RC21509(96951) 25MAY1999 Computer Science/Mathematics 26 pages

Research Report

Implementing a Multi-Echelon Inventory System at a Semiconductor Company: Challenges and Results

Alexander O. Brown
Owen Graduate School of Management
Vanderbilt University, 401 21st Avenue South
Nashville, TN 37203

Markus Ettl
IBM Research Division, T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Grace Y. Lin
IBM Research Division, T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Raja Petrakian
Xilinx Inc., 2100 Logic Drive
San Jose, CA 95124

Research Division

Almaden – Austin – China – Haifa – T.J. Watson – Tokyo – Zurich

Implementing a Multi-Echelon Inventory System at a Semiconductor Company: Modeling and Results

Alexander O. Brown[1]
Owen Graduate School of Management
Vanderbilt University, 401 21st Avenue South
Nashville, TN 37203
e-mail: alex.brown@owen.vanderbilt.edu

Markus Ettl
IBM Research Division, T.J. Watson Research Center
Yorktown Heights, NY 10598
e-mail: msettl@us.ibm.com

Grace Y. Lin
IBM Research Division, T.J. Watson Research Center
Yorktown Heights, NY 10598
e-mail: gracelin@us.ibm.com

Raja Petrakian
Xilinx, Inc., 2100 Logic Drive
San Jose, CA 95124
e-mail: raja.petrakian@xilinx.com

Abstract

We develop an inventory allocation model for the use in the semiconductor industry. The model allows for multi-echelon, multi-item optimization. The objective is to maximize the system-wide serviceability subject to constraints on inventory holding cost, allowing consideration of both build-to-order and build-to-stock finished good items. The key components of the model are an approximate method for performance evaluation and a two-phased optimization algorithm, combining heuristics with nonlinear optimization. The model allows for fast decisions when applied to supply networks of upwards of one thousand stocking locations. Results from implementation at Xilinx, Inc., show dramatic improvements in performance over current inventory policies.

Keywords: Application, multi-echelon inventory model, semiconductor industry, optimization*.*

[1] Part of work undertaken while employed at Xilinx.

1. Introduction

High-technology manufacturing has many characteristics that often makes managing operations a significant challenge. First, inventory has rapidly declining value and a high risk of obsolescence due to short product life cycles. Second, demand is highly uncertain, a characteristic that is exacerbated by the proliferation of products. Third, for many high tech manufacturers, the manufacturing lead times are long. And fourth, near immediate availability of many of the products is often required by the customer, as is the case with the company described in this article. Thus, inventory is necessary, each unit carries a large effective holding cost, and significant quantities are needed to ensure a reasonable level of customer service. Hence, properly managing the inventory is a critical task.

In this paper, we describe a semiconductor manufacturer that faces an environment with the above characteristics. The company is Xilinx, a firm that produces programmable logic integrated circuits (ICs) for sale to original equipment manufacturers in such industries as networking, computer peripherals, and telecommunications. With manufacturing costs in the industry declining at rates of anywhere from 20% to 75% annually and with obsolescence costs on the order of 5-10% of the gross inventory, inventory is clearly costly and risky. Xilinx has upwards of 10,000 different finished products, only about 5% of which would be classified as having stable demand. Thus, demand would certainly be classified as highly uncertain. Total manufacturing lead times are 2-3 months, requiring extensive fabrication, assembly, and testing in a variety of different facilities. Delivery requirements are stringent with a significant fraction of customer orders being requested for same-day delivery. Service is absolutely critical since although a Xilinx IC may only be one component of hundreds in a customer's product, delays in delivery may shutdown a production line. Thus, delivery speed and responsiveness are important for customers' selection and retention of Xilinx as a supplier.

Customer delivery requirements, long manufacturing lead times, and volatile demand require that considerable inventory be held to provide a reasonable level of service. In 1998, leading programmable logic manufacturers were holding 80 to 150 dollar days-of-inventory. (The inventory measured in dollar days is the net inventory divided by the average cost of goods sold per day.) However, since this inventory is risky and expensive, it is critical to lower this level. IC manufacturers are using product and process design strategies like postponement [1, 2] to help lower the required inventory. With a postponement strategy, inventory is held at an intermediate point in a generic, non-differentiated form and is only differentiated when demand is better known.

Simple adoption of a postponement strategy offers significant advantages, particularly since it allows one to build many of the finished goods parts to order from the generic inventory held at the intermediate stocking point [1]. However, in order to take full advantage of the postponement strategy, one must determine the inventory levels in the intermediate and finished goods points that allow for the best service at the lowest cost.

The question of how much inventory to hold at each point under a postponement strategy was one of a number of key questions that arose out of a demand supply chain initiative at Xilinx that began in 1996. The desired answer would be a set of inventory targets for each product and production stage, which would be the driving input for the MRP-like planning systems that have been and are currently in use at Xilinx. Initially, the problem was attacked by developing spreadsheet-based single-echelon inventory models. After developing an intuitive understanding of the output and successfully establishing planners' confidence in the models, the authors began development and implementation of a full multi-echelon model and automated system.

In this paper, we describe the actual use of a sophisticated inventory system at Xilinx, describing the implementation work and the integration with current planning systems. We demonstrate the significant gain in performance that the system provides. We fully describe the inventory model. The objective of the model is to find the set of inventory targets at all inventory buffers for each part throughout the supply chain that minimizes overall dollar delinquencies subject to a constraint on inventory budget. Thus, it is a full multi-item, multi-echelon optimization problem. We describe the use of decomposition with infinite server queues to evaluate the objective function, and present a two-phase heuristic that is used to solve the optimization problem.

In Section 2, we provide a brief survey of the related literature. In Sections 3, we describe the manufacturing and planning process at Xilinx. In Section 4, we discuss the implementation and the usage of the model and formulate the inventory optimization problem. In Section 5, we describe the heuristic developed to solve the problem. In Section 6, we discuss an analytical procedure to approximately compute the number of backorders in the Xilinx supply network. In Section 7, we present numerical results and observations, illustrating the financial and service impact of the model.

2. Related Literature

Models that address the problem of how to optimally allocate inventory in complex supply chains are largely unavailable, although much work has been devoted to developing quantitative performance models for such systems. Axsaeter [3] and Federgruen [4] give overviews of models and algorithms for inventory management in multi-echelon inventory systems. There have been a few documented instances of the use of multi-echelon models in practice. The classic model of a continuous review, multi-echelon inventory system was developed for the U.S. Air Force as a planning system for managing inventory of repairable items [5]. The model consists of two echelons, the upper one consisting of one single depot supplying repaired parts, and the lower one consisting of n identical stocking locations. The objective is to identify a stocking policy at the two echelons which minimizes backorders at the lower echelons subject to a constraint on inventory investment. Sherbrooke [6] improved the accuracy of the model.

Cohen et al. [7] developed a system called Optimizer that allowed IBM to make strategic changes to the configuration and control of a spare parts inventory network. Lee and Billington [8] describe the use of a multi-echelon inventory model to evaluate the performance of a printer supply chain at Hewlett Packard. Graves et al. [9] developed a model for requirements planning in a multi-echelon inventory system. The analysis is based on a deterministic model for a single echelon that serves as a building block for modeling multi-echelon systems. The model was used to determine inventory placement in a film manufacturing line at Kodak. Ettl et al. [10] described a model to characterize the inventory-service tradeoff in multi-echelon supply networks. The model features performance evaluation and optimization with service level constraints. It captures the interdependence of inventory levels at different stocking locations, as well as their effect on overall system performance. Feigin [11] discusses an example of the application of this model to a personal computer manufacturing company. Other optimization models for supply and distribution networks include Arntzen et al. [12] and Andersson et al. [13].

3. Manufacturing Process at Xilinx

Founded in 1984, Xilinx is a leading producer of programmable logic integrated circuits with $662 million in annual revenues in 1998 (fiscal year 1999). Such programmable logic fits within the larger category of logic devices, a category of devices that provide more specific functionality at lower cost and faster speed than more general-purpose microprocessors. Programmable logic devices (PLDs) have different characteristics such as speed, voltage, and gate count and have many different packaging types. However, unlike traditional logic devices, known as Application Specific Integrated Circuits (ASICs), the specific logic and functionality is not configured during physical fabrication. Instead, the specific function of the device is programmed after fabrication by the customer through software. Thus, when specification changes are needed in the design, the change can be made almost instantaneously rather than waiting months for a new device to be fabricated. Although more expensive (per unit) than ASICs, programmable logic offers increased design and manufacturing flexibility and gain in time-to-market for customers' new products. These products are marketed as having high availability, either immediate availability or within a short lead time, as opposed to ASICs that are often built-to-order with lead times on the order of months.

There are two basic stages in the IC manufacturing as illustrated in Figure 1: fabrication and assembly & test. In fabrication, many identical die are created on a raw silicon wafer through a complex process of adding and removing patterned layers of various materials. Each die on the wafer will correspond to a given device. In assembly and test, the wafers are sliced into the individual die, and each die is assembled with a particular package type and tested for functionality and speed. Xilinx, like many semiconductor manufacturers, contracts out most of the manufacturing. Fabrication is performed at the vendors UMC in Taiwan and Seiko in Japan. The wafers are then shipped to assembly vendors in Korea, Taiwan, and the Philippines, where they are held in die bank inventory until needed. Assembled parts are usually shipped to Xilinx facilities in San Jose or Ireland for testing, where they are held in finished goods inventory. Some parts are tested by the assemblers and drop-shipped directly to customers, however, this only represents a small fraction of the total demand.

FIGURE 1 ABOUT HERE

The inventory held in die bank is fairly generic, consisting of about 200 different die types based on features such as voltage, gate count, and design architecture. Differentiation of the device into finished goods occurs during assembly and test. With different package types, grades, pin counts, speeds, and customer unique specifications, the total number of finished good types is very large. Currently, there are approximately 10,000 different finished goods parts that are active in Xilinx planning systems. Figure 2 is an illustration of the product structure.

FIGURE 2 ABOUT HERE

Since die bank inventory is more generic, it serves as an inventory postponement point. The die bank inventory is managed using a wafer starts planning package. Given a desired inventory target, this package determines the total amount to start at the fabrication contractor using a monthly base-stock policy. That is, the total inventory on-hand and in WIP (work-in-process) is netted out of total requirements (forecast plus inventory target) for all finished goods that use each type of die. Finished goods inventories are managed using a planning system based on assembly starts, with separate systems being run in Ireland and San Jose. Finished goods are classified as build-to-order or build-to-stock. For build-to-order parts, assembly requests are made according to customer orders – die are pulled from the die bank and started into assembly. For build-to-stock parts, assembly starts are determined from a weekly base-stock inventory control policy, driven by a forecast. The classification is adjusted monthly using an automated process based on volume, demand variability, cycle time, holding cost, and shortage cost. In 1998, about 80% of parts were build-to-order. However, this represents only 10% of the total volume.

4. Inventory Optimization at Xilinx

The purpose of the multi-echelon inventory optimization is to determine the inventory target levels of the die bank and finished good buffers for all parts. To understand the inventory structure, consider the product structure shown previously in Figure 2. Each triangle represents an inventory buffer for which an inventory target must be determined. The shaded triangles represent the products in the intermediate stage (die bank), unshaded triangles represent finished goods that are build-to-stock; circles represent finished goods that are build-to-order. Finished good inventories in Ireland and San Jose are kept separate for tax efficiency.

The integration of the inventory optimization within Xilinx's current MRP planning system is shown in Figure 3. Demand, inventory, cost, price, and other part specific data are collected from various databases and processed using database programming scripts. A graphical user interface (GUI) is used to collect various user-specified information such as the overall inventory level and the type of optimization. The data is transformed into relational data files necessary to run the inventory optimization code. The output of the optimization is a recommended target level for each inventory buffer. This output is translated into projected service and inventory levels. Using this output report, the supply chain planner can adjust various parameters in the GUI and re-run the model. The final inventory targets are loaded into the MRP-like planning system described earlier. The planning system uses these targets along with forecasts, order backlog, current on-hand inventory levels, and WIP inventory levels to determine production starts to be issued to foundries (for wafers) and back-end assemblers (for finished goods packaging).

FIGURE 3 ABOUT HERE

As with most inventory decision problems, choosing the appropriate objective function is difficult and highly dependent on the business environment. A commonly used method is to find the minimum inventory levels so that each finished good part achieves an expected level of service, e.g., 90% fill rate. With this method, each finished goods part corresponding to the same die could be treated independently. At Xilinx, however, managers felt uncomfortable with that approach since they wanted parts to have different service levels and were very unsure about what those service levels should be. Additionally, they wished to be able to specify an input that would tie directly to key measures in the company's overall financial statements (balance sheet and income statement), measures by which the managers and company's performance is judged. This input was chosen to be the overall corporate or business unit inventory level. Thus, they wished to specify such an inventory level and have the inventory model determine the best allocation of the inventory in order provide the highest level of overall service. Thus, the problem is a complex, multi-item optimization instead of a one that is separable among items.

The service objective to minimize was chosen to be total dollar delinquencies (backorders) across all parts, with each part having a different unit delinquency cost. The number of delinquencies for build-to-stock parts is defined to be the number of demands not filled in a given week, with delinquencies from the previous week becoming additional demand in the current week. For build-to-order parts, the delinquencies are the number of demands that cannot be filled within a lead time.

The inventory budget constraint for the actual optimization is specified as the total inventory holding cost budget, representing the total amount of money the company is willing to spend holding inventory (the cost of capital and loss due to declining costs). This measure was chosen instead of simple inventory budget since different parts have very different holding cost rates depending on its stage in the product life-cycle, technology, etc. Management want to express the budget in terms of dollar days-of-inventory ($DOI) since they have a good intuitive feel for its value and since it is the measure used to present inventory results in key financial statements. Thus, using a weighted-average holding cost and using current inventory targets, we automatically convert the inventory budget in $DOI into an approximate holding cost budget to be used as a constraint in the optimization.

The actual optimization can take on one of three forms, depending on the way in which inventory budgets are provided. In the first form, a total inventory budget is provide by the user and allocated across all buffers in die bank and finished goods. This form requires true multi-echelon, multi-item optimization. In the second form, separate budgets are specified for the die bank buffers and the finished goods buffers. The decision is how to allocate the die bank budget among die bank buffers and the finished goods budget among finished goods buffers. The third form allows specification of separate budgets for die bank, finished goods buffers in San Jose, and finished goods buffers in Ireland.

The parameters of the formulation are defined as follows:

M	The number of finished goods locations in the supply network
N_0	The number of products at die bank
N_i	The number of products at finished goods location *i*, *i=1,2,…,M*
h_{ij}	The inventory holding cost per unit for product *j* at location *i, i =0, 1,2,…,M* and *j=1,2,…,* N_i
s_{ij}	The shortage cost per unit for product *j* at finished goods location *i, i =1,2,…,M* and *j=1,2,…,* N_i
x_{ij}	The base-stock level for product *j* at location *i, i=0,1,…,M* and *j=1,2,…,* N_i
$x_{ij}^{\min}$	The minimum base-stock level for product *j* at location *i, i=0,1,…,M* and *j=1,2,…,* N_i
$x_{ij}^{\max}$	The maximum base-stock level for product *j* at location *i, i=0,1,…,M* and *j=1,2,…,* N_i
$\mathbf{x}$	The vector of base-stock levels $\mathbf{x} = (x_{01}, x_{02}, ..., x_{MN_M})$
$Q_{ij}(\mathbf{x})$	The number of backorders for product *j* at location *i, I=0,1,…,M* and *j=1,2,…,* N_i
$b_{ij}(\mathbf{x})$	The inventory holding cost allocated to product *j* at location *i, i=0,1,…,M* and *j=1,2,…,* N_i

Given the above definitions, we can define the management objectives and constraints in terms of customer serviceability and inventory. Mathematically, the optimization problem is formulated as

$$\min G(\mathbf{x}) = \sum_{i=1}^{M} \sum_{j=1}^{N_i} s_{ij} E\left[Q_{ij}(\mathbf{x})\right] \tag{1}$$

subject to the following constraints:

$$x_{ij}^{\min} \le x_{ij} \le x_{ij}^{\max} \text{ for i=0,1,…,M and j=1,2,…,} N_i \tag{2}$$

$$0 \leq \sum_{i=0}^{M} \sum_{j=1}^{N_i} b_{ij}(\mathbf{x}) \leq B \qquad (3)$$

where *B* denotes the overall inventory holding cost budget for all stocking locations in the network. The decision variables are the base–stock levels x_{ij} which denote the inventory targets, target safety stock being the base-stock level minus the expected demand over the actual lead time.

The first set of constraints (2) impose upper and lower bounds on the base-stock level, and thus on the inventory holding cost, for any given part in the supply chain. Although some obsolescence cost is built into the unit holding costs, the bounds allow planners to ensure that the model does not choose high targets for parts with high obsolescence risk. For example, if a part is reasonably mature or has only a few major customers, the obsolescence risk is high and the planner certainly would not want to allow a high target inventory in finished goods. Alternatively, if a part is in the early stages of its life-cycle, supply is volatile and product availability is crucial, so a planner is likely to set a large minimum target in die bank. The bounds also allow the planners to incorporate additional concerns such as yield and technology uncertainty and to ensure that the model does not generate any nonsensical decisions caused by some strange deviations in the historical data used as input.

The second constraint (3) imposes an upper bound on the holding cost budget. When a separate inventory holding cost budget is imposed on die bank buffers and finished goods buffers (the second form of the optimization), constraint (3) is replaced by a new set of constraints stated as follows:

$$0 \leq \sum_{j=1}^{N_0} b_{0j} \leq B_0 \text{ and } 0 \leq \sum_{i=1}^{M} \sum_{j=1}^{N_0} b_{0j} \leq B - B_0 \qquad (4)$$

where B_0 denotes the maximum holding cost budget for die bank buffers.

When a separate inventory holding cost budget is imposed on die bank buffers and finished goods buffers at different finished goods sites, constraint (3) is replaced by the following set of constraints:

$$0 \leq \sum_{j=1}^{N_0} b_{0j} \leq B_0 \text{ and } 0 \leq \sum_{j=1}^{N_i} b_{ij} \leq B_i \text{ for } i = 1,2,\ldots, M \qquad (5)$$

where B_i denotes the maximum holding cost budget for finished goods location *i*. Notice that constraint (5) defines the general case where finished goods inventory can be held at *M* different sites.

After the inventory optimization is run, an output report is produced that uses the current inventory level and the inventory targets to project inventory levels for each month over a 6 month planning horizon. This projection is important because the current inventory level (actual on-hand or in WIP) may be much higher than the current target. The report sums up the total projected inventory by month. The planner can then compare this total against the initial DOI budget selected. If it is significantly higher, there are a large number of parts with significant excess inventory, and the desired DOI budget should be adjusted before re-running the model.

5. Optimization Algorithm

To solve the set of inventory planning problems defined above, we developed a two-phase optimization algorithm. The algorithm consists of a construction phase and a local optimization phase. In the construction phase, a high-quality starting solution to the inventory problem is generated. In the local optimization phase, a nonlinear optimization is performed which determines a locally optimal solution. The decision variables are the safety stock levels of all build-to-stock parts in the supply chain.

To start the construction phase, we first compute a weight coefficient, w_{ij}, for each product *j* held at finished goods location *i*. The weight coefficient is determined by three factors: its unit holding cost h_{ij}, its shortage cost s_{ij}, and the uncertainty of end-customer demand for this product. The weight coefficients are computed as follows:

$$w_{ij} = \frac{s_{ij}}{h_{ij}} c_{ij} \text{ for } i = 1,2,\ldots,M \text{ and } j = 1,2,\ldots,N_i \tag{6}$$

The first term on the right-hand side of equation (6) represents a product's shortage cost, standardized with respect to its inventory holding cost. It ensures that a larger weight is assigned to products with larger standardized shortage cost. The second term, c_{ij}, represents the coefficient of variation of demand over the actual lead time of a product. Incorporating c_{ij} into the computations allows us to differentiate between products with stable demand and products with more volatile demand patterns. Given that two products have identical standardized shortage costs, the product with a higher demand uncertainty is given a higher priority. Using the weight coefficients, we then compute a priority for every finished goods product as follows: the highest priority is assigned to the product with the largest weight coefficient; the second highest priority is assigned to the product with the next largest weight coefficient, etc. This process continues until a priority has been assigned to every product held at a finished goods location.

Using the priority scheme, the computation of the starting solution is accomplished by means of a prioritized greedy allocation scheme. Like a standard greedy algorithm, the idea is to divide up the inventory into small "buckets" and iteratively allocate each "bucket" to the buffer in the network that contributes most to the objective function. With thousands of buffers, checking the entire set of buffers for each iteration would be far too time-consuming. Thus, for each iteration, the algorithm checks all die bank buffers and a subset, or "stepping window", of active finished goods parts. As we describe in detail below, the stepping window is adjusted each iteration according to the outcome of the current greedy iteration and the priority of active finished goods parts.

For simplicity, we describe the algorithm for the case where all finished goods are held at a single site as stated in optimization problem (1) - (3). The computation of the high quality starting solution is accomplished as follows:

Input : Holding cost budget B, holding cost bucket size C, stepping window size S,
Stepping window $(W_1, W_2, \ldots, W_S)$

Step 1: Initialization.
- *1.1. compute priority for each finished goods buffer:*
- *1.2. set* $W_p = f(p)$ *for each window slot p=1,2,…,S.*

Step 2: Greedy Allocation.
- *2.1. for each die bank buffer: provisionally increase safety stock by C and compute* $G(\mathbf{x})$
- *2.2. for each finished goods buffer in stepping window: provisionally increase safety stock by C and compute* $G(\mathbf{x})$
- *2.3. assign holding cost bucket C to current best buffer*
- *2.4. advance stepping window:*
  - *if(current best buffer ==* W_n *for some window slot n)*
    - *set* $W_p = f(p+n-1)$ *for p=1,2,…,S*
  - *else*
  - *set* $W_p = f(p+S)$ *for p=1,2,…,S*
- *2.5. update allocable budget* $B := B - C$

Step 3: Termination Condition.
- *if* $B \le 0$ *then halt; otherwise goto Step 2.*

The algorithm takes as input the overall holding cost budget *B*, a holding cost bucket size *C*, and the stepping window size, *S*. During initialization, every finished goods buffer is assigned a priority as described above (Step 1.1). Let $f(p)$ denote the finished goods buffer with priority *p*, and let W_n denote slot *n* of the stepping window. The stepping window is then initialized to contain the *S* highest priority finished goods buffers, representing the list of active finished goods parts (Step 1.2). In each iteration of the greedy algorithm, a unit inventory bucket is tentatively allocated to every die bank buffer (Step 2.1) and every active finished goods buffer (Step 2.2). The current best buffer is then determined as the one where an additional bucket of inventory results in the largest decrease of delinquency costs (Step 2.3). Given that the current best buffer is in the list of active finished goods buffers, say at position W_n, the stepping window is advanced by $n-1$ positions. If the current best buffer is in die bank, the stepping window is advanced by *S* positions (Step 2.4). Notice that in the former case, the finished goods buffer at position W_n moves to position W_1, and it thus remains a candidate for allocation in the next iteration. Once the finished goods buffer with lowest priority has been evaluated, the stepping window rolls over to the buffer with priority one. At the end of each iteration, the budget available for allocation is updated (Step 2.5). The algorithm terminates as soon as the available budget is exhausted. The resulting starting solution generated is guaranteed to be feasible, that is, the budgets allocated to the various buffers are within the given bounds and the safety stock levels are within their given ranges.

In the second phase of the optimization, the starting solution generated in the construction phase is modified until a locally optimal solution is obtained. We use the generalized reduced gradient method by Lasdon and Waren [14], a standard technique for nonlinear optimization of large sparse systems. At step *k* of the optimization, the reduced gradient method requires the functional value of the objective function $G(\mathbf{x}^k)$, and the first-order partial derivatives $\nabla G(\mathbf{x}^k)$. The functional values are returned by the method described in the next section, whereas the first-order partial derivatives are computed by finite difference approximations. The local optimization phase preserves the feasibility of the solution.

6. Approximate Computation of Backorders

For a standard periodic review inventory system, a single-echelon approximation of the expected backorders can be found in textbooks on inventory management [15]. For multi-echelon situations, an estimation of expected backorders is much more difficult. Instead of being a function of just the finished goods inventory level, the expected backorders also depend on the die bank inventory levels. We use a method that explicitly models the interdependencies between die bank and finished goods buffers, as shown in [10]. With this method, each stocking location in the network under a base-stock inventory control policy is modeled as an infinite server queue in which the number in the queueing system represents the number of outstanding replenishment orders. To analyze the network, we developed an approach based on decomposition. The idea is to analyze each stocking location individually, capturing the interactions among different stocking locations through their so-called *actual lead times*. The actual lead times consist of the standard replenishment lead times plus an additional delay due to an upstream shortage. Given the input data of the supply network and the base-stock level for every stocking location, the method computes the inventory in the network and the total expected delinquency costs. The build-to-order parts are treated like build-to-stock parts with a base-stock level equal to zero.

We model each stocking location, or buffer, by a queue with batch Poisson arrivals and infinite servers with service times following a general distribution, denoted as $M^X/G/\infty$ in queueing notation. To do so, we need to first specify the arrival and the service processes. The arrival process at each location is obtained by applying the standard MRP demand explosion technique to the production structure. The service time is the actual lead time. For buffer *j* at the die bank, the actual lead time, denoted $\tilde{L}_{0j}$, is equal to the nominal lead time, L_{0j}, from given input data. Next, let *j* correspond to a finished goods buffer at location *i*, and let *i* correspond to its single die bank supplier. We define the actual lead time of finished goods buffer *j*, $\tilde{L}_{ij}$, as

$$\tilde{L}_{ij} = \begin{cases} L_{ij} & \text{w.p.} \quad f_{0l} \\ L_{ij} + \tau_{0l} & \text{w.p.} \ (1 - f_{0l}) \end{cases} \quad \text{for } i = 1,2,\ldots,M \text{ and } j = 1,2,\ldots,M \tag{7}$$

where f_{0l} is the fill rate at die bank buffer *i*, that is, the fraction of orders filled from safety stock. From the above definition, the actual lead time at finished goods buffer *j* is equal to its nominal lead time if the order is filled immediately from the die bank. If the die bank has a stockout, which happens with probability $(1 - f_{0l})$, it is the nominal lead time, plus an addition delay τ_{0l}. This additional delay is the time required for the die bank to produce the next unit to supply the order. In our model, τ_{0l} is derived from Markov chain analysis.

The key quantity in our analysis is number of jobs, Y_{ij}, in the $M^X/G/\infty$ queue. Under a base-stock policy with base-stock levels given by x_{ij}, the safety stock inventory I_{ij} and the backorder level Q_{ij} relate to Y_{ij} through the following formulas:

$$\begin{aligned} I_{ij} &= \max\left(x_{ij} - Y_{ij}, 0\right) \\ Q_{ij} &= \max\left(Y_{ij} - x_{ij}, 0\right). \end{aligned} \tag{8}$$

The distribution of Y_{ij} can be derived from standard queuing results [16]. To increase computational efficiency, we chose to approximate Y_{ij} by a normal distribution. Thus, we only need to derive the mean and standard deviation of Y_{ij}, μ_{ij} and σ_{ij}. Then, we can express x_{ij} as $\mu_{ij} + k_{ij}\sigma_{ij}$, where k_{ij} is the so-called safety factor, and $k_{ij}\sigma_{ij}$ the safety stock.

With the arrival and service processes in place, we can analyze the queue and derive performance measures such as inventory, backorders, and fill rates. In particular, the fill rate f_{ij} is computed as

$$f_{ij} = \phi(k_{ij})\frac{\sigma_{ij}}{\mu_{ij}} + \left(1 - \Phi(k_{ij})\right) \tag{9}$$

with $\phi(.)$ and $\Phi(.)$ denoting the density function and the distribution function of the standard normal variate. The expected on-hand and backorder levels are given as follows:

$$\begin{aligned} E[I_{ij}] &= \sigma_{ij}\left(k_{ij} + G(k_{ij})\right) \\ E[Q_{ij}] &= \sigma_{ij} G(k_{ij}) \end{aligned} \tag{10}$$

where the function $G(\cdot)$ is defined as $G(y) := \int_y^\infty (z - y)\phi(z)dz$. To characterize the additional delay τ_{0l} used in the actual lead time calculation (7), the following approximation is used:

$$\tau_{0l} = L_{0l}\frac{E[Q_{0l}]}{x_{0l}\left(1 - \Phi(k_{0l})\right)} \tag{11}$$

Notice that the definition of the actual lead time at a finished goods buffer depends on the fill rate of its die bank supplier location. Thus, to evaluate the performance of the network, we start the analysis with die bank buffers, and work our way to finished goods until all buffers in the network have been analyzed. Then the expected backorders for every finished goods buffer is calculated using (10), and the total delinquency cost is determined according to (1). More technical details of the approximation are presented in [10].

7. Numerical Results

The purpose of this section is twofold. First, we want to evaluate the accuracy of our heuristic optimization procedure through numerical tests. Secondly, we want to demonstrate the benefits of the optimization in practical implementation.

To address the first issue, we defined a set of test problems, and compared the inventory allocation obtained from the heuristic optimization to the best allocation obtained by running a partial enumeration over all possible allocation policies. Comparing the solution from enumeration to the solution produced by the heuristic optimization, it is possible to evaluate the accuracy and robustness of our method. We present only a sample of an extensive experimentation study in which we applied the heuristic optimization method to a range of moderate-size test problems.

Consider a simple supply network with 2 die and 3 finished goods. The product specifications are given in Table 1. All finished goods demand is modeled using normal distributions. Notice that the demand for die bank buffers is not input data; it is computed by applying an MRP demand explosion on the product structure, taking into account the mean and standard deviation of finished goods demand.

Buffer ij	*Holding cost* h_{ij} *($/unit)*	*Shortage cost* s_{ij} *($/unit)*	*Nominal leadtime* L_{ij} *(days)*	*Weekly demand (mean)*	*Weekly demand (std.dev.)*
(0,1)	12	-	63		
(0,2)	12	-	63		
(1,1)	20	500	35	300	50
(1,2)	20	500	35	800	200
(1,3)	20	500	35	100	100

Table 1. Data for example network with 2 die and 3 finished goods parts. The production structure is defined as follows: buffer (0,1) supplies (1,1) and (1,2); buffer (0,2) supplies (1,3).

The upper and lower bounds of the safety stock inventory were set to 10 DOI and 20 DOI, respectively (a range we knew was reasonable from experimentation). Enumeration was carried out in discrete steps of 0.5 DOI over the entire range of the search space. Table 2 gives a comparison between the enumeration and the heuristic optimization procedure.

Budget B	*Enumeration #feasible*	*Enumeration* $G(\mathbf{x})$	*Optimization* $G(\mathbf{x})$	*Relative Deviation*
$48,000	57	$14,508	$15,615	7.6%
$50,000	36,682	$8,421	$8,863	5.2%
$52,000	398,212	$4,825	$5,047	4.6%
$56,000	2,656,803	$1,894	$1,962	3.6%
$60,000	4,038,748	$1,335	$1,353	1.3%

Table 2. Comparison between partial enumeration and heuristic optimization procedure for example network defined in Table 1.

The enumeration procedure evaluated more than 4 million different allocations; the number of feasible solutions found by the enumeration is listed in column "#feasible". The heuristic optimization produced good sub-optimal solutions, with the percentage deviation from optimality ranging between 1.3% and 7.6%. Next, consider a supply network with 4 die and 6 finished goods parts, and product parameters as shown in Table 3.

Buffer ij	*Holding cost* h_{ij} *($/unit)*	*Shortage cost* s_{ij} *($/unit)*	*Nominal leadtime* L_{ij} *(days)*	*Weekly demand (mean)*	*Weekly demand (std.dev.)*
(0,1)	12	-	63		
(0,2)	12	-	63		
(0,3)	12	-	45		
(0,4)	12	-	45		
(1,1)	20	500	35	300	50
(1,2)	30	1000	35	800	100
(1,3)	50	1000	35	500	100
(1,4)	20	500	30	200	200
(1,5)	40	500	30	800	200
(1,6)	50	200	30	600	200

Table 3. Data for example network with 4 die and 6 finished goods parts. The production structure is defined as follows: buffer (0,1) supplies (1,1) and (1,2); buffer (0,2) supplies (1,3); buffer (0,3) supplies (1,4); buffer (0,4) supplies (1,5) and (1,6).

In this example, the enumeration evaluated over 60 million configurations, resulting in computation times well above 6 hours on a 300 MHz IBM Power PC. The heuristic optimization algorithm, in contrast, produced sub-optimal solutions in less than one second. As shown in Table 4, the delinquency costs obtained from our method were within 2.2% of optimality for all scenarios tested. The results shown here are representative of results that were generally obtained by comparing our method to results from enumeration for other test problems of varying size.

Budget B	*Enumeration* #feasible	*Enumeration* $G(\mathbf{x})$	*Optimization* $G(\mathbf{x})$	*Relative Deviation*
$195,000	*194,982*	*$26,050*	*$26,454*	*1.6%*
$200,000	*36,308,000*	*$21,306*	*$21,778*	*2.2%*
$205,000	*60,449,819*	*$19,402*	*$19,439*	*0.2%*

Table 4. Comparison between partial enumeration and heuristic optimization procedure for the example network defined in Table 3.

In order to project improvements that can be achieved with the new system, we compared the optimized policy with that of the original policy implemented at Xilinx in 1996 when the inventory improvement process began. The 1996 policy is well-defined: all finished goods part inventories (for parts that were build-to-stock) had the same target level *x* DOI, and each of the die bank inventories had the same target level of *y* DOI (although the 1996 policy is no longer used and the new policy is considerably different, Xilinx has requested that we not reveal the exact targets in the old policy). We used a full data set from 1998 in order to compare the 1996 policy and the optimized policy. The data set consisted of 104 die bank parts, 314 finished goods parts held at the Xilinx facility in Ireland, and 1194 parts held at the facility in San Jose. Comparisons between the two policies were made in two different ways. First, the total inventory holding cost budget was held at the same level for both policies, and the projected improvement in delinquency cost was found. Thus, the inventory holding cost that resulted from the 1996 policy was calculated and used as the holding cost budget constraint for the inventory optimization. As illustrated in Figure 4, the optimization produced a 62.5% reduction in delinquency cost for the build-to-stock parts, while the delinquency cost for the build-to-order parts remained essentially the same. The huge improvement was primarily due to reallocation of the inventory among the various finished goods parts. Targets of stable finished goods parts with low delinquency costs and high holding costs were set lower, while targets were set higher for less stable parts with higher delinquency parts.

FIGURE 4 ABOUT HERE

Second, the total delinquency cost was held constant for both policies and the projected improvement in inventory holding cost budget was found. Thus, the total delinquency cost for the data set was found using the 1996 policy. The inventory optimization was run with a number of different settings of the inventory holding budget until the budget was found that yielded the same delinquency cost. The optimized policy was found to yield a policy with a 19.9% reduction in overall inventory holding cost as illustrated in Figure 5. Since work-in-process inventory is not affected by the inventory policy, being purely a function of the lead time and demand rate, we considered reduction in safety stock to get a better picture of the degree of improvement due to the model. Considering safety stock inventory only, we find a 51.2% reduction in inventory holding cost budget. The reduction was achieved by determining an improved mix of inventory between die parts, devices, and between echelons.

FIGURE 5 ABOUT HERE

We should note that we expect the projected improvements to overestimate results that have been and will be achieved. First, the model optimizes based on holding and delinquency cost data. Although Xilinx has spent considerable time developing a methodology for this data and updating it regularly, these costs have some degree of uncertainty. Second, during its implementation and initial usage, the system is being controlled tightly, using tight minimum and maximum levels for various parts' inventory levels. Thus, the inventory system has little room with which to optimize. As we have noticed in other inventory model implementations at Xilinx, users will begin to relax the restrictions as they view the improved results and trust the system. Third, the inventory optimization model, as with all multi-echelon models of real-life situations, relies on simplifying assumptions. Xilinx is currently in the process of developing a simulation model of their supply chain, one purpose of which is to confirm or understand the impacts of currently unaccounted complexities on the inventory targets.

8. Conclusion

A few insights into Xilinx's inventory system have been gained from using this system. First, the 1996 inventory target for die bank was far too high for an overall target inventory level. The ability to replenish the pipeline to finished goods (i.e., the fill rate of parts held at the die bank) is an increasing but concave function of the die bank inventory target. A large fraction of the parts had inventory targets set at a level past the point of diminishing returns, and the target inventory level could be dropped significantly with a very marginal impact on the die fill rate. Xilinx has begun reducing die bank inventories based on preliminary modeling, lowering many of the identified die bank buffers to as much as half of their original level. Second, the old adage "it's not that we have too little inventory, we just don't have the right inventory" has certainly been shown to be true for Xilinx's case. Inventory target levels should be determined individually for different parts, depending on the cost and demand uncertainty characteristics. For this situation, we have found that the primary benefit comes from proper mix determination in finished goods.

FIGURE 6 ABOUT HERE

In Figure 6, we present the actual corporate wide inventory levels over the life of the inventory modeling project initiated in 1996. The figure shows a significant drop from 120 days of inventory in late 1996 to 68 days at the end of the first quarter of 1999. We believe that these improvements are partly due to the new inventory management system. However, other initiatives and external factors such as overall market demand, new product family introductions, and manufacturing improvements affect the overall inventory level. We believe that much of the improvement has come from the process of developing and implementing the system. The process involved incremental steps that yielded significant insights gained either from running spreadsheet models or prototypes of this system. The insights were used to adjust inventory policies and improve performance.

References

1. Brown, A., Lee, H., and Petrakian, R. (1998) Achieving Supply Chain Excellence Through Product and Process Postponement in the Semiconductor Industry: The Xilinx Case. Working paper, Vanderbilt University.

2. Lee, H. (1996) Effective Inventory and Service Management Through Product and Process Redesign, *Operations Research*, 44 (1), 151-159.

3. Axsaeter, S. (1993) Continuous review policies. In Logistics of Production and Inventory. S. Graves, A. Rinooy Kan, and P. Zipkin (eds.), North Holland.

4. Federgruen, A. (1993) Centralized Planning Models for Multi-Echelon Inventory Systems Under Uncertainty. In Logistics of Production and Inventory. S. Graves, A. Rinooy Kan, and P. Zipkin (eds.), North Holland.

5. Sherbrooke, C. (1968) METRIC: A Multi-Echelon Technique for Recoverable Item Control. *Operations Research*, 16, 122 - 141.

6. Sherbrooke, C. (1986) VARI-METRIC: Improved Approximations for Multi-Indenture, Multi-Echelon Availability Models. *Operations Research*, 34, 311 - 319.

7. Cohen, M., Kamesam, P.V., Kleindorfer, P., Lee, H., Tekerian, A. (1990) Optimizer: IBM's Multi-Echelon Inventory System for Managing Service Logistics, *Interfaces*, 20 (1), 65-82.

8. Lee, H., and Billington, C. (1993) Material Management in Decentralized Supply Chains. *Operations Research*, 41, 835 – 847.

9. Graves, S., Kletter, D., and Hetzel, W. (1998) A Dynamic Model for Requirements Planning with Application to Supply Chain Optimization. *Operations Research*, 46, S35 - S49.

10. Ettl, M., G. Feigin, G. Lin, and D. Yao. (1998) A Supply Network Model with Base-Stock Control and Service Requirements. To appear in *Operations Research*.

11. Feigin, G. (1998) Inventory Planning in Large Assembly Supply Chains. In Quantitative Models for Supply Chain Management. S. Tayur, R. Ganeshan, and M. Magazine (eds.), Kluver Academic Publishers.

12. Arntzen, B.C., Brown, G.G., Harrison, T.P., and Trafton, L.L. (1995) Global Supply Chain Management at Digital Equipment Corporation. *Interfaces*, 25, 69-93.

13. Andersson, J., Axsaeter, S., and Marklund, J. (1998) Decentralized Multi-Echelon Inventory Control. To appear in Production and Inventory Management.

14. Lasdon, L.S. and Waren, A.D (1979) Generalized Reduced Gradient Software for Linearly and Nonlinearly Constrained Problems. In *Design and Implementation of Optimization Software.* Greenberg, H. (ed.), Sijthoff and Noordhoff.

15. Nahmias, S. (1993) Production and Operations Analysis. 2nd ed. Irwin.

16. Liu, L., Kashyap, B.R.K., and Templeton, J.G.C. (1990), On the $GI^X/G/\infty$ System. *Journal of Applied Probability*, 27, 671 - 683.

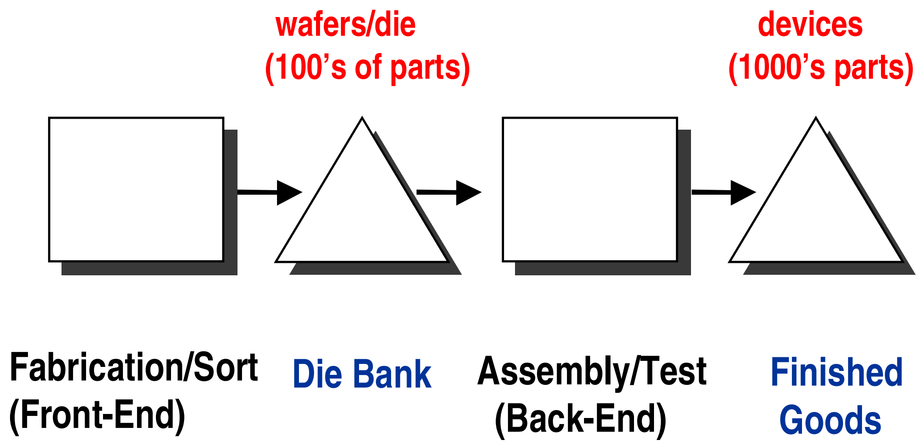

Figure 1: The Xilinx manufacturing process. We note that there is actually a second intermediate inventory buffer (Bin) that holds inventory for some parts followed by a marking step (between Assembly/Test and Finished Goods). However, for the purposes of inventory modeling, these steps have been ignored.

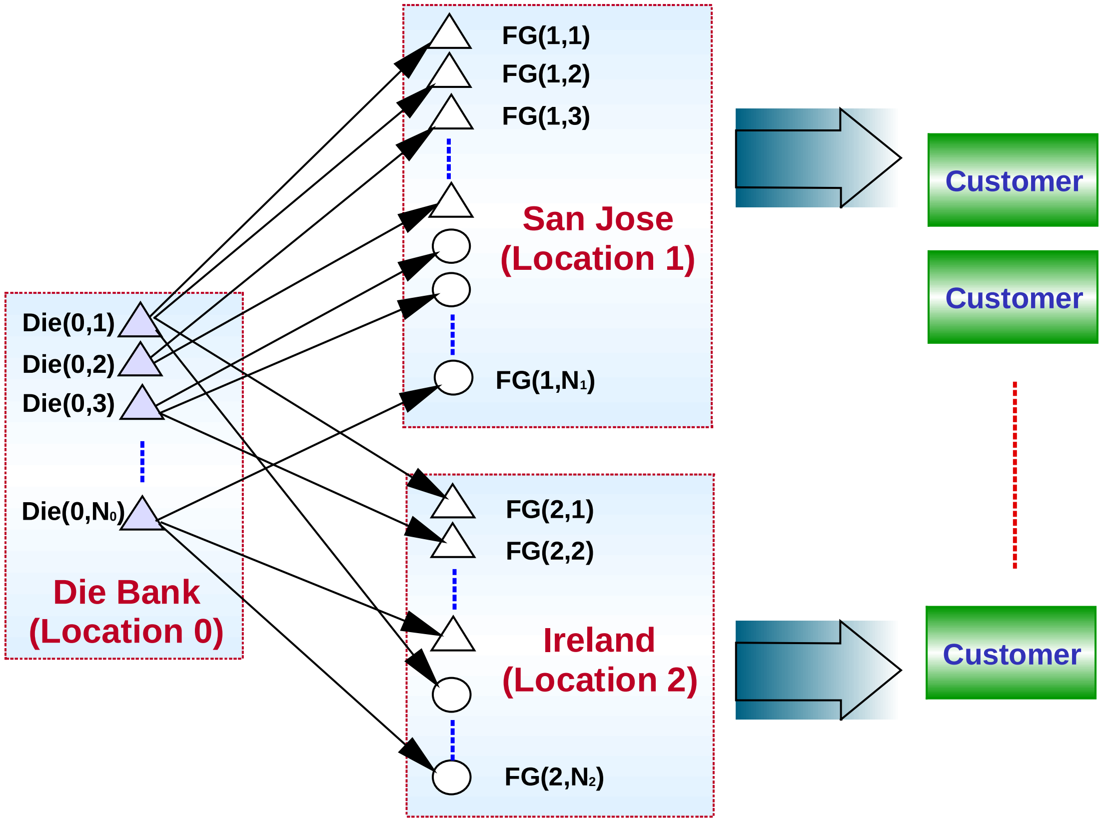

Figure 2: The Xilinx product and inventory structure. Each die type can be differentiated into about 100 different finished goods. Finished goods inventory is held at two locations San Jose and Ireland.

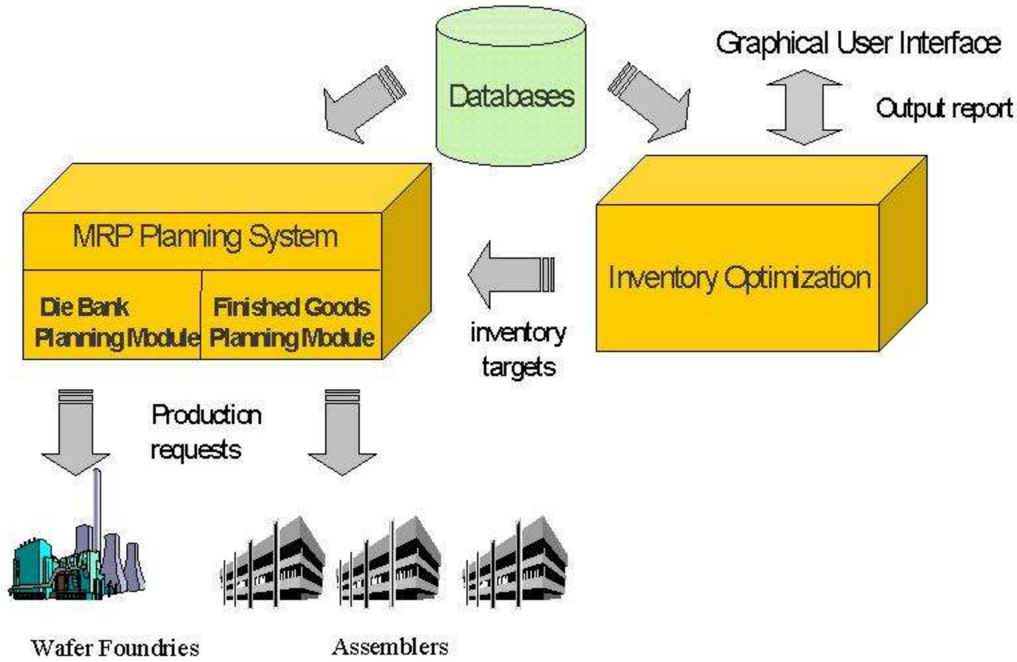

Figure 3: The integration of the inventory optimization tool within the Xilinx production planning system.

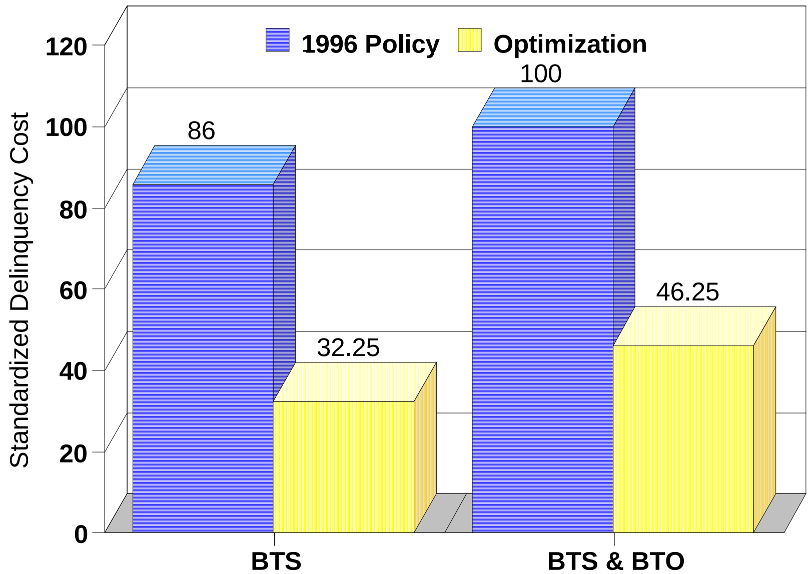

Figure 4: The delinquency cost was projected to decrease by over 50 percent using inventory optimization. The inventory holding cost budget was held constant in both scenarios. The vertical axis is standardized (for confidentiality purposes) with respect to the total delinquency cost associated with the inventory policy implemented at Xilinx in 1996. The reduction of delinquency cost is mainly achieved through optimizing the target safety stock levels of build-to-stock (BTS) parts. The product set consisted of 104 die and 1,508 finished goods parts.

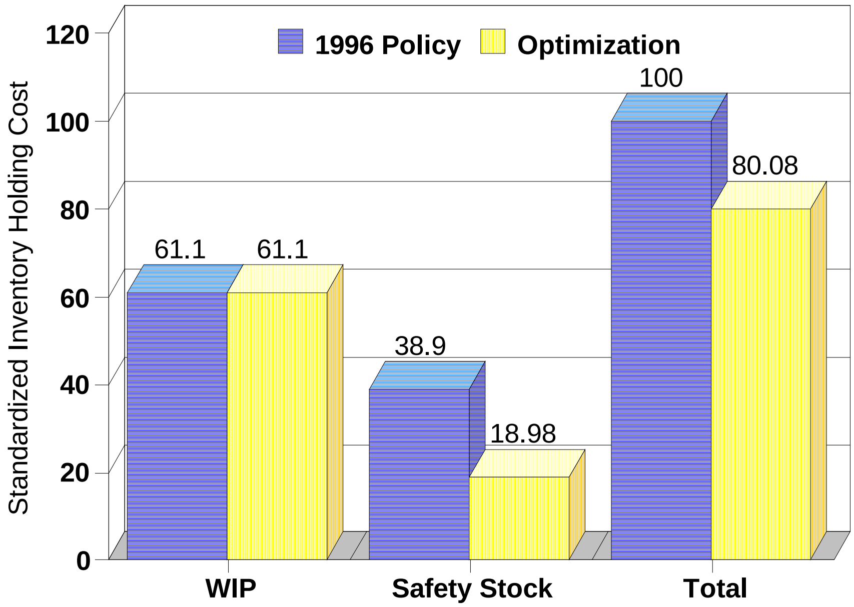

Figure 5: The holding cost for safety stock decreases by over 50 percent using inventory optimization. The delinquency cost was held constant in both scenarios. The vertical axis is standardized (for confidentiality reasons) with respect to the total inventory holding cost associated with the inventory policy implemented at Xilinx in 1996. The product set consisted of 104 die and 1,508 finished goods parts.

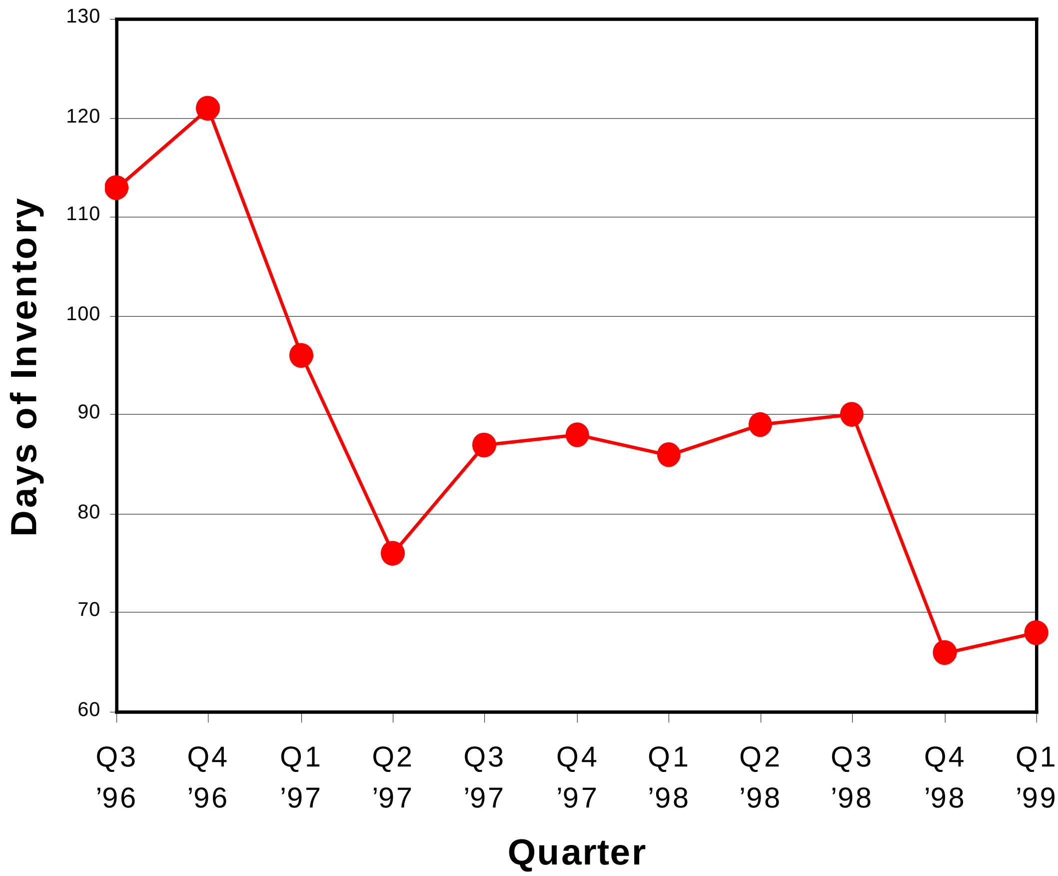

Figure 6: The actual corporate-wide inventory level over the life of the inventory modeling initiative undertaken at Xilinx.