Capacity Planning in a Semiconductor Supply Chain

R. ten Cate

SE 420531

Final project

Supervisor: Prof.dr.ir. J.E. Rooda
Coaches: Dr.ir. E.J.J. van Campen (NXP)
Dr.ir. A.A.J. Lefeber

Eindhoven University of Technology
Department of Mechanical Engineering
Systems Engineering Group

Eindhoven – March 2008

FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mechanical Engineering
Systems Engineering Group

February 2008

Student	R. ten Cate
Supervisor	Prof.dr.ir. J.E. Rooda
Advisors	Dr.ir. E.J.J. van Campen (NXP) Dr.ir. A.A.J. Lefeber
Start	January 2007
Finish	December 2007
Title	Capacity Planning in a Semiconductor Supply Chain

Subject

NXP is one of the top 10 semiconductor manufacturers in the world. Manufacturing operations are separated into a front end and a back end part. The front end of the production process deals with IC fabrication, sort and wafer cutting. The back end consists of assembly, testing and finish of the chip. For front end operations, multiple production facilities are used (around 25, of which 9 internally), and almost 400 different production routings in about 30 separate technologies. It has to deal with highly uncertain demand and long lead times. As a result, the capacity planning problem for the front end is a challenging one. At present, this problem is tackled through manual analysis by experienced planners.

In his article 'Managing Supply-Demand Networks in Semiconductor Manufacturing', Kempf describes an approach for 'the integration of optimal decision making and controlling decision execution in the manufacturing core of a supply-demand-network.' Hereto, he splits the problem into a strategic planning function and a tactical execution function. The strategic planning function deals with long term planning by means of a linear programming formulation. The tactical execution function manages daily operations through model predictive control. Used together, the two functions would achieve efficient operation of the network. Kempfs approach could also be used in the industrial planning department at NXP, and it might provide an improvement to present methods.

Assignment

The possibilities of applying Kempfs approach at NXP have to be investigated. Initially, the focus should be on the strategic planning function, applied to the front end. The capacity planning problem of NXP has to be modeled and optimized using linear programming. The optimization results have to be compared to the results of present methods. The inventory planning is to be modeled after Kempfs formulation. Again, a comparison has to be made to present methods. Possibilities and desirability of using the tactical execution function at NXP have to be investigated. A report has to be presented at the end of the investigation, containing research results and recommendations.

Prof.dr.ir. J.E. Rooda Dr.ir. E.J.J. van Campen (NXP) Dr.ir. A.A.J. Lefeber

Systems Engineering Department of Mechanical Engineering

Summary

The semiconductor manufacturer NXP belongs to the top 10 in its industry worldwide. Its head office is in Eindhoven, the Netherlands. Their production process is separated into a front end and a back end. In this report, focus is on the front end. Within front end, almost 400 different product types are in production in about 30 different technologies. For production, front end utilizes around 25 facilities all over the world, of which 9 owned and operated solely by NXP. Front end has to deal with long lead times and, because of the fast developments in the semiconductor sector, uncertain demand.

Within NXP, the department of Industrial Planning deals with capacity planning. Capacity planning fits desired production to available capacity. As much as possible of the (forecast) demand has to be fulfilled, but production capacity is limited. Facilities have a limited capacity, and because of specialization, not all factories are able to manufacture all products. Capacity planning determines which products have to be produced in what amounts at which locations, so that demand and constraints are met as best as possible. With a large number of factories, products and constraints, as is the case at NXP, this may be a complex problem.

At present, that problem is solved manually at NXP, with previous plans as reference. However, there are other ways to solve the problem. One such way is proposed by K. Kempf. As part of a planning and control strategy for optimization of the entire semiconductor manufacturing process, he proposes to use a strategic planning function. That function optimizes the long term planning, NXPs capacity planning, by means of linear programming.

The goal of this project is to find out if Kempfs approach can also be used at NXP, notably for the front end. NXP has supplied a test case of 5 factories and 5 production technologies, for which a capacity plan has to be created through linear programming. To this end, the practical problem has to be translated to a mathematical one. An objective function has to be determined and constraints have to be specified.

In this report, the problem is presented as a cost minimization. Total cost of a plan has to be minimized, while the constraints have to be obeyed. Decision variables are the number of products manufactured for all products, in all fabs, in all periods in the planning horizon. Inventory, backlog and sales numbers are introduced as extra variables.

In total, five constraints are specified. Two of them are equality constraints, coupling the values of the extra variables to the decision variable values. The remaining three are inequality constraints, that set the system bounds. All five constraints:

- Inventory balance
- Sales-balance
- Capacity constraint
- Sourcing constraint
- Lowerbounds

The inventory balance links the values of production, inventory and backlog variables. The sales balance relates inventory, production and sales. The capacity constraint gives the maximum capacity of facilities for different technologies. The sourcing constraint indicates which products can be produced at which locations. The final constraint gives a lowerbound to the variables, none of which can be negative. The objective function is specified as the sum of costs of production, inventory and backlog, minus the sum of revenue of sales. The objective function value has to be minimized.

All data needed has been supplied by NXP, but it is in many different formats. A standard format is proposed, and all data is transformed into that format. With the given data, an implementation of the linear program model is created in Matlab. LPsolve is used as solver. After transforming the equations to the format desired by LPsolve, the problem can be solved. Analysis of the output indicates that the model behaves correctly. Moreover, the calculated solution meets the expectations very well.

This shows that the Kempf approach can be implemented at NXP. However, some recommendations can be made to improve the proposed model. Also, there are several ways of extending and continuing the project.

Samenvatting (in Dutch)

De chip-fabrikant NXP, gevestigd in Eindhoven, Nederland, behoort wereldwijd tot de top 10 in de halfgeleider industrie. In het fabricageproces maakt het bedrijf onderscheid tussen *front end* en *back end*, oftewel het voorste deel en het achterste deel van het proces. De focus ligt in dit verslag op het front end. Binnen front end worden rond de 400 verschillende (half-)producten gefabriceerd in ongeveer 30 productietechnologieën. Hiervoor gebruikt NXP zo'n 25 productielocaties wereldwijd, waarvan 9 volledig in eigen beheer. Het front end heeft te maken met lange doorvoertijden en, vanwege de snelle ontwikkelingen in de halfgeleider industrie, onzekere vraag.

Binnen NXP houdt de afdeling Industrial Planning zich bezig met de capaciteitsplanning. Binnen de capaciteitsplanning worden de gewenste productie en de capaciteit op elkaar afgestemd. Een zo groot mogelijk deel van de (voorspelde) vraag dient vervuld te worden, maar er zijn beperkingen in de productiecapaciteit. Fabrieken hebben een beperkte capaciteit, en als gevolg van specialisatie zullen niet alle fabrieken in staat zijn alle producten te maken. In de capaciteitsplanning wordt bepaald welke producten op welke locaties in welke hoeveelheden gemaakt dienen te worden, zodat aan de randvoorwaarden èn zoveel mogelijk aan de vraag wordt voldaan. Vooral als het aantal producten, fabrieken en randvoorwaarden groot wordt, zoals bij NXP, kan dit een complex probleem worden.

Momenteel wordt dat probleem bij NXP handmatig opgelost aan de hand van voorgaande planningen. Er zijn echter ook andere manieren om het probleem aan te pakken. Eèn zo'n manier wordt voorgesteld door K. Kempf. Als onderdeel van een plan- en regelstrategie voor optimalisatie van het gehele chip-fabricageproces stelt hij het gebruik voor van een strategische planningsfunctie. Die functie optimaliseert de lange-termijnplanning, de capaciteitsplanning van NXP, middels lineair programmeren.

Het doel van dit project is om te zien of de aanpak van Kempf ook bij NXP toegepast kan worden, met name op het front end. NXP heeft een test case aangeleverd van 5 fabrieken en 5 technologieën, waarvoor een capaciteitsplan gemaakt dient te worden via lineair programmeren. Hiertoe moet het praktische probleem vertaald worden naar een wiskundig probleem. Een doelfunctie moet vastgesteld worden en de randvoorwaarden moeten worden gespecificeerd.

In dit verslag wordt het probleem gepresenteerd als een kosten-minimalisatie. De totale

kosten van een planning moeten zo laag mogelijk gehouden worden, terwijl aan de randvoorwaarden wordt voldaan. Beslissingsvariabelen zijn de productieaantallen van de producten in de verschillende fabrieken, gedurende alle tijdsperioden in de planning. Als hulpvariabelen worden inventaris, backlog en verkoopaantallen gebruikt.

In totaal zijn vijf randvoorwaarden gespecificeerd. De eerste twee zijn gelijkheidsconstraints, die de waarden van de hulpvariabelen koppelen aan die van de beslissingsvariabelen. De overige drie zijn ongelijkheids-voorwaarden, die de systeemgrenzen aangeven. De vijf constraints:

- Inventaris-balans
- Verkoop-balans
- Capaciteits begrenzing
- Productiemogelijkheden
- Ondergrenzen

De inventaris-balans koppelt de waarden van productie, inventaris en backlog variabelen. De verkoop-balans geeft de relatie tussen inventaris, productie en verkoop. De capaciteitsconstraint geeft de maximale capaciteit van de fabrieken voor de verschillende technologieën. De sourcing constraint geeft aan welke producten waar geproduceerd kunnen worden. De laatste constraint geeft een ondergrens aan de variabelen, die geen van allen kleiner dan nul kunnen zijn. De doelfunctie wordt gegeven als de som van kosten van productie, inventaris en backlog, min de som van opbrengsten uit verkoop. De waarde van deze doelfunctie moet geminimaliseerd worden.

Alle benodigde data is aangeleverd door NXP, maar in veel verschillende indelingen. Een standaard opmaak wordt voorgesteld, en alle data wordt omgewerkt naar die opmaak. Met de gegeven data wordt een implementatie van het model gecreëerd in Matlab. Als solver wordt LPsolve gebruikt. Na het omwerken van de vergelijkingen naar de door de solver gewenste opmaak kan het probleem opgelost worden. Analyse van de uitkomsten geeft aan dat het model zich correct gedraagt. De gegeven oplossing voldoet bovendien goed aan de verwachtingen.

Hiermee is aangetoond dat de methode van Kempf ook bij NXP toegepast kan worden. Wel kunnen de nodige aanbevelingen gedaan worden ter verbetering van het gebruikte model. Ook kan het project op meerdere wijzen vervolgd en uitgebreid worden.

Contents

Chapter 1

Introduction

NXP is one of the top 10 semiconductor manufacturers in the world. It is a multinational company, with its head office in Eindhoven, the Netherlands. Manufacturing operations at NXP are separated into a front end and a back end part. The front end of the production process deals with IC fabrication, sort and wafer cutting. The back end consists of assembly, testing and finish of the chip. For front end operations, multiple production facilities are used (around 25, of which 9 internally), and almost 400 different production routings in about 30 separate production technologies. Front end has to deal with highly uncertain demand and long lead times.

Capacity planning deals with the allocation of capacity to specific products. In other words, it decides which products are fabricated at which facilities, and in what amounts. Usually, there are limits on the capacity of different facilities, and not all facilities may be able to produce all products. The capacity planning problem for the front end of production at NXP is a challenging one, mainly because of its scale. At present, this problem is tackled through manual analysis by experienced planners.

Linear programming is a mathematical method to optimize linear or linearized problems. A linear program (LP) consists of a linear objective function, and any number of linear constraints. Both objective and constraints are dependent on the design variables of the problem. The goal of an LP is to find the minimum or maximum value (depending on the problem at hand) of the objective function by varying the design variables, while satisfying all constraint equations. Because of the linearity of the problem, even a very large LP can be easily solved by computer. A large amount of LP solvers, using different algorithms, is available. Because of the easy solving of even very large problems, linear programming is a popular method in optimization.

In his article 'Managing Supply-Demand Networks in Semiconductor Manufacturing', K. Kempf describes an approach for 'the integration of optimal decision making and controlling decision execution in the manufacturing core of a supply-demand-network.' This is a way of optimizing the entire manufacturing process. Hereto, he splits the problem into a strategic planning function and a tactical execution function. The strategic

planning function deals with long term planning by means of a linear programming formulation. The tactical execution function manages daily operations through model predictive control. The two functions are combined with an inventory planning function regulating inventory levels in the process. Used together, the three functions would achieve efficient operation of the network.

Kempfs approach could also be used at NXP. The different functions can be implemented and used independently. The strategic planning function might be of use for the industrial planning department at head office. The long term planning of Kempf corresponds to the capacity planning at NXP. The planning function might provide an improvement to present methods, or at least a useful tool in optimizing current method capacity plans. The tactical execution function could be of help in the facilities' management of operations. The inventory planning function poses a method to optimize the material flows through the manufacturing process.

The objective of this project is to investigate the possibilities of Kempfs approach for NXP. Of main interest is the application of the strategic planning function to the front end of production. This means that the front end capacity planning has to be modeled as a linear programming problem. As the entire capacity planning for all facilities and all products is a very large problem, a test case is provided. The test case contains five technologies in five production facilities. In this project, the capacity planning for the test case is created through linear programming. Implicit in this process are the creation of a linear program model and the implementation of that model in an existing solver.

The layout of this report is as follows: Chapter 2 describes the company NXP, focussing on the business planning operations. Chapter 3 goes into more detail regarding the capacity planning at NXP. In Chapter 4, a summary is given of the relevant section of the article by Kempf: a description of the LP he proposes to use in his strategic planning function [2]. A linear program formulation for the NXP test case is proposed in Chapter 5. The constraints are constructed and the LP objective function is derived. To be able to use the data provided by NXP in the proposed LP, quite a lot of data formatting is needed. This is the subject of Chapter 6. In Chapter 7, the LP and the data are combined in an implementation of the problem in a specific LP solver. Analysis of the LP output is done in Chapter 8. The final chapter, Chapter 9, handles the analysis of the project as a whole. Conclusions and recommendations for improvement and further research are in this chapter.

Chapter 2

NXP

In this chapter, the company NXP is introduced. The first section focuses on the business structure, and the location where this research is done. In Section 2.2, the production process is explained.

2.1 The company

2.1.1 Four sectors

NXP focuses on four sectors in semiconductor demand. For each of these sectors, they have a business unit (BU). These sectors/BUs are Mobile & Personal, Home, Automotive & Identification and Multimarket Semiconductors. The BUs operate independently from each other on the market, but they have shared resources and shared policies within NXP.

NXP defines four shared processes in its process model: Strategy & Business development, Innovation & Technology, Manufacturing & Supply chain and Sales & Marketing. These processes work together over all four business units to keep the company running. The process model is shown in Figure 2.1.

Strategy & Business development deals with the strategic, long-term plans for the business. It tries to assess the future needs of the market, and the general direction the market will take, and anticipates to that. Sales & Marketing then tries to determine, with the strategy in mind and together with (future) customers, what the customers' needs and desires are, and how those fit NXP. A global idea is developed of what specific products should be able to do, what their functionality should be. This global idea is then translated into actual product at Innovation & Technology. Product design and production process design are taken care of here. Next, the product has to be taken into production at Manufacturing & Supply chain. Manufacturing & Supply chain handles all the physical sides of the total process. It runs factory operations and takes care of

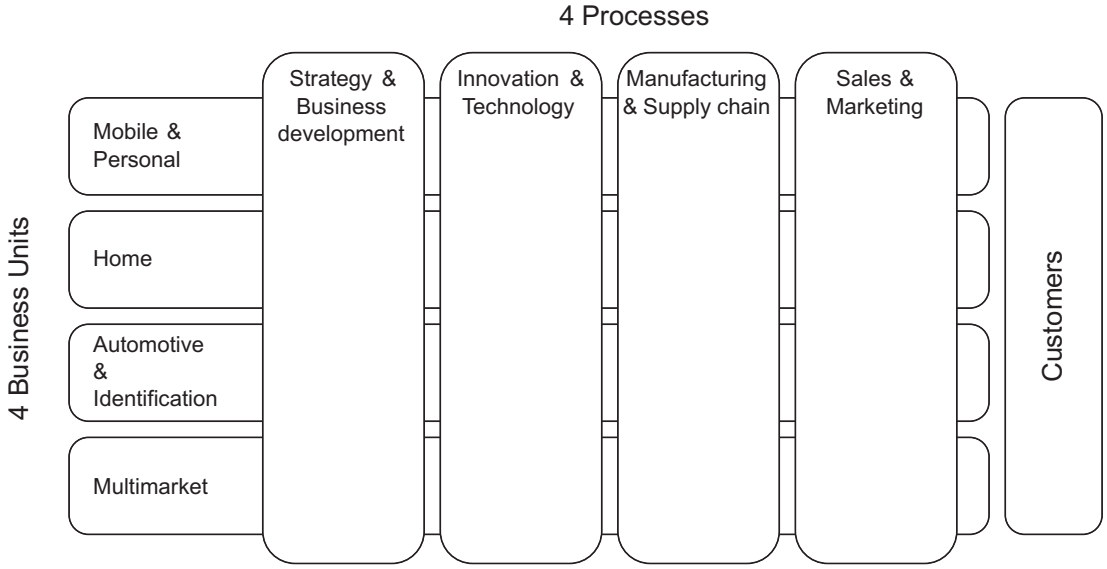

Figure 2.1: NXPs process model

actual production and distribution. Finally, the paperwork of distribution, the sales and the contact with the customers is done by Sales & Marketing.

2.1.2 Supply chain management

In this report, the focus will be on Manufacturing & Supply chain, more specifically on Supply chain management. In short, Supply chain management matches demand and capacity. This large task has been split into three parts: demand planning, supply planning and order fulfillment. Demand planning deals with forecasting demand on different timescales. On the long term, it translates the business strategy and market expectations into a production direction and investment plan. The medium term demand planning gives a demand forecast for specific products, up to 18 months ahead. The short term planning uses customer forecasts, the order book and work in progress or WIP levels to give a precise prediction of demand over the next 6 months.

The supply planning uses the outputs of the demand planning as inputs. On the long term, demand plan and business strategy give an industrial strategy, and a more precise investment plan. On the medium term, the mid term demand plan is combined with the fab capacity statements to yield a constrained capacity plan, containing a sourcing and a transfer plan. The short term supply plan takes the Short Term Plan (STP) of demand, the WIP and again the capacity statements to determine factory loading plans, material and personnel requirements and a financial budget. Order fulfillment works only on the short term. It checks the supply and demand STPs against actual orders and customer forecast, and updates and extends the plans from STP with a purchase plan and a distribution plan.

2.1.3 Mid Term Business Planning

Focussing again within Supply chain management, a closer look is taken at the mid term supply planning. This is the process directly concerned in this report. As shortly explained before, the planning forms a combination of the mid term demand planning and the factory capacity statements. How that is done is explained in this section.

The Mid Term Business Planning (MTBP) is handled by the Industrial Planning department (IP) at NXP head office. They cooperate with the forecasters of the business units, and with the management of the separate production facilities. The most important outcome of MTBP is a factory loading plan. This plan specifies how much of what product is made in which facility. This is done for six 3-month periods into the future. The plan is updated every 3 months.

So how is this plan made in practice? The BUs have access to previous plans. In the last plan, for each of the BUs it is specified where which part of their demands has been produced. Each business unit has its own demand forecast for the next six quarters. This new forecast is quite literally substituted in the old planning, without regard for capacity constraints. These new, unconstrained plans are sent to IP. From the other side, the fabs' capacity statements are sent to IP as well. IP will first take a good look at both of them, checking for logic and sanity in the plans. When they are convinced the plans are realistic, a match between the two has to be made. The match is made by hand. Factory loads are tried to be distributed evenly over all fabs. Also, as little as possible must be changed with respect to the former plan, since the fabs are adapted to that product mix. However, some transfers are inevitable, and demand may be too large to fit capacity. In that case, some of it has to be ignored. Which parts will be transferred or ignored is again a matter of common sense and logic.

When the first draft of the new planning is finished, it is sent both to fabs and BUs. Management at the fabs then has to decide whether or not they can meet the new planning, and what changes would be necessary in machinery and personnel. Some adaptions can be made, others cannot. The fab management returns a statement to IP saying if the planning is feasible for them or not, and what they would be able to do. The BUs also give feedback, accepting the new plan or asking for adaptation. With the updated statements, another run is done by IP. In this way, by a lot of communication and via an iterative process, the new planning is finalized.

2.2 The manufacturing process

At NXP, the manufacturing of semiconductors starts with smooth wafers. The wafers are produced externally, and stored in a raw materials inventory. From there, they enter the production process. The process is divided into the front end and the back end. The front end consists of diffusion and wafer test, the back end comprises assembly and final test. The process is depicted schematically in Figure 2.2.

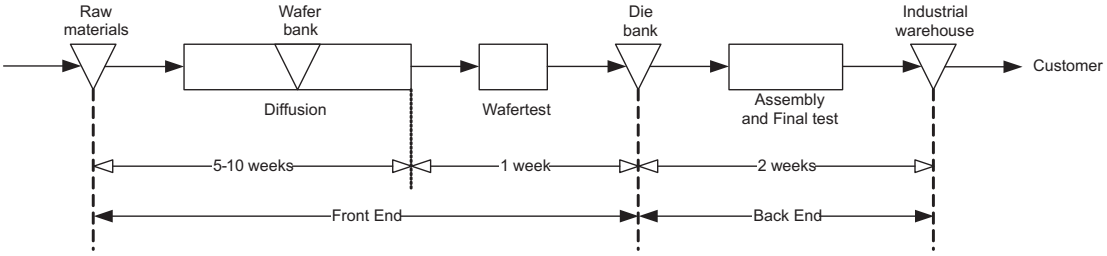

Figure 2.2: Schematic production process

In diffusion, via different processes, the electrical components are formed on the wafer. These processes include for example etching, lithography and vapor deposition. Layers of different materials are deposited onto and/or removed from the wafer. About halfway through diffusion, there is an inventory called the waferbank. After this waferbank, the second step of diffusion consists of the interconnection of the transistor layers with metal contacts. The whole diffusion process typically takes 5 to 10 weeks and can include up to 300 steps.

After diffusion, the wafers are tested. Because of the high complexity and small scale of the operations, it is inevitable that a (small) part of the dies on the wafer fail. They are marked, and later scrapped. After testing, the wafer is cut. The good, unmarked dies go to the die bank, another inventory. Testing and cutting takes roughly one week.

The die bank forms the separation between front end and back end. The dies from the die bank enter the assembly process. Here, the dies are placed in packages, die contacts are connected to external contact pins on the package, and the packages are sealed. The packages are then subjected to a final test process, where again, the correct behavior of the product is tested. Assembly and test consist of around 30 steps, and take some two weeks to complete. The complete and tested packages go to an industrial warehouse, and are then transported to the customers. The warehouse forms the end of the production chain for NXP. The complete chain thus consists of up to 350 production steps, and takes two to three months to complete.

Because of the long lead time, production cannot be completely made-to-order. Because of the uncertain demand and, for some products, fast devaluation, made-to-stock is also not a good option. A compromise is found by putting the order decoupling point at the die bank. The back end thus produces to order, while the front end works with demand forecasts.

2.3 Concluding

In this chapter, two things were done. First, the business structure of NXP was explained. Four BUs and four shared processes were introduced, and then the focus was changed first to Supply Chain Management, and later to the MTBP. Secondly, the

semiconductor production process was explained. The next chapter provides more specific information about the NXP supply chain, that is needed in solving the capacity planning problem posed by the MTBP. The present way of solving that problem has been given in Section 2.1.3. Chapter 4 introduces a different approach.

Chapter 3

The capacity planning problem

After a general introduction of the company NXP in Chapter 2, and a focus on Supply Chain Management and the Mid Term Business Plan (MTBP), this chapter provides more specifics of the company with respect to the capacity planning problem. Attention is restricted to front end from here onwards. Within the front end, two views can be taken on the production process. The first is the product view, handled in Section 3.1. It deals with the structure in the total product range. The second view is the fab view, subject of Section 3.2, looking at the capacity planning from the facilities perspective.

3.1 The product view

There are about 400 different *dietypes* in production in the front end. A dietype can be seen as the end product in front end. Only at the back end one dietype can be further differentiated into several products. Dietypes can be classified in several ways. At the top level, each type belongs to a *main capacity (maincap)*, or *technology*. This indicates a general production method for the product. NXP has about 30 different technologies. At the factories, or fabs, the fab capacities are expressed in these maincaps. Below maincap level, there are several *diffusion groups*, or *process groups* per maincap. This is mainly a logistical categorization, and it will not be used in this report. The next step down is *process* level. The process associated with a dietype indicates the exact production sequence and material needs for a dietype. Within one process, different dietypes can be produced by using different masks. The differentiation of products in front end is depicted in Figure 3.1.

The important part of costs of production for this project is the variable cost. The variable cost for a dietype can be determined using the number of masksteps needed in producing that dietype, and the cost of a maskstep. The cost of a maskstep is fab-dependant, so the variable cost of production depends both on dietype and fab.

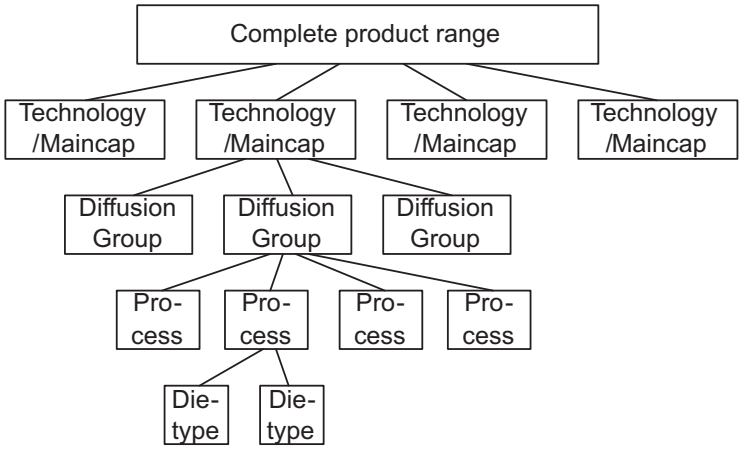

Figure 3.1: product differentiation

3.2 The fab view

NXP utilizes about 25 fabs for the front end. Of these, 9 are internal, the rest are external or shared facilities. Typically, in one fab, several maincaps can be manufactured. As mentioned, the capacity of a fab is given per maincap. Since a different maincap uses different technology, these capacities are generally not interchangeable. Thus, one fab has several capacities. In some cases, interchanging capacities is possible. The fabs, maincaps and exchange rates are known for all those cases.

Within the capacity for a certain maincap, different products can be interchanged. However, not all products in that maincap can be produced in all fabs. A fab has to be qualified to produce a certain dietype. Thus, it can only produce the dietypes it has been qualified for. New products can be introduced into a fab, but a qualification procedure has to be run. For a new dietype within an existing technology, this may take 6 to 9 months. For an entire new technology to be transferred may take up to 18 months.

When a new product is introduced, it is first qualified in one fab. This fab is called the first or main fab. When its production volume grows too large for the one fab, it has to be introduced into other fabs as well. At NXP, this dual sourcing is standard for products in the volume top 50. Dual sourcing decreases risks (if for some reason a fab goes down, production can be shifted, and is not completely lost), it increases flexibility and it increases capacity for that particular dietype.

3.3 Concluding

In Chapter 2, NXP was introduced as a company. Within NXPs business structure, the chapter focused on Supply Chain Management and the MTBP. This chapter has

provided more specifics of the NXP supply chain in two different views. Where Section 2.1.3 has given an overview of the present way of dealing with the MTBP, the next chapter introduces an alternative approach to solve the capacity planning problem in MTBP.

Chapter 4

A different approach to capacity planning

In the previous chapter, one way of solving the capacity planning problem has been presented: The way in which NXPs industrial planning department handles it. In this chapter, another approach is given. In the book 'Networks of interacting machines', Karl Kempf has written a chapter about supply chain management: Managing Supply-Demand Networks in Semiconductor Manufacturing [2]. After an introduction into the semiconductor industry and the problems involved, he focuses on manufacturing. The complete planning of all manufacturing operations is seen by him as an optimization problem, a 'continuous nonlinear stochastic combinatorial financial optimization'.

In order to solve the problem, he proposes to split it into a strategic planning function and a tactical execution function. The first function is to deal with the long-term general planning, the second function is for managing day-to-day operations. The strategic planning, a combinatorial financial optimization, is addressed through a linear program formulation. The tactical execution problem is solved by model predictive control. By combined use of the two functions, efficient operation of the network would be achieved.

The strategic planning function Kempf proposes could be of use for NXPs industrial planning department. It poses a different way of planning for the MTBP. Kempf explains his ideas about the strategic planning in more detail in section 3.1 of his article. Since that section is very relevant to this research, it is summarized below.

4.1 Kempf: A Capacity Planning Formulation, Summary

The capacity planning problem is seen as a mathematical optimization to allocate capacity to satisfy demand, while minimizing costs and maximizing revenue. Three major categories of inputs are distinguished:

1. Basic structure specification: Required material, possible manufacturing flows, *forecast* future values of performance parameters (mean cycle times, mean yields) and financial data (cost of manufacturing, transport & inventory, average selling prices). Note that all parameters can vary in time, per product and per factory.
2. Supply scenario: Capacity forecast for all facilities (product- and facilityspecific, time-varying), current WIP, current inventories and (if present) supply preferences.
3. Demand scenario: Demand forecasts, safety stock targets and (if present) demand preferences.

The optimization is presented as a linear program. The mathematical formulation can be found in his article, the textual formulation is included below. First, there are four constraints:

- Capacity constraint: Production cannot exceed capacity.
- Inventory mass balance: The inventory at time t equals the inventory at $t-1$ minus what goes out during that period plus what comes in.
- Backlog mass balance: Backlog at time t equals backlog at $t-1$ plus the difference between demand and actual sales.
- Inventory target constraint: A penalty function is given for deviation from the target inventory levels. This is a balance function for that deviation.

The objective function is given as a maximization of five factors:

$$\max \left\{ \begin{array}{l} + \text{ sales revenue} \\ - \text{ backlog cost} \\ - \text{ production cost} \\ - \text{ inventory cost} \\ - \text{ inventory target deviation cost} \end{array} \right\} \tag{4.1}$$

The primary results of execution of this mathematical optimization are: A material release plan (or product output plan) for all facilities, for all time periods in the horizon; a transportation plan; an inventory plan (including levels w.r.t. to targets); a demand satisfaction plan (including backlog); and a profit projection. Five ways of improving the performance by manipulating input parameters are stated:

- Selected demand can be moved earlier.
- Specific backlog can be authorized.
- Demand can be ignored.

- Inventory targets can be adjusted.
- Priorities can be modified.

The primary shortcoming of this mathematical optimization formulation is its treatment of the operational dynamics of the system. Firstly, stochasticity is disregarded. This problem is addressed partly in the tactical execution formulation, and partly by the inventory planning. Secondly, cycle times are both input and output: Fixed cycle times are used as input in the linear program. In the output, material releases are specified, resulting in a certain factory load. The factory loads may in turn influence the cycle times, that were initially fixed. For a feasible plan, output and input have to match. Several approaches are given in [2] to deal with that problem via iterative methods.

4.2 Concluding

In Chapter 3, NXPs method of capacity planning has been laid out. In the previous section, a different approach to capacity planning is given. Kempf goes into great detail in his formulation of the linear program. Setting aside the level of complexity he states, the linear program formulation would be useful for NXP. In the next chapter, a less complex formulation will be presented, that has been adapted for use in the NXP MTBP.

Chapter 5

Proposed linear program formulation for NXP

In the previous chapter, a linear program (LP) has been formulated to address the capacity planning problem in semiconductor industry. The level of complexity of the proposed LP is too high for it to be applied easily in practice. A much simpler, but also more practical formulation is presented in this chapter. In Section 5.1, the constraints are introduced. Section 5.2 deals with the objective function. A list of parameters is right below.

Input Parameters

F	$\in \mathbb{N}$	Number of factories
M	$\in \mathbb{N}$	Number of maincaps
P	$\in \mathbb{N}$	Number of products/Die Types
K	$\in \mathbb{N}$	Time horizon
$D_p(k)$	$\in \mathbb{R}+$	Demand for product p at time k $p \in \{1, 2, \ldots, P\}, \quad k \in \{1, 2, \ldots, K\}$
c^I, c^B	$\in \mathbb{R}+$	Unit cost of storage, backlog $p \in \{1, 2, \ldots, P\}$
s^S_{fp}	$\in \mathbb{R}+$	Unit revenue of sale of product p from fab f $p \in \{1, 2, \ldots, P\}, \quad k \in \{1, 2, \ldots, K\}$
c^a_f	$\in \mathbb{R}+$	Cost per maskstep in fab f [euro] $f \in \{1, 2, \ldots, F\}$
a_p	$\in \mathbb{N}$	Number of masksteps in production of product p $p \in \{1, 2, \ldots, P\}$

Input Parameters (continued)

I_{fp0}	$\in \mathbb{N}$	Inventory for product p and fab f at $k = 0$ $p \in \{1, 2, \ldots, P\}$ Assumption: $I_{fp0} = 0 \quad \forall f, p$
$W_f^m(k)$	$\in \mathbb{R}+$	Capacity of fab f for maincap m in k^{th} period $f \in \{1, 2, \ldots, F\}, \quad m \in \{1, 2, \ldots, M\}, \quad k \in \{1, 2, \ldots, K\}$
r_f^{mn}	$\in \mathbb{R}$	Rate of exchange of capacity for maincaps m and n in fab f $f \in \{1, 2, \ldots, F\}, \quad m, n \in \{1, 2, \ldots, M\}$ If no exchange possible, then $r_f^{mn} = r_f^{nm} = 0$, $r_f^{mm} = 1, \quad \forall r_f^{mn} \neq 0 : r_f^{mn} r_f^{nm} = 1$
$l_{fp}(k)$	$\in \mathbb{R}+$	Fixed lowerbound on production of p in fab f in k^{th} period
$\mathcal{P}_f(k)$		Set of all products that can be produced in fab f in the k^{th} period $f \in \{1, 2, \ldots, F\}, \quad k \in \{1, 2, \ldots, K\}$
$\mathcal{P}_m$		Set of all products in maincap m $m \in \{1, 2, \ldots, M\}$
$\mathcal{F}^e$		Set of all external fabs
$\mathcal{P}^e$		Set of all products produced in an external fab and at least one other fab

Variables

$x_{fp}(k)$	$\in \mathbb{R}+$	Production of product p in fab f in k^{th} period $f \in \{1, 2, \ldots, F\}, \quad p \in \{1, 2, \ldots, P\}, \quad k \in \{1, 2, \ldots, K\}$
$I_{fp}(k)$	$\in \mathbb{R}+$	Inventory for product p in fab f at time k $f \in \{1, 2, \ldots, F\}, \quad p \in \{1, 2, \ldots, P\}, \quad k \in \{1, 2, \ldots, K\}$
$B_p(k)$	$\in \mathbb{R}+$	Lost demand for product p in period k $p \in \{1, 2, \ldots, P\}, \quad k \in \{1, 2, \ldots, K\}$
$S_{fp}(k)$	$\in \mathbb{R}+$	Sales amounts for product p from fab f at time k $f \in \{1, 2, \ldots, F\}, \quad p \in \{1, 2, \ldots, P\}, \quad k \in \{1, 2, \ldots, K\}$

5.1 Constraints

The constraints to the proposed LP are introduced in this section. There are five: two equality constraints and three inequality constraints. The equality constraints are balance equations, while the inequalities give the physical boundaries of the system. They are treated one by one below.

5.1.1 Inventory balance constraint

The inventory can be seen as a separate system within the production system. In inventory, no products are created or lost. Thus, mass conservation has to apply. The inventory levels in time have to balance with the in- and outflows of products. The

inflow of products into inventory comes from production. All products will pass through inventory. The outflow of products is determined by demand. The mass conservation equation thus becomes: Inventory at time k equals the inventory at the previous time instant, $k-1$, plus production in that period, minus demand at time k.

Complicating things, inventory cannot be negative, that would be physically impossible. To account for that fact, an extra term has to be introduced: a backlog term. Backlog is that part of demand that cannot be filled from production or inventory. Logically, there is backlog only if the inventory is empty, otherwise the backlog could be (partially) filled. That assumption is not dealt with here, though, but taken care of by the objective function, in the next section.

Next, the inventory mentioned before actually consists of more than one. Even though two fabs produce the same product, the sales price for it may differ per fab. Therefore, inventory has to be kept separate per fab. For the mass balance, these different inventories of the same product can be summed. Production also has to be summed over all facilities. Thus, for any product p and any time k, the mass balance equation becomes:

$$\sum_{f=1}^{F} I_{fp}(k) - B_p(k) = \sum_{f=1}^{F} \big(I_{fp}(k-1) + x_{fp}(k)\big) - D_p(k) \qquad \forall p, k. \tag{5.1}$$

This equation holds for all products and all times. However, the special case $k = 1$ needs an extra condition, an initial condition for inventory at $k = 0$, because of the appearance of $I_{fp}(k-1)$ in the equation. This initial condition is identified as I_{fp0}, and it is assumed zero in all cases:

$$I_{fp}(0) = I_{fp0} = 0 \qquad \forall f, p. \tag{5.2}$$

5.1.2 Sales balance constraint

The total outflow of the inventory system is not necessarily equal to demand. In (5.1), the difference between outflow and demand is balanced by the backlog term. The actual amount of products that can exit the system is different. Products exit inventory when they are sold, either internally to back end or to a customer. The number of products to be sold in reality equals the demand for that product minus that part of demand that cannot be filled, the backlog.

Unfortunately, demand and backlog are not fab-dependent, but sales are. Therefore, we have to look at it from another side. Equation 5.1 allows us to equate demand and backlog with inventory and production, terms that are fab-specific. For every fab, inventory mass balance has to apply, so sales, or outflow, has to equal the change in inventory plus production (inflow). This adds up to (5.3), that has to hold for all fabs, all products, and all times.

$$S_{fp}(k) = I_{fp}(k-1) - I_{fp}(k) + x_{fp}(k) \qquad \forall f, p, k. \tag{5.3}$$

Combining (5.1) and (5.3), we can indeed see that total sales have to equal the difference between demand and backlog:

$$\sum_{f=1}^{F} S_{fp}(k) = D_p(k) - B_p(k) \qquad \forall p, k. \tag{5.4}$$

for all products and all time periods.

5.1.3 Sourcing constraint

Not all products can be produced in all fabs. NXP provides sourcing data that specifies what products can be made at what facilities. If a product cannot be produced in a certain fab, its production there has to be set to zero. This is done by a combination of two constraints. The lowerbound on production will be treated further on, the upperbound of zero is given here. The upperbound states that production cannot be larger than zero. Theoretically, it could still be smaller than zero. All products that can be produced in fab f in period k are in set $\mathcal{P}_f(k)$. When, for all products not in that set, production $x_{fp}(k)$ has to be smaller than or equal to zero, the sum of production for all these products has to be smaller than or equal to zero as well. This is expressed in (5.5):

$$\sum_{p \notin \mathcal{P}_f(k)} x_{fp}(k) \leq 0 \qquad \forall f, k. \tag{5.5}$$

This equation is valid for all fabs and all time periods.

5.1.4 Capacity constraint

The amount of products to be manufactured is, obviously, constrained by the capacity of the factories. Production cannot be larger than total capacity. Fab capacities are given by NXP, specified per maincap. Thus, every maincap has its own capacity in a certain fab. Within maincaps, capacity can be spread over all products. One product takes one unit of capacity, whatever the product.

Between maincaps, some capacity exchange is also possible. The exchange rate, however, is not necessarily one. It is specified for two maincaps m and n as r_f^{mn}. The subscript f indicates that there is a fab dependency. The exchange rate of maincap m with itself, r_f^{mm}, is 1. If no exchange is possible between m and n in fab f, r_f^{mn} is set to zero.

Automatically, r_f^{nm} is also zero. When r_f^{mn} is non-zero, the inverse exchange rate r_f^{nm} has to be the inverse of r_f^{mn}, i.e. $r_f^{mn} r_f^{nm} = 1$.

For a maincap m without exchangeability in fab f, summed production in f of all products in m cannot be larger than the capacity stated for m. Mathematically, for such a maincap, r_f^{mn} is zero for all n, except for $n = m$, where $r_f^{mm} = 1$. When the capacity for m is defined to be $W_f^m(k)$, the constraint thus becomes (solely for maincap m):

$$\sum_{p \in \mathcal{P}_m} x_{fp}(k) \leq W_f^m(k), \tag{5.6}$$

where $\sum_{p \in \mathcal{P}_m} x_{fp}(k)$ indicates the sum of production of all products in maincap m.

For maincaps with exchangeability, things are a bit more complex. Production of one maincap could be larger than its capacity, but at the cost of capacity for another maincap. For two maincaps m and n in fab f, the equation becomes:

$$r_f^{mm} \sum_{p \in \{\mathcal{P}_m\}} x_{fp}(k) + r_f^{mn} \sum_{p \in \{\mathcal{P}_n\}} x_{fp}(k) \leq r_f^{mm} W_f^m(k) + r_f^{mn} W_f^n(k). \tag{5.7}$$

The generalized version of this equation for more maincaps is:

$$\sum_{n=1}^{M} \left(r_f^{mn} \sum_{p \in \{\mathcal{P}_n\}} x_{fp}(k) \right) \leq \sum_{n=1}^{M} r_f^{mn} W_f^n(k) \qquad \forall f, m, k. \tag{5.8}$$

This equation can be seen to be valid for all cases, also the case where no exchange is possible, by substituting the relevant values of r_f^{mn}. Thus, (5.8) is valid for all maincaps, all fabs and all times.

5.1.5 Lowerbounds

The final constraint deals with the physical impossibility of variables being negative. Production has to be zero or positive, and the same goes for inventory and backlog. The lowerbounds are expressed in (5.9), (5.10) and (5.11). Sales is positive by definition, so there is no need to define a lowerbound for sales.

$$x_{fp}(k) \geq 0 \qquad \forall f, p, k, \tag{5.9}$$

$$I_{fp}(k) \geq 0 \qquad \forall f, p, k, \tag{5.10}$$

$$B_p(k) \geq 0 \qquad \forall p, k. \tag{5.11}$$

Some additional lowerbounds also have to be defined. For some products in external fabs, a minimum production quantity is desired by NXP. This may be because of contractual obligations, or simply to keep a process running in that fab. This extra lowerbound will take the form of (5.9), but with a positive number in the place of the zero, see (5.12). This equation is valid for all k, but only for certain f and p.

A set of external fabs has been defined as $\mathcal{F}^e$, a subset of the set of all fabs. Within this subset of fabs, the lowerbound is not for all products. Another set has to be defined, the set of all products that are manufactured in an external fab and at least one other fab, $\mathcal{P}^e$. Products produced only in on external fab do not need a lowerbound, because production takes place in that fab anyway. Setting a lowerbound anyway would not harm the model, but here it has not been done. For fabs in $\mathcal{F}^e$ and products in $\mathcal{P}^e$, the equation is valid:

$$x_{fp}(k) \geq l_{fp}(k) \quad \forall f \in \mathcal{F}^e, \forall p \in \mathcal{P}^e, \forall k. \tag{5.12}$$

With the lowerbounds, the constraints are fully determined. Two balance equations have been constructed to strictly define inventory levels, backlog and sales numbers. The upper boundary of production has been defined in the capacity constraint, and the sourcing constraint ensures that no product is produced where this is not possible. The lowerbound of zero and the additional positive lowerbound complete the definition of the feasible domain. In the next section, the objective function is treated. This function determines the direction of search within the feasible domain, and the solution to the problem.

5.2 Objective

Now that the constraints have been defined, an objective has to be constructed. The LP is set up as a cost minimization, the program will try to find the lowest possible total cost point in the feasible domain. This means a cost has to be assigned to the value of all variables. The summed cost over all variables then has to be minimized.

There are four kinds of variables in the system: production, sales, inventory and backlog variables. All variables are linked through the balance equations (5.1) and (5.3). This means that by fixing one of them, the entire system state will be determined. The model is based on the main cost factor in the system, production. Thus, a minimum cost will be determined by varying the production numbers.

With four kinds of variables, there will also be four cost factors. The two large contributors to total cost are sales and production. Backlog and inventory cost are a lot smaller. The four cost factors are treated one by one in the remainder of this section.

5.2.1 Minimizing costs

The cost of production can be calculated from NXP data. Every product has a number of production steps, masksteps. The number of steps depends on the product. The cost of a maskstep is dependant on the facility. Production cost thus depends both on product and fab. In reality, there is a time dependency as well, but this is not taken into account in this model. The unit cost of production c^x_{fp} becomes, with c^a_f denoting the cost of a maskstep and a_p being the number of masksteps, $c^x_{fp} = c^a_f a_p$. To obtain the total cost for a product p in fab f over period k, the unit cost has to be multiplied by the amount produced, so that $c_{fp}(k) = c^x_{fp} x_{fp}(k)$. This number then has to be summed over all products, all fabs and all periods to get the total cost of production. The total production cost J^x is given by:

$$J^x = \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{f=1}^{F} c^a_f a_p \cdot x_{fp}(k). \tag{5.13}$$

The second large cost factor mentioned is sales. However, sales in general means revenue, not cost. By adding a minus sign to the equation, sales are converted to a negative cost. In this way, sales can be accounted for in the minimization problem. Sales prices for this problem can be found again in NXP data. The prices depend on both product and fab. Multiplication of unit sales price and amount of sold products and summation over time, product and fabs give total sales cost J^s:

$$J^s = -\sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{f=1}^{F} s^S_{fp} \cdot S_{fp}(k). \tag{5.14}$$

For the two remaining factors, no direct link exists to NXP data. The cost of inventory is negligible compared to the cost of production, and for backlog, or missed production, no data is available. The main cost there can be found in a reduction of sales. The main reason a cost is associated with the two, is a mathematical one. Without an associated cost, there would be no need to minimize either inventory or backlog. A situation could arise where both inventory and backlog are positive. In that case, backlog could be (partially) filled from inventory, but there is no incentive to do so. The system would be undetermined. With a small cost for both inventory and backlog, the situation where either one is zero will yield the lowest cost, fixing the system state. The cost of inventory is not dependent on product, fab or time, as long as it is much smaller than the lowest production cost or sales price. The same goes for backlog with respect to time and product. The equations for total inventory and backlog cost are (5.15) and (5.16), respectively.

$$J^I = c^I \cdot \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{f=1}^{F} I_{fp}(k), \tag{5.15}$$

$$J^B = c^B \cdot \sum_{k=1}^{K} \sum_{p=1}^{P} B_{fp}(k). \tag{5.16}$$

With all four cost factors defined, the total objective function is simply the sum of its parts.

$$J = J^x + J^s + J^I + J^B. \tag{5.17}$$

The objective of the LP then is to minimize the value of the objective function J through variation of x. Filling in (5.13)–(5.16) in (5.17), and rearranging, this can be stated as:

$$\min_x J = \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{f=1}^{F} \left(c_f^a a_p \cdot x_{fp}(k) - s_{fp}^S \cdot S_{fp}(k)\right) + c^I \cdot \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{f=1}^{F} I_{fp}(k) + c^B \cdot \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{f=1}^{F} B_p(k). \tag{5.18}$$

5.2.2 Concluding

The definition of the objective of the LP completes the definition of the LP itself. In the first section of this chapter the constraints have been defined, setting the boundaries of the feasible domain. The objective function of this section defines the point (or points) we are looking for within that domain.

In Chapter 4 the idea of using linear programming for optimizing the capacity planning was first stated. The LP proposed in that chapter was too complex to be implemented in the time set for this research. In this chapter, an LP has been defined that is more practical and better fits the needs of NXPs planning department. Solving this LP is an actual possibility. Two important things have to be done for solving. First, the data needed has be to be gathered. Where needed, its format has be changed, or other pre-treatment has to take place. This is the topic of Chapter 6. The other thing that needs to be done is the implementation of the LP in a mathematical solver. That subject is treated in Chapter 7.

Chapter 6

Data

In the previous chapter, an LP formulation has been presented. It was stated that this particular LP can be solved with NXP data. To come to this solution, two conditions have to be met. The first condition is the availability of the necessary data in a useful format. The second condition is the implementation of the LP in a mathematical solver. Chapter 7 deals with the implementation. This chapter is about data. First, in Section 6.1, the information needed is listed. The next section handles data formatting. The different datasets needed come in different formats. A standard format is proposed for better insight in the information and easy use in implementation. In Section 6.3, it is explained how the data has been obtained and transformed to the proposed standard format. The chapter finishes with some concluding remarks in Section 6.4.

6.1 Data identification

This section deals with the data needed for solving the capacity planning problem. These data are the inputs into the LP. All inputs have been mentioned implicitly in the last chapter, but here they are stated explicitly. In order to be able to find the right data, one has to know exactly what it is. The following data (in no particular order) are needed from NXP:

- The number of products, maincaps, fabs and periods (P, M, F and K). Also, the distribution of products over maincaps needs to be known. In practice, these data are not stated explicitly, but they will follow from the size of other data sets.
- Demand data: The forecast demand for all products, for all periods ($D_p(k)$).
- Fab capacities per maincap for all periods ($W_f^m(k)$).
- Sourcing data: Data identifying the possible production locations for each product in every period.

- Sales prices for all products and all fabs (s^S_{fp}).
- Maskstep cost: The cost per maskstep for all fabs (c^a_f).
- The number of masksteps for each product (a_p).
- Exchange ratios for all maincaps and all fabs (r^{mn}_f).
- Set of external fabs ($\mathcal{F}^e$).
- Lowerbounds for external fabs $l_{fp}(k)$.
- Set of products produced in an external fab and at least one other fab ($\mathcal{P}^e$).

NXP has provided all data above for a small subproblem in the total capacity planning, a case study. NXP uses Microsoft Excel for all its data management. The files that have been provided and the data contained are listed in Table 6.1. The files can all be found on the CD accompanying this report. Data not mentioned in Table 6.1 is not to be found in the files. It has been obtained via E.J.J. van Campen, employed at NXP and coach of this project.

Table 6.1: NXP data files and data contained

File name	**Data**
'NXP MTBP dec06 demand analysis.xls'	P, M, F and K
	$D_p(k)$
	$W^m_f(k)$
'NXP Sourcing.xls'	Sourcing data
	$\mathcal{P}^e$
'NXP Internal Wafer Prices.xls'	s^S_{fp}
'NXP Variable cost per mask.xls'	c^a_f
'NXP Masksteps Mar07.xls'	a_p

Unfortunately, within Excel, different formats are used for different datasets. Also, most of the data is not in a format that a mathematical program can easily work with. That is not necessary for all data, but for datasets that have to be loaded into the computer program, a standard format would be useful.

6.2 Proposal for a standard format

In the last section, the information needed has been identified. Also, it has been established that a standard data format would be useful in working with that information. In this section, one such standard format is proposed.

All data considered here consists of two parts: The data itself, e.g. a capacity statement of 17k wafers, and additional information stating what the data is about, saying that the 17k wafers capacity is for maincap C50PMU in fab ICF in the first quarter of 2008. Maincap, fab and period are indices of capacity. The indices correspond to the definition of capacity in Chapter 5, with $W_f^m(k)$ showing the same indices. There are many ways to present the data in matrix form, putting the indices on the x and y axis in different ways, see for example Table 6.2.

Table 6.2: Example of data presentation

		quarter			
maincap	fab	08Q1	08Q2	08Q3	...
C50b	ICF				
C50b	SSMC		data		
C50PMU	ICF				
C50PMU	SSMC				
...					

It is proposed here to put all data in a dataset in a single column, and to put all indices in columns in front of the data column. There are four indices in total, period, maincap, product and fab. They are included in the data matrix in that same order. If one or more of the indices is not used for a certain datatype, those columns are omitted. In the case of capacity, the product index is not used. Table 6.2 shows an example of the proposed format.

Table 6.3: Column format

quarter	maincap	fab	data
08Q1	C50b	ICF	
08Q1	C50b	SSMC	
08Q1	C50PMU	ICF	
08Q1	C50PMU	SSMC	
08Q2	C50b	ICF	
08Q2	C50b	SSMC	
08Q2	C50PMU	ICF	
08Q2	C50PMU	SSMC	
08Q3	C50b	ICF	
...			

The new format allows easy sorting to any desired index, or to the data itself. In Excel, the function PivotTable still allows for different views of the data, such as in Table 6.2. For the data to be used in a mathematical program, another change has to be

made. The program would have difficulties understanding indexes such as SSMC or C50b. Therefore, all indexes are replaced by numbers. In Chapter 5, M, P, F and K have been defined as the number of maincaps, products, fabs and periods, respectively. Thus, maincaps are numbered one to M, products one to P, etc. A legend has to be kept to be able to identify the numbers with the actual names.

A possible problem in this approach is in the number of indexes. The index SSMC can be identified as a fab, but the index 2 could just as well indicate a product or a period. This problem is prevented by the fixed order of the indexes mentioned before. Thus, a 2 in the second column of Table 6.2 can only identify a maincap. An additional advantage of this situation is, that as long as the nature of the data is specified in the file name, the column headers are no longer necessary. The index names can be derived from the data. As an example, Table 6.2 shows some capacity data. We know that capacity has the indexes maincap, fab and period. They have to be in the order period, maincap, fab, and the last column gives the actual capacity data. Thus, we can derive that the third line of the table gives a capacity of 10 for maincap 1 in fab 1 in the third quarter. The legend allows us to find the names to the numbers. Sorting of the rows does not affect the data.

Table 6.4: Standard format capacity data

1	1	2	12
2	2	2	0
3	1	1	10
3	1	2	15
...			

The data format in Table 6.2 contains no text strings any more. It has been demonstrated that this does not affect the information contained. It does allow for easy reading by Matlab. This format is set as standard for the remainder of this report. Do note that even in standard format, a single table contains a single set of data. For different datasets, different tables are needed.

6.3 Data gathering

Now that a standard format has been set, it can be applied to the NXP data. Unfortunately, the translation is not always as straightforward as posed in Section 6.2. All datasets and their treatment will be handled in this section. It should be noted that in most cases the data concerns forecasts, not actual values. Also, this section does not deal with any physical data, only the data handling is explained. All the actual data is on the CD attached in the back of this report. An overview of the CD contents is in Appendix A.

6.3.1 Demand data, $D_p(k)$

The Excel file 'NXP MTBP dec06 demand analysis.xls' contains a lot of information. Both demand and capacity data can be found in this file. Also, the number of products, maincaps, fabs and periods can be derived from it. As mentioned before, a product is considered to be a dietype. Besides all the information needed for the LP, the file also contains a lot of other information not needed. A typical line of data is shown in Figure 6.1.

DieType	Order	BL name	BU name	DieType before /-	Cluster	Fab	MainCap	Process Group	Process	Demand Status
MC5230	1	CS	MP	MC5230 (CS) Q	8"	PSF	QUBIC	QUBIC4	FQ45LMIMHV	0. C

MTBP	2007Q1STP	2007Q2STP	2007Q1	2007Q2	2007Q3	2007Q4	2008Q1	2008Q2	12NC desc (produced)
2006m12	0.1105	0.113	0.167	0.219	0.135	0.124	0.154	0.137	MC5230P9A

Plant (resource)	2007-M02	2007-M03	2007-M04	2007-M05	2007-M06	2007-M07	BU/BL	BL
ICF	23	45	23	45	45	68	MP	MP-CS

Figure 6.1: A typical line of data from 'MTBP dec06 demand analysis'

It can be seen in the figure that only about a third of the data is of use for the LP. The columns needed are 'DieType', 'Fab', 'Maincap' and the six columns '2007Q1'–'2008Q2'. Indirectly, the column 'process' is also needed. The other columns can be ignored or deleted. The file contains three data worksheets for five maincaps. The planning runs from the first quarter of 2007 to the second of 2008, so over six periods of a quarter. Five different fabs can be found.

The appearance of fabs in demand data needs some special attention. The desired demand data is not fab-specific, and in fact, it does not matter for the customer at which fab a product is made. In order to avoid large changes of production locations from one planning to the next, demand is divided over the fabs internally. Every new planning initially uses the same distribution over fabs as the one before.

The demand cannot simply be summed over all fabs to get total demand for a product. For its capacity planning, NXP is mainly interested in large-volume products. The large volumes largely determine the production distribution over the fabs. Focus is therefore on the products that form the top 80% of demand volume. Since demand can vary over different maincaps, every maincap is considered separately.

For products in the top 80%, demand can be summed over all fabs to get total demand. For the remaining 20%, a different approach is used. These products are assigned to the fab where the demand is specified. One product sourced in multiple fabs is treated as as many products as fabs, each with its own demand. All demand of the assigned small volume products in a fab can then be summed. The resulting demand is treated as demand for a single product, that is assigned to that fab. The new product is named

to indicate fab and maincap concerned. Production of the product is assigned to the fab concerned via the sourcing data. Sales price and number of masksteps are averaged over all products in the sum. Again, this process is used for every maincap separately. The sum and the products contained are added to the legend.

Summation of small volume products decreases the total number of products used for the LP. The resulting amount of products is 54, divided over five maincaps. For the 54 products, the demand data can be converted to the standard format. The resulting data sheet, with the names of maincaps and dietypes added, is sheet '7. dem' in the Excel file 'test case.xls', on the attached CD. It is the first of four input files for Matlab.

6.3.2 Sourcing data

Sourcing data is given in the file 'NXP Sourcing.xls'. This file contains a list of dietypes and fabs where those dietypes are produced or will be produced. In a column 'Release date', several statements can be made to give further information. A few lines of sourcing data are shown in Figure 6.2. The statements '1st' and 'now' indicate that production of the product is possible. If a quarter is mentioned there, that is the quarter when production of that dietype can be started in that fab. The statement 'hold' or the absence of a product-dietype combination in the list indicate that no production is possible.

BU	BL	Plant (resource)	FAB	Maincap	Basic resource	Diffusion Process	Nickname	Die Type	Actual DS	Release date	Comment
A&I	R63	ID	SSMC	0.18um	CMOS14FFLV	ZCMOS14FFLV		5CD040/14	Planned	1st	1st
HOME	65	DE80	ICB	BIMOS	BIMOS5	DBIMOS5	SHERPA	N100591	yes	now	actual
MP	MP-CS	TW88_FABE	UMC	0.25um	C50	C050FM	Jose3	OM6384-3A	Y	07Q3	
MP	MP-CIPT	US71	ICF	QUBIC	QUBIC4	QUBIC4+	Vega1	RC481	Y	1st	

Figure 6.2: A few lines of data from 'Sourcing'

The desired format of the sourcing data is a list stating whether production of a product is or is not possible in a fab in a period. It is chosen to use the numbers 0 and 1 to indicate that production of a dietype is not or is possible, respectively. The statements 1st and now thus can be replaced by a 1 for all periods and hold will yield a 0 for all periods, just like the absence of a combination in the list. The statement of a quarter indicates that production is not possible before that period, giving a 0, and is possible starting at that quarter, yielding a 1 from that quarter onwards.

The sourcing data has been changed into the standard format for legibility, but it has been implemented manually. Automated use of the sourcing data would reduce the amount of programming in regular use.

6.3.3 Capacity data, $W_f^n(k)$

The capacity data is contained in the same file as the demand data: 'NXP MTBP dec06 demand analysis.xls'. A data line for capacity can be recognized by the statement

'capa' in the columns 'DieType' and 'Demand Status' in Figure 6.1. The capacity data is extracted from the file manually, and copied into a new sheet. Capacity is given per maincap, per fab. However, not every maincap can be produced in every fab. The available data only concerns existing combinations of maincaps and fabs. Capacity for maincaps that cannot be produced in a fab is, logically, zero. That zero capacity has to be added to the data manually.

One zero capacity could be added for each maincap that cannot be produced, so that the dataset includes all maincap-fab combinations. However, when two maincaps both have zero capacity in a fab, their combined capacity is also zero. Therefore, a single zero capacity suffices for all maincaps that cannot be produced in a fab. In fact, when this one capacity is added, it might also include all other products that cannot be produced in that fab, i.e. those products in maincaps that have capacity, that are not sourced in the fab concerned. In this way, the sourcing constraint and the capacity constraint are combined in their implementation. The implementation of the constraints is treated in more detail in the next chapter, in Section 7.2.2.

With the extra maincap, every fab has as many capacities as there are maincaps in that fab, plus one. This one extra, indicating the maincaps and products that are not sourced in that fab, is zero by definition. The extra capacity is given a maincap index number zero in the standard format. The maincaps and products contained in maincap 0 may differ per fab. The exact contents of maincap 0 can be derived by fab number ad sourcing data. The resulting data in standard format is in sheet '9. cap' in the Excel file 'test case.xls'. It is the second of the input files for Matlab.

In a different way of modeling, every maincap in each fab might be assigned their own (zero) capacity. This would indicate an extension of the dataset, and a separation of the capacity and sourcing constraints. While the present approach is more compact, it is also more complex. For clarity, the alternative approach might be the better one.

6.3.4 Sales prices, s^S_{fp}

The sales price forecasts can be found in the file 'NXP Internal Wafer Prices.xls'. The internal sales prices are needed for all products and all fabs. They are assumed constant in time. However, they are not in fact constant. Also, they are not given per product, but per process. The first problem is easily solved: The average value of the price over the six quarters concerned is taken as time-independent value.

The second problem is to find a direct link between process and product. One process may produce several different dietypes, and one dietype may have more than one associated process. Fortunately, a link exists between processes and fabs. One dietype generally has as many associated processes as production locations. The process name is different because the fab is different, but the process itself is the same. The correlation of dietypes and processes is extracted from the demand data in 'NXP MTBP dec06 demand analysis.xls'. In this research, the coupling of process and product-fab

has been done manually. Automation of this process would significantly speed up the data handling.

In matching the different sets of data, sales prices and process-product coupling information, the available data was found to be incomplete. Not all processes needed could be found in the sales data. In those cases, data has been used from a similar process (indicated by identical fab and similar process name). This approach is justified by the largely identical prices for similar processes for as far as they are available.

The standard format sales data are in sheet '11. price' of the file 'test case.xls', as the third Matlab input file.

6.3.5 Maskstep cost, c_f^a

The maskstep cost, or cost per maskstep, is fab-dependent. Like sales, it is assumed constant in time. It is taken to be the variable cost per maskstep from the file 'NXP Variable cost per mask.xls'. This file contains the quarterly cost per maskstep for the internal fabs. The 2006 numbers are actual costs, the 2007 numbers are forecast values. 2008 data were not yet available at the start of this research. Therefore, the year forecast for 2007 is used.

For the external fabs, no cost information is available. However, it is known that NXP Front End does not make a profit on wafers produced externally. Thus, the total wafer cost should be close to the wafer sales price. Since the number for masksteps for a given product is known, the cost per maskstep can be calculated. This is done for the product with the lowest price in the fab concerned. The resulting number is used as maskstep cost estimate for that fab.

Because of the small size of this dataset, it has not been transformed to an input file to be loaded in Matlab. Instead, it has been manually copied. The data is in sheet '13. maskcost' of the file 'test case.xls'.

6.3.6 Number of masksteps, a_p

Like the sale prices, the number of masksteps is specified per process, not per product. The same coupling is used to obtain the number of masksteps per product. Unfortunately, the same problem appears as well: incomplete data. Not all processes needed are mentioned in the maskstep data file ('NXP Masksteps Mar07.xls'). For as far as data was available, it was found that the number of masksteps was identical for all products in a maincap. The missing data has been filled in according to that finding. Excel sheet '14. maskst' in file 'test case.xls' forms the fourth and last input file for Matlab.

6.3.7 Exchange ratios, r_f^{mn}

It has been stated that there are not many cases of interchangeability between maincap capacities. For this reason, an exchange ratio has not been defined for all fabs or maincap combinations. This parameter has not been formalized in any dataset. The existing ratios have been manually implemented into the constraint equations in Matlab. The first is a $1 : 1.6$ ratio between two maincaps, 'C50PMU' and 'C50b', that is valid for all three fabs where these maincaps are produced. The second is a $1 : 1$ exchange ratio between C50PMU and a third maincap, 'QUBIC4', valid only in a single fab. With the first-mentioned ratio, this also leads to a $1 : 1.6$ ratio for QUBIC4 and C50b in that fab. All other exchange rates are zero.

As mentioned, the exchangeability data has not been formalized and it has been implemented manually, like the sourcing data. In a situation where the LP would be regularly used, it might prove more efficient to apply the standard format. This would result in a reduction of manual labor in the constraint formulation in Matlab.

6.3.8 External fabs

Three things remain to complete the dataset needed for the LP: The set of external fabs, the lowerbounds for external fabs and the set of products produced in an external fab and at least one other fab. The set of external fabs is not formalized, but it is stated that two of the five fabs are external facilities, and also which two. The set of products produced externally and at least in one other fab follows from the sourcing data. This set is time-dependent, so it may change from period to period. In total, it contains 7 products.

The lowerbound on production for the products in the last-mentioned set is set the same for all products in the set, all fabs and all periods. A certain minimum production is desired to keep the production process running. As an indication, one waferstart per week or per two weeks would suffice. With 13 weeks in a quarter, one start per two weeks, and a start containing 25 wafers, that comes to 162.5 wafers per quarter. As a rounded number, 200 is used, or 0.2k wafer.

6.4 Concluding

In Chapter 5, a linear program was specified to solve the capacity planning problem for NXP. This chapter started with a list of all the data needed in order to solve the LP. Next, a standard format was proposed for the data, to obtain a better overview and to facilitate implementation. In Section 6.3, the data needed has been matched with the data supplied. In the same process, it was largely transformed into the standard format.

Four input files have been created for Matlab. Together with a fifth dataset that was manually copied, they contain all data which use has been automated in implementation. Other datasets have been manually implemented. The input files are repeated in Table 6.4.

Table 6.5: Input files for implementation

Datasheet	**Description**
'7. dem'	A PF by 4 matrix containing demand data. The first three columns contain period, maincap and dietype index, respectively.
'9. cap'	A 4-column matrix containing capacity data. The first three columns contain period, maincap and fab index, respectively.
'11. price'	A PF by 4 matrix containing sales prices. The first three columns contain maincap, dietype and fab index, respectively.
'13. maskst'	A P by 3 matrix containing the number of masksteps per product. The first two columns contain maincap and dietype index, respectively.
'14. maskcost'	(manually copied) An F by 2 matrix containing the maskstep cost. The first column contains the fab index

With all data available in a clear and standardized manner, implementation of the LP in Matlab is the second condition for actually solving the LP. Implementation is the topic of the next chapter. When both conditions are fulfilled, the LP can be solved. Output will be generated and the results can be analysed. The analysis of results is done in Chapter 8.

Chapter 7

Implementation

In Chapter 5, an LP was proposed to help solve NXP's capacity planning problem. The proposed LP can be solved with the case study data provided by NXP. The data has been sorted and formatted in Chapter 6. The next step is implementation of the LP in a mathematical program for the actual solving. Implementation is the subject of this chapter.

First, in Section 7.1, a choice is made for a certain solver. The reasons for this choice and its consequences are treated. The main consequence is the need for rearrangement of the equations in Chapter 5 to fit the input standard of the solver. That rearrangement is the subject of Section 7.2. With both equations and data in the right format, the problem can easily be converted into computer code. The conversion into code is not treated in the main text, but the code and comments on programming can be found on the CD accompanying this report.

At this point, a Matlab script exists containing the complete problem definition and the command to solve it. When the script is executed, the problem is solved: an optimal solution is returned as output. However, the output as returned by the solver is not in a readable format. Some postprocessing is needed. That is the topic of Section 7.3. After postprocessing, the solution is analyzed in Chapter 8.

7.1 LP Solver selection

For solving linear programming problems, many programs, packages and other tools are available. Commercial tools are often expensive, and for that reason not suitable for this project. Other available solvers are not always able to deal with the large number of variables, so they cannot be used either. Because of the available experience with the program and the sizeable amount of solvers usable within the program, Matlab was chosen as platform for modeling the LP.

Matlab has its own function for solving LP's: the function *linprog*. However, *linprog* does not always give the right results, and for large problems, calculation time becomes large as well. Fortunately, Matlab has a number of toolboxes and add-ons that allow it to work with other solvers as well. The first solver that was tried was *cddmex*, an LP-solver from the MultiParametric Toolbox for Matlab [3]. This solver did provide good results for small subproblems, but for a large problem, calculation time also became excessively large. The second solver tried, Cplex, provided good results with an excellent calculation time. Unfortunately, access to Cplex was rather difficult, so its practical use was limited.

LPsolve [1] is a free mixed integer linear program solver (also capable of solving LP's), originally developed at the TU/e. The solver and documentation can be downloaded from the internet (http://lpsolve.sourceforge.net/). LPsolve works with many different programming languages and programs, one of which is Matlab. Implementation and use of the solver in Matlab are straightforward, and calculation time is good. LPsolve has been chosen to work with in this project, with Matlab as platform.

LPsolve can also be called from other platforms than Matlab alone. In fact, it can also be implemented in Microsoft Excel. NXP uses Excel for its data management, so this forms an additional advantage of this solver. While in this project the solver has been used from Matlab, implementation in Excel is recommended if the LP is to be used at NXP in practice.

Implementation of the LP requires rewriting of the equations in Chapter 5. LPsolve uses a standard matrix formulation, and the inputs need to fit the required format. The basic formulation used is in (7.1)–(7.4):

$$\max_y J = f'y \tag{7.1}$$

$$\text{subject to:} \quad A_i y \leq b_i \tag{7.2}$$

$$A_e y = b_e \tag{7.3}$$

$$v_{lb} \leq y. \tag{7.4}$$

The objective function is stated in (7.1), while (7.2)–(7.4) represent the inequality equations, equality equations and lowerbounds, respectively. In total, the system has six input arguments: f, A_i, b_i, A_e, b_e and v_{lb}. The vector of variables is denoted by y.

f is a vector of coefficients for a linear objective function.

A_i is a matrix representing the inequality constraints.

A_e is a matrix representing the equality constraints.

b_i is a vector of right sides for the inequality constraint equations.

b_e is a vector of right sides for the equality constraint equations.

v_{lb} is a vector of lowerbounds for all variables.

LPsolve uses five input arguments to define the system above: The matrices A_i and A_e, and the vectors b_i and b_e are combined, resulting in a single matrix A and a single vector b. An extra vector has to be defined to indicate the sign for each equation, i.e. '=' or '$\leq$'. This vector is defined for LPsolve as e.

- $e(i) = -1$ means that $A(i)y \leq b(i)$
- $e(i) = 0$ means that $A(i)y = b(i)$

With the five input arguments, LPsolve is called in Matlab as:

```
[obj,y] = lp_solve(f,A,b,e,vlb),
```

resulting, if a solution is found, in two output arguments, *obj* and y. Of these outputs, y is the optimal value of the variable vector, and *obj* gives the optimal objective function value. Both for input and output, more (optional) arguments can be called in LPsolve. These extra arguments have not been used, and are therefore omitted.

As mentioned, the constraint equations (5.1), (5.3), (5.5), (5.8) and (5.9)–(5.12) and the objective function (5.18) have to be transformed to the format above. This transformation is the subject of the next section.

7.2 Equation rearrangement

7.2.1 Variable vector and objective function

First, the two equations for the objective function J are matched. On the one hand, in Chapter 5, the objective function was defined in (5.18), which can be rewritten as:

$$\begin{aligned} J &= \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{f=1}^{F} c_f^a a_p \cdot x_{fp}(k) \\ &+ c^I \cdot \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{f=1}^{F} I_{fp}(k) \\ &+ c^B \cdot \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{f=1}^{F} B_p(k) \\ &- \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{f=1}^{F} s_{fp}^S \cdot S_{fp}(k). \end{aligned} \tag{7.5}$$

On the other hand, as shown in (7.1), LPsolve requires that $J = f'y$. To match the two equations, or indeed the two equation systems, first variable vector y has to be defined. The variables in (7.5) are $x_{fp}(k)$, $I_{fp}(k)$, $B_p(k)$ and $S_{fp}(k)$. All variables have to be placed in a single vector y in a logical order. The first selection has already been made implicitly by noting four different kinds of variables. They will be put in the vector y in the same order as they are in (7.5), so that:

$$y = \begin{pmatrix} y^x \\ y^I \\ y^B \\ y^S \end{pmatrix}. \tag{7.6}$$

Each of the sub-vectors contains of a number of elements. For y^x, y^I and y^S, this number equals FPK, since x, I and S are specified dependent on F, P and K. Vector y^B, depending only on P and K, contains PK elements. An order has to be found within the subvectors. In Section 6.2, a fixed order was proposed for data indexing. This proposition set the index order period, maincap, product, fab. The variables in each of the sub-vectors of y are sorted according to that order: first by period, then by product and lastly by fab. In this sorting, the index numbers from the standard format are used, not the index names. An indication of the result is given for y^x in (7.7):

$$y^x = [\quad x_{11}(1) \ldots x_{F1}(1) \quad x_{12}(1) \ldots x_{F2}(1) \quad \ldots \quad x_{FP}(1) \quad x_{11}(2) \quad \ldots \quad x_{FP}(K) \quad]'. \tag{7.7}$$

With this definition, there is a one-to-one relation between the variables $x_{fp}(k)$, $I_{fp}(k)$, $B_p(k)$ and $S_{fp}(k)$, and the elements in y. For each variable, its place in y can be determined. This relation can be expressed as:

$$x_{fp}(k) = y_{FP(k-1)+F(p-1)+f} \tag{7.8}$$
$$I_{fp}(k) = y_{FPK+FP(k-1)+F(p-1)+f} \tag{7.9}$$
$$B_p(k) = y_{2FPK+P(k-1)+p} \tag{7.10}$$
$$S_{fp}(k) = y_{2FPK+PK+FP(k-1)+F(p-1)+f} \tag{7.11}$$

With the variable vector y defined, the vector of coefficients, f can be constructed in the same manner. For every variable in y, there is one coefficient in f. The coefficient corresponds to the cost factor for that variable. The factors simply have to be put in the same order to obtain f. At the highest level, again, there are four kinds of factors, corresponding to four kinds of variables. One level below that, the variables are sorted by period. Since all cost factors were assumed time-independent, the coefficient vector for one period can simply be repeated K times to obtain the total vector. For example, the last part of f, representing the sales prices s^S_{fp} and denoted f^S, looks like this:

$$f^S = [\quad \underbrace{s^S_{11} \dots s^S_{FP}}_{\text{period 1}} \quad \underbrace{s^S_{11} \dots s^S_{FP}}_{\text{period 2}} \quad \dots \quad \underbrace{s^S_{11} \dots s^S_{FP}}_{\text{period K}} \quad]'. \tag{7.12}$$

Within periods, the same order can be kept as for variables by sorting the factors to the same indexes, first product, then fab. For inventory and backlog, the cost factor is the same for all variables, resulting in (in total) FPK and PK repetitions of c^I and c^B, respectively.

7.2.2 Constraint equations

Now that variable vector y has been defined, the constraint equations (5.1), (5.3), (5.5) and (5.8) can also be converted to matrix format. The equations are repeated below as (7.13)–(7.18). They have been rewritten so that all variable terms are on one side of the equation and all fixed terms on the other. For (5.1) and (5.3), this means a separation of the equation for two cases: The term $I_{fp}(k-1)$ in the two equations can be on both sides. In the first period, for $k-1=0$, $I_{fp}(0)$ is data. It is assumed zero for all f and p (see the input parameters in Chapter 5). For all other periods, $I_{fp}(k-1)$ represents a design variable.

$$\sum_{f=1}^{F} \big(x_{fp}(k) - I_{fp}(k)\big) + B_p(k) = D_p(k) \qquad \forall p, k = 1, \tag{7.13}$$

$$\sum_{f=1}^{F} \big(x_{fp}(k) + I_{fp}(k-1) - I_{fp}(k)\big) + B_p(k) = D_p(k) \qquad \forall p, k > 1, \tag{7.14}$$

$$x_{fp}(k) - I_{fp}(k) - S_{fp}(k) = 0 \qquad \forall f, p, k = 1, \tag{7.15}$$

$$x_{fp}(k) + I_{fp}(k-1) - I_{fp}(k) - S_{fp}(k) = 0 \qquad \forall f, p, k > 1, \tag{7.16}$$

$$\sum_{p \notin \mathcal{P}_f(k)} x_{fp}(k) \le 0 \qquad \forall f, k, \tag{7.17}$$

$$\sum_{n=1}^{M} \left(r_f^{mn} \sum_{p \in \mathcal{P}_n} x_{fp}(k) \right) \le \sum_{n=1}^{M} r_f^{mn} W_f^n(k) \qquad \forall f, m, k. \tag{7.18}$$

Each constraint represents a number of equations. For example, the inventory balance (7.13) for $k=1$ is valid for all p, so it represents P equations, while (7.14) represents $P(K-1)$ equations. In matrix format, each equation is specified separately. This specification is done below for each of the constraints.

Inventory balance constraint

The inventory balance constraints (7.13) and (7.14) are treated together. The difference between the two is made clear where necessary. The total constraint consists of PK equations, one for each combination of p and k. This system of equations can be written in vector notation as:

$$b_i^x + b_i^I + b_i^B + b_i^S = b^D, \quad \text{where:} \tag{7.19}$$

$$b_i^x = \begin{pmatrix} \sum_F x_{f1}(1) \\ \sum_F x_{f2}(1) \\ \vdots \\ \sum_F x_{fP}(1) \\ \sum_F x_{f1}(2) \\ \vdots \\ \sum_F x_{fP}(K) \end{pmatrix}, \quad b_i^B = \begin{pmatrix} B_1(1) \\ B_2(1) \\ \vdots \\ B_P(1) \\ B_1(2) \\ \vdots \\ B_P(K) \end{pmatrix}, \quad b_i^S = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad b^D = \begin{pmatrix} D_1(1) \\ D_2(1) \\ \vdots \\ D_P(1) \\ D_1(2) \\ \vdots \\ D_P(K) \end{pmatrix}, \tag{7.20}$$

$$\text{and} \quad b_i^I = \begin{pmatrix} \frac{b_{i,k=1}^I}{b_{i,k=2}^I} \\ \vdots \\ b_{i,k=K}^I \end{pmatrix} = \begin{pmatrix} & - & \sum_F I_{f1}(1) \\ & - & \sum_F I_{f2}(1) \\ & \vdots & \\ & - & \sum_F I_{fP}(1) \\ \hline \sum_F I_{f1}(1) & - & \sum_F I_{f1}(2) \\ \sum_F I_{f2}(1) & - & \sum_F I_{f2}(2) \\ & \vdots & \\ \sum_F I_{fP}(1) & - & \sum_F I_{fP}(2) \\ \sum_F I_{f1}(2) & - & \sum_F I_{f1}(3)) \\ & \vdots & \\ \sum_F I_{fP}(K-1) & - & \sum_F I_{fP}(K) \end{pmatrix}. \tag{7.21}$$

The subscript i indicates the constraint concerned: the inventory balance. In b_i^I, the distinction is made between the first P equations where $k = 1$ and no term $I_{fp}(k-1)$ is present, and the rest of the equations, where $k > 1$ and that term is present. The vector b_i^S is included so that all elements in y are taken into account in the equation.

Next, each vector on the left hand side of (7.20) can be written as the product of a matrix A and a part of variable vector y, so that:

$$A_i^x y^x = b_i^x \quad A_i^I y^I = b_i^I \quad A_i^B y^B = b_i^B \quad A_i^S y^S = b_i^S \tag{7.22}$$

$$A_i^x y^x + A_i^I y^I + A_i^B y^B + A_i^S y^S = b^D \tag{7.23}$$

The inproduct of the first row of a matrix A and the corresponding y gives the first element of b, the second row and y the second element, and so on. The rows of A thus contain the multiplication factors for the variables in y. From a known y and a known b, A can be constructed.

For vector b_i^S, the construction of A_i^S is easy. Since the sales variables are not in the inventory balance constraint, all multiplication factors are zero. Thus, A_i^S is a zero matrix with PK rows and FPK columns (as many columns as elements in y^S). Construction of A_i^B is also straightforward: b_i^B is equal to y^B, so A_i^B should be a $PK \cdot PK$ identity matrix.

Vector b_i^x contains sums of production. Writing out the sum for the first element, $b_i^x(1) = x_{11}(1) + x_{21}(1) + \ldots + x_{F1}(1)$, it can be seen that it contains the first F variables in y^x. This because of the variable order in y, sorted lastly to fab. Thus, the first F elements of the first row of A_i^x should be ones, the rest is zero. The second element of b_i^x contains the second F variables, so the second row of A_i^x should have the second F elements 1 and the rest zero. This structure is continued to fill the $FPK \cdot PK$ matrix A_i^x. The structure is shown in (7.24) for $F = 3$:

$$A_{\text{inv}}^x = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \\ & \vdots & & & \vdots & & & \vdots & & \end{pmatrix}. \tag{7.24}$$

Lastly, b_i^I has to be converted. It was given in (7.21) with a vertical separation between $b_{i,k=1}^I$ and $b_{i,k>1}^I$. For the construction of A_i^I, a different separation is more suitable:

$$b_i^I = \left(\begin{array}{rl|l} & - \sum_F I_{f1}(1) \\ & - \sum_F I_{f2}(1) \\ & \vdots \\ & - \sum_F I_{fP}(1) \\ \sum_F I_{f1}(1) & - \sum_F I_{f1}(2) \\ \sum_F I_{f2}(1) & - \sum_F I_{f2}(2) \\ & \vdots \\ \sum_F I_{fP}(1) & - \sum_F I_{fP}(2) \\ \sum_F I_{f1}(2) & - \sum_F I_{f1}(3)) \\ & \vdots \\ \sum_F I_{fP}(K-1) & - \sum_F I_{fP}(K) \end{array}\right) = \underbrace{\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \sum_F I_{f1}(1) \\ \sum_F I_{f2}(1) \\ \vdots \\ \sum_F I_{fP}(1) \\ \sum_F I_{f1}(2) \\ \vdots \\ \sum_F I_{fP}(K-1) \end{pmatrix}}_{b_i^{I1}} + \underbrace{\begin{pmatrix} -\sum_F I_{f1}(1) \\ -\sum_F I_{f2}(1) \\ \vdots \\ -\sum_F I_{fP}(1) \\ -\sum_F I_{f1}(2) \\ -\sum_F I_{f2}(2) \\ \vdots \\ -\sum_F I_{fP}(2) \\ -\sum_F I_{f1}(3) \\ \vdots \\ -\sum_F I_{fP}(K) \end{pmatrix}}_{b_i^{I2}} \tag{7.25}$$

For each of the resulting vectors, a matrix can be constructed. Adding the two matrices gives A_i^I. The second vector, b_i^{I2}, contains the elements $\sum_F I_{fp}(k)$, similar to the elements in b_i^x, but negative. The corresponding matrix A_i^{I2} is therefore equal to the negative of A_i^x.

The vector b_i^{I1} is very similar to the second, but positive, and all elements have moved down one period, or P places. The first P elements are zero, the last element is now not $\sum_F I_{fP}(K)$, but $\sum_F I_{fP}(K-1)$. The corresponding matrix A_i^{I1} can be formed also by taking the positive, moving all rows down P places and setting the first P rows to zero. With the two matrices constructed, the equation for b_i^I becomes:

$$\left(A_i^{I1} + A_i^{I2}\right) y^I = A_i^I y^I \quad = \quad \left(b_i^{I1} + b_i^{I2}\right) = b_i^I. \tag{7.26}$$

An example of A_i^I is given in (7.27) for $F = 3$ and $P = 2$:

$$A_i^I = \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \\ & \vdots & & & \vdots & & & \vdots & & & \vdots & & \end{pmatrix}. \tag{7.27}$$

Now that all matrices in (7.23) have been defined, a final step can be set. Just as the sub-vectors of y can be combined to form y, so can the four matrices be combined to form a single matrix A_i. With that step, and b^D given in (7.20), (7.13) and (7.14) are written in matrix notation as:

$$\begin{aligned} A_i^x y^x + A_i^I y^I + A_i^B y^B + A_i^S y^S &= \begin{pmatrix} A_i^x & A_i^I & A_i^B & A_i^S \end{pmatrix} \begin{pmatrix} y^x \\ y^I \\ y^B \\ y^S \end{pmatrix} \\ &= \begin{pmatrix} A_i^x & A_i^I & I & 0 \end{pmatrix} y \\ &= A_i y = b^D. \end{aligned} \tag{7.28}$$

Sales balance constraint

Like the inventory balance constraints, the sales balance equations (7.15) and (7.16) are treated as one. The sales balance is defined for all f, p and k, so it consists of FPK equations. Working along the same method as for the inventory balance, four matrices

can be constructed for the sales balance (indicated with a subscript s). The vector equation for (7.15)–(7.16) becomes:

$$b_s^x + b_s^I + b_s^B + b_s^S = b_s, \quad \text{with:} \tag{7.29}$$

$$b_s^x = \begin{pmatrix} x_{11}(1) \\ \vdots \\ x_{F1}(1) \\ x_{12}(1) \\ \vdots \\ x_{FP}(1) \\ x_{11}(2) \\ \vdots \\ x_{FP}(K) \end{pmatrix}, \quad b_s^B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad b_s^x = \begin{pmatrix} -S_{11}(1) \\ \vdots \\ -S_{F1}(1) \\ -S_{12}(1) \\ \vdots \\ -S_{FP}(1) \\ -S_{11}(2) \\ \vdots \\ -S_{FP}(K) \end{pmatrix}, \quad b_s = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

$$b_s^I = \begin{pmatrix} & - & I_{11}(1) \\ & \vdots & \\ & - & I_{F1}(1) \\ & - & I_{12}(1) \\ & \vdots & \\ & - & I_{FP}(1) \\ I_{11}(1) & - & I_{11}(2) \\ & \vdots & \\ I_{F1}(1) & - & I_{F1}(2) \\ I_{12}(1) & - & I_{12}(2) \\ & \vdots & \\ I_{FP}(K-1) & - & I_{FP}(K) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ I_{11}(1) \\ \vdots \\ I_{F1}(1) \\ I_{12}(1) \\ \vdots \\ I_{FP}(K-1) \end{pmatrix}}_{b_s^{I1}} + \underbrace{\begin{pmatrix} -I_{11}(1) \\ \vdots \\ -I_{F1}(1) \\ -I_{12}(1) \\ \vdots \\ -I_{FP}(1) \\ -I_{11}(2) \\ \vdots \\ -I_{F1}(2) \\ -I_{12}(2) \\ \vdots \\ -I_{FP}(K) \end{pmatrix}}_{b_s^{I2}}, \tag{7.30}$$

And the matrix equations for each vector become:

$$A_s^x y^x = b_s^x, \qquad A_s^I y^I = b_s^I, \qquad A_s^B y^B = b_s^B, \text{ and} \qquad A_s^S y^S = b_s^S. \tag{7.31}$$

Since the backlog variables are not in the sales balance, A_s^B will be a $FPK \cdot PK$ zero matrix. Vector b_s^x is equal to y^x, so A_s^x is an FPK-square identity matrix. With b_s^S being equal to $-y^S$, A_s^S becomes the negative of A_s^x, or a negative FPK-square identity matrix. For the vector b_s^I, the same separation can be used as for b_i^I in (7.25). In this case, the second vector b_s^{I2} is equal to $-y^I$, so that A_s^{I2} is equal to A_s^S: Another negative FPK-square identity matrix. The first vector, and matrix, can be constructed

in an similar fashion to b_i^{I2} and A_i^{I2}: By taking the positive, moving all elements down one period, in this case FP places, and setting the first period rows (again FP) zero. Adding both matrices gives A_s^I. The result is given in (7.32) for F, P and K equal to 2:

$$A_s^I = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}. \tag{7.32}$$

Filling in all matrices in vector equation (7.29) and rewriting, the final result is $A_s y = 0$:

$$\begin{aligned} A_s^x y^x + A_s^I y^I + A_s^B y^B + A_s^S y^S &= \begin{pmatrix} A_s^x & A_s^I & A_s^B & A_s^S \end{pmatrix} y \\ &= \begin{pmatrix} I & A_s^I & 0 & -I \end{pmatrix} y \\ &= A_s y = b_s = 0. \end{aligned} \tag{7.33}$$

Capacity and sourcing constraints

Although the capacity and sourcing constraints are two separate constraints, they are very similar. Both constraints give an upper bound on the production of a certain set of products. Because of the capacity data structure adopted in Section 6.3.3, the two constraints can be implemented as one. This data structure combined the sourcing and capacity data, by restricting the capacity for a certain maincap to include only the products that the fab concerned can produce. An extra maincap, maincap 0, was introduced for each fab to include all products that that fab cannot produce. Mathematically, this amounts to the definition of maincap 0 and the adaptation of the sets of products used in equations (7.17) and (7.18). The equations and relevant set definitions are repeated here:

$$\sum_{p \notin \mathcal{P}_f(k)} x_{fp}(k) \leq 0 \qquad \forall f, k, \tag{7.34}$$

$$\sum_{n=1}^{M} \left(r_f^{mn} \sum_{p \in \mathcal{P}_n} x_{fp}(k) \right) \leq \sum_{n=1}^{M} r_f^{mn} W_f^n(k) \qquad \forall f, m, k. \tag{7.35}$$

$\mathcal{P}_f(k)$	Set of all products that can be produced in fab f in the k^{th} period $f \in \{1, 2, \ldots, F\}, \quad k \in \{1, 2, \ldots, K\}$
$\mathcal{P}_m$	Set of all products in maincap m $m \in \{1, 2, \ldots, M\}$

The two set definitions can be combined to give: $\mathcal{P}_{mf}(k) = \mathcal{P}_m \cap \mathcal{P}_f(k)$, so $\mathcal{P}_{mf}(k)$ is the set of all products in maincap m, that can be produced in fab f in the k^{th} period. As long as (7.34) stands, the sum over $\mathcal{P}_n$ in (7.35) can be replaced by a sum over $\mathcal{P}_{nf}(k)$. The only products excluded from the equation in that way, were already excluded by (7.34):

$$\sum_{n=1}^{M} \left(r_f^{mn} \sum_{p \in \mathcal{P}_{nf}(k)} x_{fp}(k) \right) \leq \sum_{n=1}^{M} r_f^{mn} W_f^n(k) \qquad \forall f, m, k. \tag{7.36}$$

The collection of $\mathcal{P}_{mf}(k)$ over all maincaps m equals $\mathcal{P}_f(k)$ (all products in $\mathcal{P}_f(k)$ have to belong to a maincap): $\cup_{m=1}^{M} \mathcal{P}_{mf}(k) = \mathcal{P}_f(k)$. Therefore, $\cup_{m=1}^{M} \mathcal{P}_{mf}(k) \cup \mathcal{P} \backslash \mathcal{P}_f(k) = \mathcal{P}$. When maincap 0 is defined to contain $\mathcal{P} \backslash \mathcal{P}_f(k)$, a new maincap set can be defined as:

$$\mathcal{P}_{cf}(k) = \begin{cases} \mathcal{P} \backslash \mathcal{P}_f(k) & \text{for } c = 0 \\ \mathcal{P}_{mf}(k) & \text{for } c = m, \end{cases} \tag{7.37}$$

With $f \in \{1, 2, \ldots, F\}$, $k \in \{1, 2, \ldots, K\}$ and $c \in \{0, 1, 2, \ldots, M\}$. This set contains all products: $\mathcal{P}_{cf}(k) = \mathcal{P}$. With the new set definitions, (7.34) can be written as:

$$\sum_{p \in \mathcal{P}_{0f}(k)} x_{fp}(k) \leq 0 \qquad \forall f, k, \tag{7.38}$$

If, next, the exchange ratios r_f^{m0} are defined as $r_f^{m0} = 0$ for $m \neq 0$ and $r_f^{m0} = 1$ for $m = 0$ (no exchangeability) and the capacity $W_f^0(k)$ is set to $W_f^0(k) = 0$ (by definition, maincap 0 has zero capacity), this can be written:

$$\sum_{n=0}^{M} \left(r_f^{0n} \sum_{p \in \mathcal{P}_{nf}(k)} x_{fp}(k) \right) \leq \sum_{n=0}^{M} r_f^{0n} W_f^n(k) \qquad \forall f, k, c = 0. \tag{7.39}$$

This definition is identical to the sourcing constraint definition, and is in the format of the capacity constraint. The two constraints can be merged by extending the maincap set and maincap sum in (7.35) to include maincap 0, or extending m to c. The result is a single equation:

$$\sum_{n=0}^{M} \left(r_f^{cn} \sum_{p \in \mathcal{P}_{nf}(k)} x_{fp}(k) \right) \leq \sum_{n=0}^{M} r_f^{cn} W_f^n(k) \qquad \forall f, c, k. \tag{7.40}$$

This is the equation that is implemented, so this equation has to be rewritten in matrix format. The construction of matrix A_c and vector b_c (subscript c indicating the capacity constraint) is less straightforward than construction of the other constraint matrices. It takes place in several stages, and it starts with the capacity data vector defined in 6.3.3. This vector, b_c^1, contains all $W_f^c(k)$. The vector is sorted first by period, then by fab, then by maincap (starting at maincap 0). In b_c^1, no maincap exchangeability is taken into account yet. Exchangeability is incorporated in a later stage. The simplified equation without exchangeability is:

$$\sum_{p \in \mathcal{P}_{cf}(k)} x_{fp}(k) \leq W_f^c(k) \qquad \forall f, c, k. \tag{7.41}$$

The first step in the construction of constraint matrix A_c is the construction of a matrix A_c^1, that obeys $A_c^1 y \leq b_c^1$ and sets it equal to (7.41). Because only variable $x_{fp}(k)$ is present in both (7.40) and (7.41), the matrix equation may be rewritten $A_c^{x1} y^x \leq b_c^1$. The other submatrices A_c^{I1}, A_c^{B1} and A_c^{S1} contain only zeros. Now, since both b_c^1 and y^x are sorted first by period, the vectors can be written as:

$$b_c^{x1} = \begin{pmatrix} b_c^{x1}(1) \\ b_c^{x1}(2) \\ \vdots \\ b_c^{x1}(K) \end{pmatrix}, \text{ and } \quad y^x = \begin{pmatrix} y^x(1) \\ y^x(2) \\ \vdots \\ y^x(K) \end{pmatrix}, \tag{7.42}$$

and because the capacities and production amounts of different periods are completely independent, A_c^{x1} can also be rewritten:

$$A_c^{x1} = \begin{pmatrix} A_c^{x1}(1) & 0 & \cdots & 0 \\ 0 & A_c^{x1}(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_c^{x1}(K) \end{pmatrix}. \tag{7.43}$$

In this way, the matrix equation $A_c^1 y \leq b_c^1$ has been reduced to $A_c^{x1}(k) y^x(k) \leq b_c^{x1}(k), \forall k$. However, the matrices $A_c^{x1}(k)$ still have to be constructed for the reduced equation to reflect (7.41).

Each $A_c^{x1}(k)$ is formed from an $FP \cdot FP$ identity matrix. An identity matrix implies a separate constraint for each product in each fab. The identity matrix also says that the capacity statements are sorted first by product, then by fab, because that is the order in $y^x(k)$. The order of the rows can be changed so that they are sorted first by fab, then by product. In the resulting matrix, $A_c^{x2}(k)$, the first P rows represent the separate constraints of all products in fab 1, the second P of all products in fab 2, and so on up to fab F.

The adding of different rows of the $A_c^{x2}(k)$ combines the constraints associated with those rows. According to (7.41), all products in a maincap share one capacity constraint per fab. Thus, to obtain the constraint for maincap 0 in fab 1, all of the first P rows that correspond to products in maincap 0 in that fab, can be added to form a single constraint, a single row. The sourcing data provides information about which products are in which maincaps. When this is done for all maincaps in that fab, in increasing order, the first P rows of $A_c^{x2}(k)$ are collapsed into as many rows as there are maincaps in that fab. When again, this is done for all fabs from 1 to F, or for all F sets of P rows, $A_c^{x1}(k)$ is the resulting matrix. It should be noted that the number of rows per fab may differ, just as it does in $b_c^{x1}(k)$, because the number of maincaps in a fab is not the same for all fabs.

At this point, $A_c^{x1}(k)$, $b_c^{x1}(k)$ and therefore A_c^{x1} and b_c^{x1} are known, but maincap exchangeability has not been taken into account. Because the exchangeability is the same for all periods, and again periods are completely independent, exchangeability can be implemented in $A_c^{x1}(k)$ and $b_c^{x1}(k)$. Constraint (7.40) with exchangeability between maincaps states that the weighted sum of production of all exchangeable maincaps should be smaller than or equal to the same weighted sum of their capacities. The sums can be implemented in both $A_c^{x1}(k)$ and $b_c^{x1}(k)$ in a similar way the sums of products within a maincap were implemented, only with weights r_f^{cn}: By adding the weighted rows corresponding to the maincaps concerned. An example is given below for two maincaps with $r_f^{cn} = 2$. Only the relevant elements of $A_c^{x1}(k)$ and $b_c^{x1}(k)$ are given.

$$A_c^{x1}(k)y^x(k) = \begin{pmatrix} & & \vdots & & & \vdots & & \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ & & \vdots & & & \vdots & & \end{pmatrix} y^x(k) = \begin{pmatrix} \vdots \\ 10 \\ 6 \\ \vdots \end{pmatrix} = b_c^{x1}(k) \quad (7.44)$$

$$\Downarrow$$

$$A_c^{x}(k)y^x(k) = \begin{pmatrix} & & \vdots & & & \vdots & & \\ 1 & 0 & 2 & 1 & 0 & 2 & 0 & 0 \\ & & \vdots & & & \vdots & & \end{pmatrix} y^x(k) = \begin{pmatrix} \vdots \\ 22 \\ \vdots \end{pmatrix} = b_c^{x}(k). \quad (7.45)$$

Doing this for all $A_c^{x1}(k)$ and $b_c^{x1}(k)$, $A_c^{x}(k)$ and $b_c^{x}(k)$ and therefore A_c^{x} and b_c^{x} result.

Final matrix system and e-vector

With all constraint matrices and vectors known (lowerbounds have a separate definition in LPsolve, and are therefore treated separately at the end of this section), the entire matrix system can be constructed. The constraint matrix A, containing all constraints, can be formed by a combination of the separate constraint matrices. Vector b can be constructed in a similar way:

$$A = \begin{pmatrix} A_i \\ A_s \\ A_c \end{pmatrix} \qquad b = \begin{pmatrix} b_i \\ b_s \\ b_c \end{pmatrix}. \tag{7.46}$$

After the combination of both equality and inequality matrices in a single system, the relation between A, y and b is no longer clear. To identify the constraint sense, vector e has to be defined. For the inequalities, e should contain a -1, for the equalities a 0. The vector should have as many elements as there are equations, or as many as vector b. It can be separated in sections parallel to vector b as:

$$e = \begin{pmatrix} e_i \\ e_s \\ e_c \end{pmatrix}. \tag{7.47}$$

Each of the subvectors of e can be defined separately. The first subvector e_i, representing the sense of the inventory balance, should contain only zeroes, since the inventory balance is an equality constraint. There are PK equations in the constraint, so e_i is a column of PK zeroes. Vector e_s also contains zeroes, because the sales balance is an equality as well. The sales balance consists of FPK equations, so e_s is a column of FPK zeroes.

The last vector, e_c, has to contain elements -1. However, where the vector length of e_i and e_s is strictly defined, the length of e_c is known, but it cannot be easily expressed in known parameters. The vector is loosely defined here as a column of the same length as b_c, with all elements equal to -1.

Lowerbounds

In Section 5.1, a lowerbound of zero was defined for all $x_{fp}(k)$, $I_{fp}(k)$ and $B_p(k)$. The sales variables $S_{fp}(k)$ did not need a lowerbound, since they are non-negative by definition. In the vector v_{lb} however, the vector representing lowerbounds, all elements have to be defined. Since the sales variables are definitionally non-negative, setting a lowerbound of zero for all $S_{fp}(k)$ does not affect the problem. This done, all variables, and thus all elements of y have a lowerbound of zero. Vector v_{lb} becomes a vector with as many elements as y, containing $3FPK + PK$ zeroes.

For some of the products produced in external facilities, an additional lowerbound was defined in (5.12). In v_{lb}, the elements corresponding to the variables in (5.12) have been set to the (higher) extra lowerbound manually. Another option would have been to define extra constraints within matrix A. The choice between the options is arbitrary, the amount of work needed for either is about the same.

With the lowerbounds, the entire problem has been rewritten in vector notation, and according to solver standards. All matrices and vectors needed can be constructed in

Matlab. The shape of the matrices and vectors has been treated extensively in the preceding pages, so the actual Matlab code to construct those elements is not included in the main text. With the information in this section, it should not be difficult to reconstruct. The code, including comments on programming, is included on the CD attached to this report as 'test case.m'.

7.3 Postprocessing

The entire LP problem now being available in a Matlab script, the script can be run and the problem solved. The outcomes of a script run are an optimal objective function value J^* and an optimal vector y^*. The objective function value is not very relevant as an output, it is more of a tool in solving to find y^*. The optimal variable vector is the relevant output. To be able to extract any information from it, some postprocessing is needed. In Matlab, two output matrices are constructed to be exported back into Excel for analysis.

The most desirable state in the model is the state where in each period, total production of a product and its demand are equal. In that case, total sales for that product are equal to its demand, sales per fab are equal to production per fab. There is no inventory and no backlog. The first matrix is structured so that these conditions can be easily checked for each product in each period.

The first check is on the equality of total production of a product and its demand, in a certain period. For this, a column is created that gives total production, for all products and all periods. That total is the sum of production over all fabs, so it can be specified per fab. Therefore, in front of the total production column, F columns are constructed that give production per fab, again, for all products and all periods. Behind the total production column, a demand column is created.

When total production and demand equal each other, total sales should also be equal to the two. Behind the demand column, $F + 1$ columns of sales are constructed similar to the production columns: The first F columns give sales per fab, the last gives the sum over all fabs, total sales. If production and demand are not equal, this results in a change in inventory, or backlog. The size of that change is equal to production minus demand. That is the next column. Behind there, inventory columns are added. Again parallel to production: first per fab, and then the total. Backlog is set as the last column.

All desired information is now in the matrix, but for clear identification it has to be indexed. Three columns are added at the front stating period, maincap and product index. The final column lay-out thus becomes:

Period index	Maincap index	Product index	Production				Demand	
			Fab 1	⋯	Fab F	Total		

Sales				Production	Inventory				Backlog
Fab 1	⋯	Fab F	Total	− Demand	Fab 1	⋯	Fab F	Total	

The second output matrix is constructed to be able to compare fab capacities and production. To this purpose, the production vector y^x is matched to the capacity vector. To be able to do this, the capacity for a certain maincap in a certain fab is assigned to the first product in that maincap that is produced in that fab. Capacities for the other products are set to zero. In Excel, visualization tools allow to show both capacity and production for all products in one maincap, allowing for easy comparison. In the second matrix, like in the first, indexes are places in front, in this case period, maincap, product and fab indices. Once imported in Excel, both matrices are further extended by adding the original index names matching the index numbers used in Matlab. The resulting tables can be analyzed using the functions PivotTable and PivotChart.

This chapter has dealt with the implementation of the LP. In implementation, the LP equations constructed in Chapter 5 are combined with the data from Chapter 6 to result in an actual problem. A rather large part of implementation consists of the full extension of the equations into a matrix notation. By means of Matlab and LPsolve, this actual problem in matrix notation can be solved to yield an optimal variable value vector y^*. That vector is then transformed into a matrix and into a table in Excel. The table finally allows for analysis of the results. That is the subject of the next chapter, Chapter 8.

Chapter 8

Results

In the previous chapters, an LP was proposed to solve the capacity planning problem at NXP. Data for a test case, provided by NXP, have been sorted and formatted. With the test case data, an implementation of the LP has been realized. The problem has been solved, and the results processed to allow for analysis. In this chapter, a closer look is taken at the outcome of the LP. First, in Section 8.1, the expected results are presented. By taking a look at the model, some predictions can be made with respect to the outcomes. In Section 8.2, the actual results are presented and compared to the expectations of the first section.

8.1 Expectations

This section focuses on what might be expected as model output. Taking a look at the model, quite a few things can be said about what the output should look like. It is not possible to go into specifics, and predict beforehand exactly what the production of DieType 4 in fab 2 should be in period 5. If that were the case, the entire model would not be needed. However, a number of general things can be said.

The first thing that needs to be stated, is that the outcome of the LP probably does not look too much like the manual planning of the NXP planning department. NXP's planning is based on an existing previous planning, and deviations from that previous planning are kept as small as possible. The planning from the LP has no regard for any previous data, the LP operates strictly on cost minimization. On the other hand, the sourcing data does quite seriously constrain the problem, limiting the room there is for deviation from the NXP planning. In any case, the LP outcome should look plausible to the planners at NXP.

Next, a number of things can be said by looking at the objective function of the LP. Looking only at the objective, and disregarding for a moment the constraints, an ideal situation can be derived. This situation has already been sketched in section 7.3: For

all products and all periods, production is exactly equal to demand. Production takes place only at the cheapest production location. All production is sold immediately, and there is no inventory and no backlog. This situation has the lowest costs, or the highest profits, resulting in the lowest value of the objective function.

In a situation where the constraints are taken into account, this optimal situation cannot always be realized. When none of the inequality constraints are active for a certain product, the optimal situation still occurs for that product. However, this is probably not the case for all products. The capacity and sourcing constraints are expected to limit production in some cases. When this happens, a few solutions are possible.

The first thing to be done in a case where capacity is insufficient, is a (partial) move of production from the cheapest production facility to a more expensive fab. Which of the possible products is moved is irrelevant, but the sourcing constraint has to be obeyed. A lower profit is expected, but a profit is still made. Even after the inclusion of more expensive fabs, capacity for some products might not be sufficient.

If in a previous period capacity is sufficient, some of the desired product can be produced in that period and stored in inventory. Since inventory cost is low in comparison to production cost and sales price, this would be a favorable option. Since inventory cost is constant for all fabs and all products, a choice between two products to be produced in a previous period or in the period itself is arbitrary. Inventory cost is calculated per period, so production should take place as close as possible to the moment it is demanded. Inventory is expected to be empty in the last period, since no products are needed after that period.

If no capacity is available in the period itself or any previous period, that results in demand that is not filled, backlog. Although that is the most expensive option, it may not be avoided in all cases. When backlog does occur, it should be with the product with the lowest profit, or the smallest difference between production cost and sales price.

A final situation that could occur, is the case where the revenue of sales does not outweigh the costs in production. In that case, the product should not be produced at all. While missing the revenue of sale of the product, the cost of production is saved. This situation is not expected in the results, since it would indicate NXP to operate on a loss for that particular product.

An interesting question to be asked is about the exchangeability of capacities. The exchangeability has been incorporated in the model, but beforehand it is not sure that it is actually helpful. However, exchangeability increases system flexibility. As different maincaps may have different costs in different fabs, capacity shifts might result in lower overall cost. Some exchanges are expected in the results, but no accurate prediction can be made.

8.2 Actual results

In this section, the output from the test case LP is presented and analyzed. The actual data is not included here, but can be found on the CD, in the Excel file 'test case.xls'. The output does prove to follow expectations quite well. The different scenarios presented in the previous section are handled below.

First of all, the results have been checked by E.J.J. van Campen, working at the industrial planning department of NXP. Although there are differences with the NXP planning, he does not note any large abnormalities. He concludes that the LP returns a plausible planning for the test case problem. This conclusion validates further analysis.

The next conclusion that can be drawn, is that for the majority of products and periods, production and demand are well-matched. In just over two thirds of cases (68%), total production and demand are equal. In all of these cases, total sales also equal demand and production. This situation corresponds to the scenario where more expensive fabs have been included, and sufficient capacity is available.

When demand and production are not equal, in a further 27% of cases, demand is still completely fulfilled. Of these 27%, half shows a production larger than demand. This means an increase in inventory: Besides filling demand for that period, there is sufficient capacity to fill (part of) demand in a later period. The other half shows a production smaller than demand. In those cases, inventory shows a decrease: Products from inventory are used to fill the gap between production and demand, and thus demand is still completely filled. Note that the fifty-fifty split between these two scenarios is coincidental: One large overproduction can be used to fill several small gaps, and vice versa.

In total, demand is completely filled in 95% of cases. The backlog scenario, where demand cannot be completely filled, thus occurs in about 5% of cases. As a percentage of total demand, less than 2% of total demand is not filled. All products that experience backlog are in the lower half of the profit range. This supports the expectation of backlog occurring with low-profit products.

The results also show quite some capacity exchanges. In Figure 8.1, stated capacity (sum of capa) and actual production (sum of prod) are shown for the exchangeable maincaps in the relevant fabs. Two periods are shown.

It can be seen in the figure that there are few major exchanges, but quite a few small ones. The largest exchange is in fab 2 in period 2, where a one-to-one shift of about 7.5k wafers has taken place from maincap 3 to maincap 1. The exchange rate of 1 between those two maincaps corresponds to the one-to-one exchange. In the same place, a small amount of maincap 2 is produced, even though no capacity is given for that maincap. By expressing total capacity and total production in one maincap, it can be seen that the total capacity is used in this case.

Fab 4 in period 4 also shows an interesting thing: production is larger than capacity. By

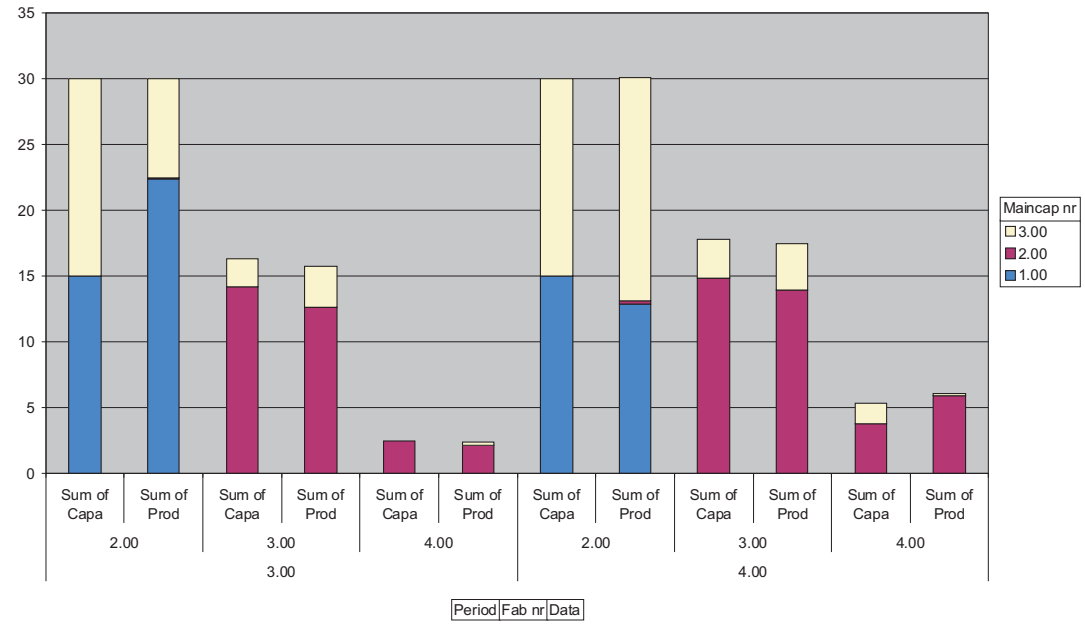

Figure 8.1: Maincap capacity exchange

an exchange at a rate of 1.6 of capacity of maincap 3 for maincap 2, absolute capacity is extended. Expressing capacity and production in a single maincap, production can again be seen to equal capacity.

8.2.1 Unexpected

The situation where revenue does not outweigh costs does not occur, as expected. However, for two products, OM48001 and OM6384-3A in maincap 0.25umB, a related situation is found. These two products have been given a lowerbound on production larger than zero in an external fab. It can be seen that production for the two products is set to that lowerbound in that fab, but production is not used to fill demand. Instead, all production of those two products in the external fab is stored in inventory. Inventory grows at a constant rate, and at the last period, inventory is not empty. The same two products are also manufactured in a different fab, and there, production and sales are equal to demand.

The cause of this phenomenon is found by taking a look at the costs and benefits of producing a single one of either product, in two scenarios. First, the cost of the expected scenario is calculated: The product is produced in the external fab, and sold from there in the same period. Total cost thus equals: $c_e^a a_p - s_{ep}$. The subscript e denotes the external fab, p the product concerned. The constants can be filled in from the data files.

Then, the actually occurring scenarios costs are calculated: The product is produced in and sold from the internal fab, *and* the product is produced in the external fab and stored in inventory there. Total cost becomes: $c_f^a a_p - s_{fp} + c_e^a a_p + c^I$ (subscript f denoting the internal fab). Filling in all values from the data and comparing the two equations, it shows that:

$$c_e^a a_p - s_{ep} > c_f^a a_p - s_{fp} + c_e^a a_p + c^I, \text{ so} \tag{8.1}$$

$$s_{ep} < s_{fp} - c_f^a a_p - c^I. \tag{8.2}$$

In other words, the profit for the internal fab is larger than the sales price of the externally produced product, and thus larger than the profit in the expected scenario. It can be concluded that for these two products, production at the external facility should be stopped. Even though a profit might be made, profits are higher when production is completely shifted to the internal fab.

8.2.2 Concluding

Accounting for the unexpected result concludes the analysis of the results. The expectations could all be confirmed in the actual output. Even though the model has not been explicitly validated, it does appear to work correctly, and it gives sensible results.

The analysis of results is not the main scope of this report, the construction of a working model is. The analysis of the results is therefore not exhaustive, it is simply a final step in showing that the LP can be constructed, and that it works correctly. The next and final chapter gives the conclusions for the entire project, and deals with a number of recommendations that can be made for the improvement and possible continuation of the project.

Chapter 9

Conclusions and recommendations

In this final chapter, two things are done. First, in Section 9.1, the conclusions of this project are presented. Next, Section 9.2 contains a number of recommendations for improvement of the project and extended research.

9.1 Conclusions

Until now, the capacity planning of the front end of manufacturing operations at NXP has been a manual job. In the article by Kempf [2], a different method to solve the problem is proposed: The capacity planning problem can be formulated as a linear program (LP). This allows for the problem to be solved by computer. Moreover, for as far as the model is representative, it returns an optimal solution. Optimality cannot be verified in the manual situation. Thus, the LP may be an improvement to the present method.

The objective of this project was to develop such a linear program model to calculate the optimal capacity planning for a case study provided by NXP. To this purpose, the capacity planning problem had to be translated into a mathematical problem. A linear program tries to find the minimum value of an objective function by varying the decision variables of the problem, while satisfying the constraints posed on the problem.

To match this problem lay-out, the capacity planning problem was posed as a cost minimization. There are costs associated with production, keeping inventory or failing to meet demand, and there is revenue in selling the produce. The total cost of a planning has to be minimized. The parameters that can be varied to find this minimum, the decision variables, are the production amounts of all products in all production facilities in all time-periods in the planning. A number of extra variables has been defined to

be able to completely capture the problem: inventory, backlog and sales variables. The extra variable values are coupled to the decision variable values in the constraints.

The problem solution is constrained in several ways. The values the variables can take are limited. The first two constraints are coupling constraints, coupling the values of the extra variables to the decision variable values. The first constraint is an inventory mass balance, coupling the inventory and backlog variables to demand and production. The second is a sales balance, linking the sales variables to inventory and production. Both are equality constraints. The other constraints are inequality constraints, limiting the values the decision variables can take. The sourcing constraint identifies which products can be produced at what locations. The capacity constraint limits the total production to the capacity of the facility. The last constraint sets lowerbounds on the values of all variables excluding the sales variables. Sales lowerbounds are implicitly defined. None of the variables can take negative values, so a lowerbound of at least zero is set.

To be able to apply the proposed LP to the test case, two things had to be done. First, the data provided had to be formatted. The data provided by NXP came in different formats, none of which was particularly suitable for use with the LP. A standard data format was developed, and all data was transformed into that format. Secondly, the LP had to be implemented in a mathematical program to allow for solving. As a solver, the free internet download LPsolve was chosen. Matlab was used as platform for the solver. A Matlab script was written to construct the constraint matrices and objective function from the NXP data, according to solver specifications.

After solving the LP, analysis was performed to check the correctness and validity of the model. The model was concluded to behave correctly: The constraints were met and a viable solution was found. Also, model validity was checked and approved by NXP. No irregularities were discovered.

While working on the project, a number of possible improvements and project extensions have been found. They are treated in the next section of this chapter.

9.2 Recommendations

As with any project, the end result of this project is not perfect, nor is the work really finished. While working, possibilities for improvement are discovered, and it is not always possible to incorporate those improvements right away. Because of a lack of time, or a late discovery of the improvement opportunity, the only way to include them may be as recommendations at the end of the research report. Recommendations of this kind can be found in the next section, Section 9.2.1.

As mentioned, the work needs not be finished either. The project may have rendered new questions, or may be part of a bigger project. The current project can be extended in several ways, and it may serve as the starting point for a number of new projects. Options for further research are given in Section 9.2.2.

9.2.1 Improvements

As said before, a number of possible improvements have been found during the work on this project. They are treated below.

Backlog

In the current LP model, a backlog variable is present (See Chapter 5). This variable has been incorporated in the model in an early stage of the project. At that stage, production in a later period than demanded was allowed, backlog in its known definition. As backlog does not occur in that situation at NXP, the definition of backlog was adapted in the model to mean lost demand. As a result of this change, the backlog variable itself has become redundant. The variable could therefore be eliminated from the model. The missed demand it represents can be calculated from the other variables. The resulting model would be one with fewer variables, and might be easier to understand.

Capacity and sourcing constraints

In the implementation of the proposed LP, the capacity and sourcing constraints are combined in a single equation (See Section 7.2.2). In actual fact, the combined constraint had been modeled before the separate constraints were defined. The single constraint has been separated to obtain a clear definition of the problem, but in modeling, this has not been adapted. As a result, some complex operations have to take place to translate between the two constraint definitions.

The model could be adapted to reflect the two separate constraints: Each constraint is modeled apart from the other. Although this would increase the number of constraints on the problem, it would simplify the model. As the number of equations does not pose a problem in solving the LP, this is a preferable situation.

Sales prices

In Chapter 5, the sales price of a product is introduced as a time-independent parameter. In reality, this parameter is time-dependent. Because of the rapid development of new products and production methods in semiconductor manufacturing, the value of a product can decrease significantly over the planning period. The sales price data supplied by NXP reflect this time-dependency. Incorporation of time-dependent sales prices in the LP can be realized without much trouble: Where the vector f^S from (7.12) presently contains K times the same subvector, it becomes a single large vector. Thus, with a small extension of the model, a much more accurate representation of reality can be obtained.

Incomplete data

In Chapter 6, all data was gathered that was needed to solve the LP. However, in a few cases, NXP data was inconsistent or incomplete. Sourcing and demand data were found not always to match, a fab not sourced for a product would be assigned production for it. Products and processes could not be matched in all cases. Sales data could not be completed because not all processes had prices associated with them. The number of masksteps needed for a process could also not be found in all cases. Lastly, no information was available with regard to the cost of production in external fabs.

In this project, the missing information has been complemented by using similar data, or by assuming values. For the model to give correct results, the correct data has to be available. Thus, this information has to be found for the model output to be correct, or the model has to be adapted so that the information is no longer needed. First it is recommended to adopt either of these solutions, so that the model works with complete data.

Secondly, to make sure that all information is present, it is advised to use exhaustive data systems. For example, sourcing data is only provided in the positive sense: products and fabs that match are in the data, products and fabs that do not match may or may not be mentioned. An exhaustive data system would give the status for all products in all fabs, so that no confusion is possible.

Reduction of manual labor

In the creation of the current model, quite a lot of manual labor has been used. Most of this labor is in data formatting, and cannot be avoided. However, in the data use in the model, some reductions can be realized. Most notably, there are two data types that have been manually implemented in the model. Automation of the data use for these two types not only reduces manual labor, it also makes the model easier to use. The separation between the capacity and sourcing constraint mentioned previously actually facilitates the proposed automation.

The first data type is the exchange ratio between maincaps, $r_f^{mn}(k)$. The actual exchangeability of different maincaps is rather limited. In one-time modeling, as is the case here, manual implementation is the faster method. It saves on writing out all equations for non-exchangeable maincaps, but does result in a more complex capacity constraint matrix. In regular use, the automated approach is thought to be better. When the ratios are all written down in a matrix, and this matrix is used in calculating the capacity constraint matrix, the number of equations increases, but the implementation of the ratios is no longer a complex manual job. As mentioned before, the number of equations does not pose a limit in this problem, and ease of use is preferred.

The second data type implemented manually is the sourcing data. Automated use would be difficult in the combined constraint, but when the sourcing constraint is mod-

eled separately, automation is quite easily realized. Again, the main advantages are a reduction of manual labor and a reduction of complexity for the user.

A third way to reduce manual work is to establish an automatic coupling between dietypes and processes. Some of the NXP data is given by process, while it is needed by dietype. Thus, the process associated with a dietype has to be found before the information for that dietype becomes available. In this project, finding the right process has been a manual job. An automated cross-referencing system could be developed to eliminate that job.

After automation of all the data use, a fourth way of reducing manual labor also becomes feasible. In the current approach, all small-volume products are summed. This results in a lower number of products to be incorporated in the model, and thus in a reduction of labor in later stages. The data handling involved in summing those products (increasing labor) is expected to be less than that reduction. With the automation, the number of products becomes less important, and the reduction by summing is no longer expected to outweigh the work involved in it.

Excel

If the proposed model is ever to be used in practice at NXP, it will have to work with Microsoft Excel. This is the program used for all data applications at NXP. In this project, the LP was solved in Matlab, but with a solver that can also be run from within Excel. An implementation of the exact same LP in Excel is therefore possible. That implementation would save a large amount of work in data exporting and importing, and again, make the model easier to use in practice.

9.2.2 Future research

Besides improvements on the current work, some recommendations can be made with respect to further research, continued from or based on the work done here. The possibilities for continued research are presented below.

Implementation of the entire capacity plan at NXP

In this project, Kempfs approach to capacity planning has been used on a test case. Five fabs and five production technologies with some 55 products were modeled. The test case has proven the approach to work on a small scale. The main difference with a larger scale use is in the data handling. There being a larger amount of data however, does not increase the complexity of the model itself. If the LP is ever to be used in practice, it has to implemented full-scale, for all fabs and all products of NXP. This is not so much interesting from a researchers perspective, but it would be of much use for NXP.

A more dynamic use of the proposed model

The model proposed in this project can be used as a tool in the capacity planning at NXP. However, it is not a dynamic model. In reality, semiconductor manufacturing is highly dynamic. Products are taken out of production, new products are introduced, growth products are transferred to new locations, even new technologies can be introduced within the time scale of a single capacity plan. For the model, decisions with respect to introductions and transfers have to made a priori. For example, a new product is assigned a production facility via the sourcing data. Whether or not the choice for that facility is a good one, is not within the scope of this project. The data are simply implemented as they are.

However, the model could be used in a more dynamic setting, and be of help in the making of those decisions. Especially when the model is made easier to use, different scenarios can be implemented and compared. In that way, a better choice (the optimal choice) can be made with respect to transfers and product introductions.

Another consequence of the semiconductor industry being so dynamic, is the uncertainty in demand. The model works with demand forecasts, but actual demand may be quite different from the forecast. An important question is how sensitive the model output is to such changes in demand (and other parameters). If a slight change in the demand results in a very different plan, the model output is not very robust, and it may actually not be of very much use. Sensitivity analysis of the model and its output answers the question of sensitivity. As different plans may have different sensitivities, the model can again be used to find the best scenario.

Further implementation of the approach by Kempf

In the current project, one part of the approach of K. Kempf has been implemented for NXP: The part concerning the capacity planning, what he calls strategic planning function, and only for the front end. However, this is only one part in a much larger integrated approach, including other aspects of semiconductor manufacturing operations. Kempf proposes to use the strategic planning function in tandem with an inventory planning formulation and a tactical execution function. Also, in his view, the approach can be used for the entire manufacturing process, including back end. Thus, there are two ways to extend the implementation of Kempfs functions.

Dealing first with the more straightforward extension, the strategic planning function can be applied to the back end as well as the front end. However, the planning problem for the back end is different from that for the front end. Two important differences are production system and planning horizon. While front end uses demand forecasts and has a planning up to 18 months ahead, back end produces to order, with a time horizon of only about 6 months. Before starting implementation of the strategic planning function for the back end, its use should be investigated.

The other extensions of Kempfs approach have to do with the other functions he proposes. Both the inventory planning formulation and the tactical execution function might be used within NXP. The inventory planning can be used to optimize the material flows in the manufacturing network. The tactical execution function can optimize operations in the separate fabs. For both inventory planning and execution function, an investigation is recommended to identify the feasibility of implementation at NXP. Also, the advantages and disadvantages have to be identified.

Bibliography

[1] M. Berkelaar, K. Eikland, and P. Notebaert. *lp_solve (alternatively lpsolve)*. Open source (Mixed-Integer) Linear Programming system, http://lpsolve.sourceforge.net/, Version 5.1.0.0 dated 1 May 2004.

[2] K Kempf. Managing supply-demand networks in semiconductor manufacturing. In D. Armbruster, K. Kaneko, and A.S. Mikhalilov, editors, *Networks of Interacting Machines*, chapter 3, pages 67–100. World Scientific, 2005.

[3] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004.

Appendix A

Data file index

All computer files concerning this project can be found on the CD accompanying this report. The following files are present:

- **NXP Internal wafer prices.xls**: The original Excel file as received from NXP containing the sales price data for the test case.
- **NXP Masksteps Mar07.xls**: The original Excel file as received from NXP containing the maskstep data (the number of masksteps for each product) for the test case.
- **NXP Sourcing.xls**: The original Excel file as received from NXP containing the sourcing data for the test case.
- **NXP Variable cost per mask.xls**: The original Excel file as received from NXP containing the data for the test case concerning costs per maskstep.
- **NXP MTBP dec06 demand analysis.xls**: The original Excel file as received from NXP containing the demand and capacity data for the test case.
- **test case.m**: This Matlab file contains the implementation of the test case LP in Matlab. Included in the code are comments with respect to programming, explaining the approaches used.
- **test case.xls**: This Excel file contains all model inputs and outputs, as well as several extra sheets, included for clear data representation and handling or for analysis. The first sheet contains an index stating the content of each of the other sheets.
- **Kempf.pdf**: The article 'Managing supply-demand networks in semiconductor manufacturing' by K. Kempf.