum_{k=0}^{2016-2} k \cdot x_{13k} \leq \sum_{k=0}^{2016-4} ((k+4) \cdot x_{11k} + (k+6) \cdot x_{12k}) + \Delta. \tag{J.38}$$

The constant Δ in (J.33) through (J.38) allows a small time interval between the completion of the processing jobs and the start of the unloading jobs, because the operator will not always be directly available for unloading.

The sixth set of constraints, used to equate the lower boundary of the variable z to the maximal completion time of all jobs, is

$$z \geq (k + p_j) \cdot x_{jk} \quad \forall j, k. \tag{J.39}$$

When z is minimized, it represents the minimal length of the schedule. Because general jobs are used, in (J.39) this set can not be simplified by summing over k, because we would not get the length of the schedule for result of the minimization of z.

The final set of constraints, used to restrict the main variables x_{jk} to integer values and to present them a lower and an upper boundary, is

$$x_{jk} \in \{0, 1\} \quad j \notin \{2, 3, 7, 8, 19\}, \forall k, \tag{J.40}$$

$$x_{jk} \in \{0, 1, 2\} \quad j \in \{2, 3, 7, 8, 19\}, \forall k. \tag{J.41}$$

The upper boundary of the variables equals 2, (J.41), for the loading and unloading jobs on the ion implantation machines, because both machines can be loaded and unloaded at the same time. The boundary also equals 2 for the off time of the operators. All other main variables have an upper boundary of 1, (J.40). This concludes the integer linear programming model of the Intel case.

Appendix K

χ model with improved control

In this appendix, the final χ model of the flow line with improved control is shown. This model is based on the χ model of Appendix A. Machine break down has been removed from the model. The design and implementation of the improvements are described respectively in Section 5.1 and 5.2. The control of the generator, the incoming buffers and the personnel dispatcher of the flow line is improved.

```
from std import *
from random import *

const ih:   nat^3   = <|2,2,2|>                          // ini history
    , imi:  real^5  = <|600.0,600.0,55.0,600.0,600.0|> // ini machine idle times
    , imit: real^5  = <|0.0,0.0,0.0,0.0,0.0|>          // ini machine idle times tech
    , imst: nat^5   = <|0,0,0,0,0|>                    // ini machine status
    , imsp: nat^5   = <|0,0,2,1,1|>                    // ini machine step
    , ip:   nat^2*  = [<|0,0|>,<|2,1|>,<|2,2|>]        // ini state personnel (pos,type)
    , isdc: nat^5   = <|0,0,0,0,0|>                    // ini sched down counter
    , isdw: nat^5   = <|2,2,2,2,2|>                    // ini scheduled down window
    , isp:  nat^2   = <|2,0|>                          // ini setup (sp,tp)
    , itl:  nat     = 0                                // ini pos. transporter lots
    , itsd: real^5  = <|0.0,0.0,0.0,0.0,0.0|>          // ini sched down couter raise time
    , mca:  nat^8   = <|0,0,0,1,1,2,2,2|>              // channel to op area mapping
    , mcm:  nat^8   = <|0,0,1,2,2,3,3,4|>              // channel to machine mapping
    , mmc:  nat^5   = <|1,2,4,6,7|>                    // machine to channel mapping
    , mr:   nat^7   = <|1,3,2,3,1,2,4|>                // lot routing mapping
    , mw:   nat^5   = <|0,0,1,2,2|>                    // machine to workstation mapping
    , pim:  bool^8  = <|false,true,true,false,true,false,true,true|> // process is machine
    , rot:  nat^3^2 = <|<|2,2,2|>,<|1,1,1|>|>          // required off-times
    , tasd: real^5  = <|1440.0,1440.0,720.0,720.0,720.0|> // average time betw sched down
    , nos:  real^3  = <|3.0,51.0,30.0|>                // nr of starts per week
    , to:   nat^3^2 = <|<|45,60,60|>,<|30,60,60|>|>    // off times (break,meeting)
    , te:   nat^6   = <|225,30,55,50,255,10|>          // process times
    , tol:  nat^3   = <|20,10,15|>                     // op load times
    , tou:  nat^3   = <|40,10,15|>                     // op unload times
    , trun: nat^6   = <|285,60,75,80,315,30|>          // run times
    , tsd:  nat^3   = <|75,30,120|>                    // scheduled down times
    , tsh:  nat     = 720                              // shift time
    , tsp:  nat^4   = <|0,5,10,12|>                    // setup times (none,tp,sp,both)
    , ttlu: nat     = 2                                // transp loading + unloading time
    , ttr:  nat^2   = <|4,1|>                          // transport times (transp,pers)
```

```
type lot  = id.nat#tp.nat#sp.nat#hs.nat^3#st.real
          // identification#type (0,1,2)#step (0..6)#history#starttime
   , n1ls = nat#lot*
   , n2ls = nat#nat#lot*
   , n3ls = nat#nat#nat#lot*
   , n4ls = nat#nat#nat#nat#lot*

proc Gs(a: !void, n: nat) =
|[ ti: real
 | [ n = 0 -> ti:= 10080/nos.0 | n = 1 -> ti:= 10080/(nos.1 + nos.2) ]
 ; *[ true -> a!; delta ti ]
]|

proc Gc(a: (?void)^2, b: !void, c: !lot*) =
|[ xs: lot*, i,n,lt: nat
 | xs:= []; i:= 0; n:= 51; lt:= 1
 ; *[ j: nat <- 0..2: true; a.j?
      -> [ j = 0 -> xs:= xs ++ [<i,j,0,ih,time>]
         | j = 1 -> xs:= xs ++ [<i,lt,0,ih,time>]; n:= n - 1
                  ; [ n = 0 -> [ lt = 1 -> lt:= 2; n:= 30 | lt = 2 -> lt:= 1; n:= 51 ]
                    | n > 0 -> skip
                    ]
         ]; i:= i + 1; b!
    | len(xs) > 0; c![hd(xs)] -> xs:= tl(xs)
    ]
]|

clus G(a: !void, b: !lot*) = |[ c: (-void)^2 | j: nat <- 0..2: Gs(c.j,j) || Gc(c,a,b) ]|

proc Tpld(a: (?void)^4, b: !nat) =
|[ xs: nat*
 | xs:= []
 ; *[ j: nat <- 0..4: true; a.j? -> xs:= xs ++ [j]
    | len(xs) > 0; b!hd(xs)      -> xs:= tl(xs)
    ]
]|

func tt(ft: nat^2, n,k: nat) -> int = |[ ret n * abs(+ft.0 - ft.1) + k ]|

proc Tpl(a: ?nat, b: (?lot*)^4, c: (!lot*)^4) =
|[ ft: nat^2, xs: lot*
 | ft.0:= itl
 ; *[ true
      -> a?ft.1; delta tt(ft,ttr.0,0); b.(ft.1)?xs; ft:= <| ft.1, mr.(hd(xs).sp) |>
       ; delta tt(ft,ttr.0,ttlu); c.(ft.1 - 1)!xs; ft.0:= ft.1
    ]
]|

clus Tl(a: (?void)^4, b: (?lot*)^4, c: (!lot*)^4) =
|[ d: -nat | Tpld(a,d) || Tpl(d,b,c) ]|

func sel(xs: n1ls*, cs: bool, n,maxn: nat) -> (nat#lot*)#bool#nat =
|[ zs: n1ls*, x: lot, b: bool
 | [ len(xs) = 0 -> ret <<0,[]>,cs,n>
   | len(xs) > 0
     -> zs:= xs
      ; *[ len(zs) > 0
           -> x:= hd(hd(zs).1); b:= x.sp < 3 and not cs or x.sp >= 3 and cs
            ; [ b and n < maxn     -> ret <hd(zs),cs,n+1>
              | b and n = maxn     -> zs:= tl(zs)
              | not b and n < maxn -> zs:= tl(zs)
```

```
              | not b and n = maxn -> ret <hd(zs),cs,1>
              ]
         ]
       ; [ b     -> ret <hd(xs),not cs,n>
         | not b -> ret <hd(xs),not cs,1>
         ]
   ]
]|

func feasbat(r: nat, x,y,z: lot) -> bool =
|[ testlots: lot*, a,b,c: bool, n: nat
 | testlots:= [ p | p: lot <- [x,y,z], p.tp = 0 ]; n:= len(testlots)
 ; a:= x.tp = y.tp; b:= x.tp = z.tp; c:= y.tp = z.tp
 ; [ n >= 2              -> ret false
   | n < 2 and x.sp < 3  -> ret true
   | n = 0 and x.sp >= 3 -> ret a and b
   | n = 1 and x.sp >= 3 -> ret (a or b or c) and hd(testlots).hs.0 /= r
   ]
]|

func dispd(rs: nat*, xs: lot*, cs: bool, n,maxn: nat) -> (nat#lot*)#bool#nat =
|[ ret sel([ <r,[x,y,z]>
           | r: nat <- rs
           , x: lot <- xs, y: lot <- xs, x.id < y.id, x.sp = y.sp
           , z: lot <- xs, y.id < z.id, y.sp = z.sp
           , feasbat(r,x,y,z)
           ],cs,n,maxn)
]|

func dispi(rs: nat*, xs: lot*, cs: bool, n,maxn: nat) -> (nat#lot*)#bool#nat =
|[ ret sel([ <r,[x]> | r: nat <- rs, x: lot <- xs, x.tp /= 0 or x.hs.2 /= r ],cs,n,maxn) ]|

proc Bmi(a: ?lot*, b: (?void)^2, c: !n2ls, d: ?n2ls
        , e: (!n1ls)^2, disp: (nat*,lot*,bool,nat,nat) -> (nat#lot*)#bool#nat, maxn: nat) =
|[ xs,ys,zs: lot*, n,p,r: nat, rs: nat*, rzss: (n1ls)*, ncs,cs: bool
 | xs:= []; rs:= []; rzss:= []; cs:= false; n:= 0
 ; *[ true
     -> <<r,zs>,ncs,n>:= disp(rs,xs,cs,n,maxn)
      ; [ true; a?ys                -> xs:= xs ++ ys
        | j: nat <- 0..2: true; b.j? -> rs:= rs ++ [j]
        | len(zs) > 0; c!<1,r,zs>
          -> rzss:= rzss ++ [<r,zs>]; rs:= rs -- [r]; xs:= xs -- zs; cs:= ncs
        | true; d?<p,r,zs>           -> rzss:= rzss -- [<r,zs>]; e.r!<p,zs>
        ]
   ]
]|

proc Bmo(a: (?n1ls)^2, b: !nat^2, c: !void, d: !lot*) =
|[ xs,ys: lot*, i,p: nat
 | xs:= []
 ; *[ j: nat <- 0..2: true; a.j?<p,ys>
     -> xs:= xs ++ ys; b!<|p,j|>; i:= len(ys); *[ i > 0 -> c!; i:= i - 1 ]
   | len(xs) > 0; d![hd(xs)]
     -> xs:= tl(xs)
   ]
]|

clus Bm(a: ?lot*, b: (?void)^2, c: !n2ls, d: ?n2ls, e: (!n1ls)^2, f: (?n1ls)^2, g: !nat^2
       , h: !void, i: !lot*, disp: (nat*,lot*,bool,nat,nat) -> (nat#lot*)#bool#nat, maxn: nat) =
|[ Bmi(a,b,c,d,e,disp,maxn) || Bmo(f,g,h,i) ]|

func displ(xs:lot*, cs,ct,n:nat) -> lot*#nat#nat#nat =
```

```
|[ zs: lot*
 | zs:= [ y | y: lot <- xs, y.sp = cs, y.tp = ct, n < 6 ] ++
        [ y | y: lot <- xs, y.sp = cs, y.tp /= ct, y.tp /= 0, n < 6 ] ++
        [ y | y: lot <- xs, y.sp = cs, y.tp /= ct, y.tp = 0, n < 6 ] ++
        [ y | y: lot <- xs, y.sp /= cs, y.tp = ct ] ++
        [ y | y: lot <- xs, y.sp /= cs, y.tp /= ct, y.tp /= 0 ] ++
        [ y | y: lot <- xs, y.sp /= cs, y.tp /= ct, y.tp = 0 ]
 ; [ len(zs) = 0  -> ret <[],cs,ct,n>
   | len(zs) /= 0
     -> [ hd(zs).sp = cs  -> ret <[hd(zs)],hd(zs).sp,hd(zs).tp,n+1>
        | hd(zs).sp /= cs -> ret <[hd(zs)],hd(zs).sp,hd(zs).tp,1>
        ]
   ]
]|

proc Bsi(a: ?lot*, b: ?void, c: !n2ls, d: ?n2ls, e: !n1ls) =
|[ xs,ys,zs: lot*, rq: bool, p,q,ncs,cs,nct,ct,n,nn: nat
 | xs:= []; rq:= false; cs:= isp.0; ct:= isp.1; n:= 0
 ; *[ true
      -> <zs,ncs,nct,nn>:= displ(xs,cs,ct,n)
       ; [ true; a?ys                  -> xs:= xs ++ ys
         | true; b?                    -> rq:= true
         | rq and zs /= []; c!<1,0,zs> -> rq:= false; cs:= ncs; ct:= nct; n:= nn
         | true; d?<p,q,ys>            -> xs:= xs -- ys; e!<p,ys>
         ]
    ]
]|

proc Bso(a: ?n1ls, b: !nat^2, c: !void, d: !lot*) =
|[ xs,ys: lot*, p: nat
 | xs:= []
 ; *[ true; a?<p,ys>          -> xs:= xs ++ ys; b!<|p,0|>; c!
    | len(xs) > 0; d![hd(xs)] -> xs:= tl(xs)
    ]
]|

clus Bs(a: ?lot*, b: ?void, c: !n2ls, d: ?n2ls, e: !n1ls, f: ?n1ls, g: !nat^2, h: !void
       , i: !lot*) =
|[ Bsi(a,b,c,d,e) || Bso(f,g,h,i) ]|

func su(psp,ptp,sp,tp: nat) -> nat =
|[ [ sp = psp and tp = ptp   -> ret tsp.0
   | sp = psp and tp /= ptp  -> ret tsp.1
   | sp /= psp and tp = ptp  -> ret tsp.2
   | sp /= psp and tp /= ptp -> ret tsp.3
   ]
]|

func updm(x: lot, wi,mi: nat) -> lot = |[ x.sp:= x.sp + 1; x.hs.wi:= mi; ret x ]|

proc M(a: !void, b: ?n1ls, c: !nat^2, d: !n2ls
      , e: ?n2ls, f: !n1ls, wi,mi: nat) =
|[ p,q,psp,ptp,sp,tp: nat, ud: bool, qs,xs: lot*
 | <|psp,ptp|>:= isp; ud:= false; a!
 ; *[ true; b?<p,xs>
      -> [ wi = 1
           -> sp:= hd(xs).sp; tp:= hd(xs).tp; delta su(psp,ptp,sp,tp); psp:= sp; ptp:= tp
         | wi /= 1 -> skip
         ]
       ; delta tol.wi; c!<|p,0|>; delta te.(hd(xs).sp); d!<1,0,[]>; e?<p,q,qs>; delta tou.wi
       ; f!<p,[ updm(x,wi,mi) | x: lot <- xs ]>; a!
    | true; e?<p,q,qs>
```

```
      -> delta tsd.wi; c!<|p,0|>
    ]
]|

func doscm(n: nat, msp: nat^5) -> bool =
|[ [ n < 3  -> ret true | n >= 3 -> ret msp.2 = 5 ] ]|

func feaswp(w,j,p: nat) -> bool =
|[ ret w = p and (w = 0 or w = 1 and (j < 3 or j > 4)) or w = 2 and p = 1 and j > 2 ]|

func saw(j,p: nat, ot: nat^3^2, mi: real^5, rs: (n3ls)*, t: real) -> bool =
|[ b0,b1: nat*
 | b0:= [ 0 | r: n3ls <- rs, r.0 < 3 ]; b1:= [ 0 | r: n3ls <- rs, r.0 >= 3, r.0 < 5 ]
 ; [ p = 0 -> ret true
   | p = 1 -> ret j < 3 or j > 4 and len(b0) = 0
   | p = 2
     -> ret j < 5 or j >= 5 and not (mi.2 - t = 55 and (ot.0.2 > 0 or ot.1.2 > 0)) and len(b1) = 0
   ]
]|

func tkot(p: nat, mi,mit: real^5, sdc: nat^5, t: real) -> bool =
|[ ns: nat*
 | [ p = 0 -> ns:= [ n | n: nat <- [0,1,2], sdc.n > 0, mit.n - t <= 30 ]; ret len(ns) = 0
   | p = 1 -> ns:= [ n | n: nat <- [0,1], mi.n - t <= 20 ]; ret len(ns) = 0
   | p = 2 -> ret mi.2 - t = 55
   ]
]|

func dispp(ot: nat^3^2, rs: (n3ls)*, wps: nat^2*, mst,sdw,sdc,msp: nat^5, mi,mit: real^5, t: real)
          -> (nat#nat^2)*#((n3ls)#nat^2)*#nat^5 =
|[ ops: (nat#nat^2)*, dps: ((n3ls)#nat^2)*, dp: (n3ls)#nat^2, k,m: nat
 | dps:= [ <<mmc.n,0,0,[]>,w> | w: nat^2 <- wps, w.1 = 0, n: nat <- [0,1,2,3,4], mst.n = 0
                              , sdw.n /= 1, doscm(n,msp) or sdw.n = 0 ] ++
         [ <<j,p,q,xs>,w> | w: nat^2 <- wps, <j,p,q,xs>: n3ls <- rs, feaswp(w.1,j,p), xs /= []
                          , saw(j,w.1,ot,mi,rs,t), sdw.(mcm.j+q) > 0 or pim.j ] ++
         [ <<j,p,q,xs>,w> | w: nat^2 <- wps, <j,p,q,xs>: n3ls <- rs, feaswp(w.1,j,p), xs = []
                          , saw(j,w.1,ot,mi,rs,t), sdw.(mcm.j+q) > 0 or pim.j ]
 ; [ len(dps) = 0
     -> ops:= [ <n,w> | w: nat^2 <- wps, n: nat <- [0,1], ot.n.(w.1) > 0, tkot(w.1,mi,mit,sdc,t) ]
   | len(dps) > 0
     -> ops:= []; dp:= hd(dps); k:= dp.0.1; m:= mcm.(dp.0.0)
      ; [ k = 0 and mst.m = 0 -> sdw.m:= 0 | k /= 0 or mst.m /= 0 -> skip ]
   ]
   ; ret <ops,dps,sdw>
]|

func pns(x,y: real#nat) -> real#nat = |[ [ x.0 <= y.0 -> ret x | x.0 > y.0 -> ret y ] ]|

func nsdwe(sdw: nat^5, twc,two: real^5) -> (real#nat)* =
|[ nsdwns: (real#nat)*
 | nsdwns:= [ <twc.n,n> | n: nat <- [0,1,2,3,4], sdw.n = 2 ] ++
            [ <two.n,n> | n: nat <- [0,1,2,3,4], sdw.n = 1, twc.n - tasd.n <= two.n ] ++
            [ <twc.n-tasd.n,n> | n: nat <- [0,1,2,3,4], sdw.n = 1, twc.n - tasd.n > two.n ]
 ; [ len(nsdwns) = 0 -> ret [] | len(nsdwns) > 0 -> ret [fold(tl(nsdwns),pns,hd(nsdwns))] ]
]|

func nsde(tnsd: real^5) -> real#nat#real^5 =
|[ sdns: (real#nat)*, tsdc: real, u: nat
 | sdns:= [ <tnsd.n,n> | n: nat <- [0,1,2,3,4] ]
 ; <tsdc,u>:= fold(tl(sdns),pns,hd(sdns)); tnsd.u:= tnsd.u + tasd.u; ret <tsdc,u,tnsd>
]|
```

```
func updp(j,p,q: nat, mst,sdc,sdw,msp: nat^5, twc,two,mi: real^5, t: real)
         -> nat^5#nat^5#nat^5#real^5#real^5#real^5 =
|[ n: nat, trem: real
 | n:= mcm.j + q
 ; [ p > 0 and mst.n = 1 -> mi.n:= t + te.(msp.n)
   | p = 0 or mst.n /= 1 -> skip
   ]
 ; [ p = 0 and mst.n = 1
     -> sdw.n:= 1; sdc.n:= sdc.n - 1; two.n:= t + (tasd.n)/2; mst.n:= 0
      ; trem:= t + tasd.n - rmod(t,tasd.n)
      ; [ sdc.n = 0 -> twc.n:= trem + tasd.n
        | sdc.n = 1 -> twc.n:= max(two.n,trem)
        | sdc.n > 1 -> twc.n:= two.n
        ]
   | p > 0 or mst.n /= 1
     -> mst.n:= mst.n mod 2
   ]
 ; ret <mst,sdc,sdw,twc,two,mi>
]|

proc Pd(a: (?n2ls)^8, b: (?nat^2)^8, c: ?nat^2*, d: !nat#nat^2, e: !n4ls) =
|[ dps: ((n3ls)#nat^2)*, rs: (n3ls)*, wps,ws: nat^2*, w: nat^2, mst,nsdw,sdc,sdw,msp: nat^5
 , ops: (nat#nat^2)*, op: nat#nat^2, r: n3ls, m,n,p,q,u,tno: nat, xs: lot*
 , nsdwes: (real#nat)*, ntnsd,tnsd,twc,two,mi,mit: real^5, ot: nat^3^2, tsdc: real
 | rs:= []; wps:= ip; mst:= imst; twc:= tasd; sdc:= isdc; sdw:= isdw; ot:= rot
 ; tno:= tsh; tnsd:= itsd; msp:= imsp; mi:= imi; mit:= imit
 ; *[ true
      -> <ops,dps,nsdw>:= dispp(ot,rs,wps,mst,sdw,sdc,msp,mi,mit,time)
       ; nsdwes:= nsdwe(sdw,twc,two); <tsdc,u,ntnsd>:= nsde(tnsd)
       ; [ j: nat <- 0..8: true; a.j?<p,q,xs>
           -> rs:= rs ++ [<j,p,q,xs>]
         | j: nat <- 0..8: true; b.j?<|p,q|>
           -> wps:= wps ++ [<|j,p|>]
            ; <mst,sdc,sdw,twc,two,mi>:= updp(j,p,q,mst,sdc,sdw,msp,twc,two,mi,time)
         | true; c?ws
           -> wps:= wps ++ ws
         | len(ops) > 0; d!hd(ops)
           -> op:= hd(ops); wps:= wps -- [op.1]; m:= op.0; n:= op.1.1; ot.m.n:= ot.m.n - 1
         | len(dps) > 0; e!<hd(dps).1.0,hd(dps).0.0,hd(dps).1.1,hd(dps).0.2,hd(dps).0.3>
           -> <r,w>:= hd(dps); rs:= rs -- [r]; wps:= wps -- [w]; n:= mcm.(r.0) + r.2
            ; [ mst.n = 0 and w.1 /= 0 -> msp.n:= hd(r.3).sp; mit.n:= time + trun.(msp.n)
              | mst.n = 0 and w.1 = 0  -> mi.n:= time + tsd.(mw.n)
              | mst.n = 1 and w.1 /= 0 -> skip
              ]
            ; mst.n:= mst.n + 1; sdw:= nsdw
         | len(nsdwes) > 0; delta hd(nsdwes).0 - time
           -> n:= hd(nsdwes).1; sdw.n:= (sdw.n + 1) mod 3
         | true; delta tsdc - time
           -> sdc.u:= sdc.u + 1; tnsd:= ntnsd
         | true; delta tno - time
           -> tno:= tno + tsh; m:= 2
            ; *[ m > 0
                 -> m:= m - 1; n:= 3; *[ n > 0 -> n:= n - 1; ot.m.n:= ot.m.n + rot.m.n ]
               ]
         ]
   ]
]|

func ptpp(x,y: real#nat#nat#nat#lot*) -> bool = |[ ret x.0 <= y.0 ]|

proc Pt(a: ?n4ls, b: (!n2ls)^8) =
|[ xs: (real#nat#nat#nat#lot*)*, p,q,u,v: nat, ys:lot*, x: real#nat#nat#nat#lot*
```

```
 | xs:= []
 ; *[ true; a?<u,v,p,q,ys>
      -> xs:= insert(xs,<time+tt(<|mca.u,mca.v|>,ttr.1,0),v,p,q,ys>,ptpp)
    | len(xs) > 0; delta hd(xs).0 - time
      -> x:= hd(xs); b.(x.1)!<x.2,x.3,x.4>; xs:= tl(xs)
    ]
]|

func ppo(x,y: real#nat^2) -> bool = |[ ret x.0 <= y.0 ]|

proc Po(a: ?nat#nat^2, b: !nat^2*) =
|[ os: (real#nat^2)*, op: nat^2, m: nat
 | os:= []
 ; *[ true; a?<m,op>                     -> os:= insert(os,<time+to.m.(op.1),op>,ppo)
    | len(os) > 0; delta hd(os).0 - time -> b![hd(os).1]; os:= tl(os)
    ]
]|

clus P(a: (?n2ls)^8, b: (?nat^2)^8, c: (!n2ls)^8) =
|[ d: -nat^2*, e: -nat#nat^2, f: -n4ls
 | Pd(a,b,d,e,f) || Pt(f,c) || Po(e,d)
]|

proc E(a: ?lot*) = |[ xs: lot* | *[ true -> a?xs ] ]|

clus Intelimproved()=
|[ a: (-void)^4, b,c: (-lot*)^4, d,n: (-void)^2, e,f: (-n2ls)^8, g,i,o,p: (-n1ls)^2
 , h: (-nat^2)^8, l,m: -n1ls, k: -void
 | G(a.0,b.0)
|| Tl(a,b,c)
|| Bm(c.0,d,e.0,f.0,g,i,h.0,a.1,b.1,dispd,2)
|| j: nat <- 0..2: M(d.j,g.j,h.(j+1),e.(j+1),f.(j+1),i.j,0,j)
|| Bs(c.1,k,e.3,f.3,l,m,h.3,a.2,b.2)
|| M(k,l,h.4,e.4,f.4,m,1,0)
|| Bm(c.2,n,e.5,f.5,o,p,h.5,a.3,b.3,dispi,6)
|| j: nat <- 0..2: M(n.j,o.j,h.(j+6),e.(j+6),f.(j+6),p.j,2,j)
|| P(e,h,f)
|| E(c.3)
]|

xper = |[Intelimproved()]|
```