European University Institute
Economics Department

EUI Working Paper **ECO** No. 97/35

Managing Design Complexity to Improve on Cost, Quality, Variety, and Time–to–Market Performance Variables

Spyros Vassilakis

Part A — pp. 1-29

Printed in Italy in December 1997
European University Institute
Badia Fiesolana
I – 50016 San Domenico (FI)
Italy

Managing Design Complexity to Improve on Cost, Quality, Variety, and Time–to–Market Performance Variables

Spyros Vassilakis

Department of Economics
European University Institute
Badia Fiesolana
I–50016 S. Domenico di Fiesole (FI)
Italy
Tel.: +39–55–46.85.305
Fax: +39–55–46.85.202
E-mail: vassilak@datacomm.iue.it

July 1997

Abstract

This paper contains a model of waste elimination through design. It argues for the importance of managing design complexity in improving cost, quality, variety, and time–to–market performance variables. Management of design complexity is identified with creation, choice, and application of design problem representations, divisions of design labor, and product architectures that provably eliminate waste. The paper's thesis is illustrated with a comparison of Toyota's technology strategy (based on waste elimination) to that of General Motors (based on frontier–shifting investment).

JEL classification numbers: C69, D20, L23, O32, M19.
Keywords: Cost, quality, variety, time–to–market, design complexity.

1 Introduction

The economic theory of technological change is a theory of investment subject to appropriability problems. At any point in time, there is a feasible set of values of cost, quality and variety variables. Each firm is on the frontier of its feasible set. The only way to simultaneously improve in all dimensions is investment. A firm that buys a flexible manufacturing system (FMS) and trains workers in its use, for example, has invested in equipment and training that allow it greater variety with the same cost and quality as before. The FMS itself was invented by a firm that invested in research. Investment is modelled as foregone consumption used as an input into a black–box process; the output of this process is a larger feasible set. Scale is important because of nonconvexities: in particular some investments are fixed costs to be spread over as many units as possible. Appropriability problems arise because of imperfect competition, externalities, or asymmetric information.

Some recent literature has proposed looking into the black–box process. Solow (1994, p. 52) suggested that "... the production of new technology may not be a simple matter of inputs and outputs. I do not doubt that high financial returns to successful innovation will direct resources into R&D. The hard part is to model what happens then!" Milgrom and Roberts (1992, p. 93) argue that not all resource allocation problems are the same: problems with design attributes require different coordination mechanisms. And Hayek (1948, p. 196) remarks on the treatment of cost curves as objectively given data: "What is forgotten is that the method which under given conditions is the cheapest is a thing which has to be discovered, and to be discovered anew, sometimes almost from day to day, by the entrepreneur, and that, in spite of the strong inducement, it is by no means regularly the established entrepreneurs, the man in charge of the existing plant, who will discover what is the best method." I will summarize this literature to motivate the mathematical model introduced in the main body of the paper.

The first point this literature makes is that existing arrangements do not usually exhaust the possibilities afforded by current equipment, knowledge and people. Large improvements can be obtained by discovering and eliminating waste. Hammer and Champy (1993, p. 37) describe how IBM Credit reduced its response time to a credit application from six days to four hours. The first step was the discovery of waste: "Two senior managers at IBM Credit took a financing request and ... asked personnel in each office to put aside whatever they were doing and to process this request as they normally would, only without the delay of having it sit in a pile ... performing the actual work took in total only ninety minutes. The remainder — now more than seven days on the average — was

consumed by handing the form off from one department to the next." The second step consisted in understanding the relative importance of investment vs. waste elimination. A new computer system might be able to "double the personal productivity of each individual, but total turnaround time would have been reduced by only 45 minutes" (ibid., p. 38). The reason is that waste had not yet been eliminated; the new computer system "would have done nothing to eradicate the queue time that awaited the forms when they arrived at each office" (ibid., p. 84). The third step was to identify the source of waste: "every request (was handled as if) it was unique and difficult to process, thereby requiring the intervention of four highly paid specialists. In fact, most of the work those specialists did was clerical ... and well within the capacity of a single individual when he is supported by a computer system." The fourth step was the installation of the computer system. The fifth step was the routing of difficult cases to a team of specialists. The result was that IBM Credit reduced turnaround time from seven days to four hours; increased the number of cases handled one hundred times (not 100%); and reduced the number of employees involved (ibid., p. 39). The hundredfold increase in productivity could be attributed to investing in a new computer system. The reasoning in step two shows why this would be a mistake; the critical step in increasing productivity was the classification of cases into routine and hard, and their different handling. The next point illustrates how costly such a mistake can be.

The second point the literature makes is that misdiagnosis of a waste elimination problem for a lack of investment problem is both possible and costly. A well–known example is the attempt of General Motors (GM) to approach the industry leader, Toyota, in the 1980s. In 1980, Toyota could build a car for $1500 less than GM; in small cars, the difference was $2874 (Keller, 1990, pp. 82, 83). GM had a wide product range, but its models were considered by consumers and dealers as only cosmetically different, "victims of badge engineering – changing the nameplate and a few decorative features"(ibid., p. 72). Finally, GM had a reputation for low–quality, defect–ridden products: "by the Summer of 1981, GM was forced to recall all of its 1980 standard transmission X–cars (about 245,000 cars) to fix clutch and rear–brake systems. At Cadillac, the V–8–6–4 engine was fraught with mechanical problems. It followed the diesel engine, also a disaster, and caused massive defections from the Cadillac brand. In 1981, the J.D. Power survey ranked Cadillac number fifteen out of twenty–two brands" (ibid., pp. 74 and 76). At the same time, the competition was doing better: "... consumers now expected good performance and high quality to be standard features on their cars. The Japanese had taught them to demand that" (ibid., p. 69). As a result, "... in 1980, GM posted its first loss in sixty years — a sum of $763 million" (ibid., p. 69).

GM, and all the Big Three, believed they faced cost–quality–variety tradeoffs, i.e. that they were on the frontier of their feasible set. Womack et al. (1990, p. 65) report that "most Western companies concluded that the Japanese succeeded because they produced standardized products in ultra–high volume. As recently as 1987 a manager in Detroit confided in an interview with members of our project that ...[the Japanese]... are making identical tin cans; if I did that I could have high quality and low cost too." This belief was based on experience. One of the most striking findings of Womack et al. (1990, pp. 93, 98) was that there were no significant relations between plant cost, quality, and variety. Once, however, Japanese plants were removed from the sample, tradeoffs appeared. An example of action based on this belief is reported in Ingrassia and White (1994, p. 167). "Reuss [then a top GM executive] argue that to achieve high quality GM should dedicate three of the four GM–10 factories to building just one model each. One model, fewer variations, fewer chances for the assembly workers to screw up, the argument went. Thus the GM–10 factory in Doraville, Georgia, got the Cutlass Supreme, the new plant in Fairfax, Kansas, got the Pontiac Prix, and the Oshawa, Ontario, plant got the Chevrolet Lumina. It was another gigantic error." The diagnosis for a firm that believes it is on its efficiency frontier but lags the competition is clear: lack of investment. GM invested $70 billion during the 1980s (ibid., p. 33). GM's CFO Alan Smith, as quoted in Keller (1990, p. 196), provides some perspective on the magnitude of this sum: "From 1980 to 1985, GM spent $45 billion in capital investment, yet increased its worldwide market share by only one percentage point, to 22 percent. For the same amount of money GM could buy Toyota and Nissan outright, instantly increasing its market share to 40 percent."

The results of the $70 billion investment were not those anticipated by GM. Its US market share decreased continuously from 46 percent in 1980 to 30 percent in 1996. Average pretax return on assets in the 1982–1991 period was 2.8% for GM and 4.8% for its suppliers; the corresponding figures for Toyota were 13% and 7%, respectively (Dyer, 1994, p. 178). GM loses money in passenger cars: "The Big Three have been able to raise car prices only by 6% a year since 1988, while manufacturing costs have been rising at an annual rate of 6.5%. ... The magnitude of the profit drain from cars is difficult to calculate because it depends on the allocation of corporate overhead, but it's huge. No US company has made a profit on cars once during the last decade. Losses have to be in the tens of billions of dollars" (Taylor, 1996, p. 14). GM makes money in trucks in North America, and is also profitable in Europe; until very recently, though, it has not faced Japanese competition in either of these segments. Ingrassia and White (1994, p. 353) state that, in the protected European market, "GM Europe could charge hundreds or even thousands of dollars more for its vehicles than GM could have demanded for the

same cars in the US. ... GM Europe would lose money, too, if it had to sell at North American prices." Its protected position withstanding, GM Europe in 1994 generated $188,278 in revenues per employee, vs. Toyota's $939,233 (*Automotive Industries* 1995, p. 48). Taylor (1992, p. 64) explains the situation in trucks: "GM and Ford control the full–size pickup market and price their vehicles as any duopoly would. The Japanese aren't strong competitors, and a 25% import duty on two–door light trucks puts them at a severe disadvantage." *The Economist* (1997) reports that the import duty is still in place in 1997.

The $70 billion investment did not improve GM's relative cost position. Ingrassia and White (1994, p. 33) report that "... when the 1980s began, GM had the lowest production costs among the Big Three. By the middle part of the decade, GM had the highest cost of any major automaker in the world." Taylor (1997 (a), p. 62) reports that this was still true in 1996: "[GM] makes almost no money in North America, where it has higher costs than its competitors and some of the weakest brands."

The $70 billion investment did not deliver the expected quality improvements. GM's quality problems were probably the main reason for the decline in its market share. Its warranty costs in 1985 were $2 billion, or $300 per vehicle (Ingrassia and White, 1994, p. 931). Between 1985 and 1991, it processed 2.7 million warranty claims for a single problem (stalling) in Buick and Oldsmobile models; they were recalled in 1991 (ibid., pp. 98–99). The Fiero model was recalled twice, in 1987 and 1989, to fix problems that caused fires (ibid., pp. 108, 110). The extent of the damage to GM's reputation is indicated by a 1989 customer survey that found that "GM cars got fewer recommendations on average than any brand except the Yugo, a comical subcompact that had become the industry's benchmark for bad quality" (ibid., p. 181). Internal GM studies (ibid., p. 427) showed that in 1992 GM attained the 1986 quality levels of an average Japanese car; its trucks had the highest number of defects of all brands in the market; its warranty costs in 1992 were $3 billion, or $829 per car.

The $70 billion investment did not deliver the expected results in variety and model renewal either. Many of GM's large number of models were still perceived as only cosmetically different; and the new models as not being significantly better than the ones they replaced. An example is described in Taylor (1992, p. 78): "Buyers not only couldn't distinguish an Oldsmobile from a Buick but also had a hard time telling a $9,000 Pontiac Grand Am from a $25,000 Cadillac Eldorado. ... This luxury car fiasco cost GM $1 billion in 1986 alone." Ford was able to inflict further damage to GM's reputation with negative advertising that satirized look–alike cars (ibid., p. 134). The GM–10 cars, that were being designed for 8 years (1982–90), were perceived by

consumers as worse than the models they were replacing: "... these cars had lost GM $7 billion. They had generated nothing but huge losses, steady market–share erosion, and a belief among consumers that GM didn't know how to design good cars" (Ingrassia and White, p. 431). Womack et al. (1990, p. 109) report that "... the A bodies that GM–10 was to replace have proved much more profitable in the late 1980s, and the company now plans to continue the production of the Oldsmobile and Buick variant indefinitely." This was still true as late as 1997: "The company only recently retired the midsize Buick Century and Olds Cipra, which dated to 1982, and it still sells the compact Buick Skylark and Olds Achieva" (Taylor, 1997 (a), p. 65). Five years after GM's new management took over and started implementing waste elimination ideas, GM "has improved its cost structure, streamlined product development, and improved its image, but it still lags behind the industry" (ibid., p. 62). Toyota does have more assets per employee, $144,189 vs. GM's $37,559 in 1991 (Williams et al., 1994, table 3.3, p. 35). This does not explain its superior performance. As Womack et al. (1990, p. 236) state, extra investment rationalizations "did not explain why Japanese firms gained major benefits from automation while Western firms often seemed to spend more than they saved."

The third point the literature makes is that waste elimination is the outcome of product and process design. Product design eliminates waste in the form of redundant components; process design eliminates waste in the form of redundant processing steps. Low cost, high quality and large variety have to be designed — in the products and processes of the firm; they are not straightforward consequences of investment (hence GM's unexpected experience). To understand this, consider a product as a device that receives inputs from the environment and the user, and produces outputs that constitute the user experience. Clark and Fujimoto (1991, p. 5) illustrate this point using a car driver as an example: "Seated behind the wheel, the customer receives a barrage of messages about the vehicle's performance. Some of these messages are delivered directly by the car: the feel of acceleration, the responsiveness of the steering system, the noise of the engine, the heft of the door. ... All these messages influence the customer's evaluation. ... In essence, the customer is consuming the product experience, not the physical product itself." The behavior of the product, then, is the set of input–output pairs it allows.

Each such pair represents a function the product has to fulfil. Design starts from desired behavior. Each function is mapped into a physical component that realizes it. Product size, mass, ease of fabrication, and therefore cost, depend on the number and the individual characteristics of components in the product. Cost reduction at the

product design stage is achieved by mapping as many functions as possible into individual components, i.e. by eliminating waste in the form of redundant components. This is called function sharing by Ulrich (1995, p. 433), who also provides the following example: "A conventional motorcycle contains a steel tubular frame distinct from the engine and transmission. In contrast, several high–performance motorcycles contain no distinct frame. Rather the cast aluminum transmission and motor casing acts as the structure for the motorcycle. The motorcycle designers adopted function sharing as a means of exploiting the fact that the transmission and motor case had incidental structural properties which were redundant to the structural properties of the conventional frame".

Product design is probably the major determinant of cost for many products. Whitney (1988) reports on GM and Rolls–Royce studies to this effect. He then states: "When senior managers put most of their efforts into analyzing current production rather than product design, they are monitoring what accounts for only about a third of total manufacturing costs ... they now face competition that is reducing drastically the number of components and subassemblies for products and achieving a 50% or more reduction in direct cost of manufacture." Morton (1994, p. 11) stresses the importance of design for other performance variables: "Good design is the key to manufacturing. It is the difference between a product that does a great job reliably, is easily fixed if damaged and is made cheaply and quickly; and a pile of junk."

GM neglected cost reduction at the product design stage. Ingrassia and White (1994, p. 112) describe Hamtramck, a GM factory that opened in 1985: "Hamtramck was supposed to erase the nearly $2000 a car cost advantage Toyota enjoyed ... (Hamtramck's) cars were hard to build. The front and rear bumper of a Cadillac Seville had more than 460 parts and took thirty–three minutes of labor to put together. Two years after it opened, Hamtramck put a stunning 100 hours of labor — five times as much as Toyota — into building each car." The GM–10 cars were more difficult to build than comparable Ford Taurus cars, because they had so many parts they were difficult to assemble: "As GM engineers tore apart Tauruses and gathered intelligence about the factories in Atlanta and Chicago where they were built, they realized their crosstown rival had designed cars that were cheaper to build. Building a Grand Prix was like assembling a jigsaw puzzle. It required some thirty–five hours of assembly labor. Building a Taurus took about twenty hours of assembly labor. In 1988, this translated to a roughly $300 per car advantage to Ford — on assembly labor alone. Ford's advantage was all the more stunning because GM had spent billions to outfit the GM–10 plants with the latest automation. Eventually GM would lose as much as $1800 on every GM–10 car it sold" (ibid., pp. 160–161). During the redesign of its J–cars in 1989–90, GM engineers

studied Toyota and Honda cars: "the more the J–car engineers learned, the harder they worked to eliminate extraneous parts and make the new mold easier to build. The old Cavalier took nearly 50 percent more labor time to assemble than a Corolla. The new J–car would have nearly 20 percent fewer bolts and widgets than the old model" (ibid., p. 423). This paper discusses design for cost reduction in Sections 3, 4, and 5.

Product variety can be obtained in several distinct ways. The first way is to design each product separately; no provision is made for parts–sharing between products. A firm can then either build its product range before orders come in, and provide fast service at high (inventory) cost; or it can build to order, providing slow service at lower (inventory) cost. The second way, modular product design, allows the firm to avoid these two extremes. All the products are designed together. Waste elimination in the form of minimizing the number of components is then equivalent to maximizing the number of shared parts; build shared parts before orders come in; and then assemble to order. Feitzinger and Lee (1997) report that this is done at Hewlett–Packard; and Whitney (1988; 1995) at Nippondenso, one of Toyota's main suppliers. The same principle applied to equipment, modular equipment design, allows a given set of machines to make different products, while minimizing setup and changeover costs. Shingo (1989) calls this principle function standardization, and provides numerous examples, many of them drawn from Toyota. The same author, in his study of Toyota, warns: "Mechanization should be considered only after every effort has been made to improve setups using the techniques described. [They] can reduce a two–hours setup to three minutes, and mechanization will probably reduce that time only by another minute" (Shingo, 1989, p. 44).

GM neglected design for variety, and Shingo's advice on automation. It seems to have reversed the principle of modular product design in two ways. First, similar cars used different components, foregoing the benefits of eliminating redundancies: "... for years the company produced 17 ignition systems where three would have sufficed, and 40 types of catalytic converters instead of three or four. The engineering was 180 degrees out of phase. GM cars looked alike outside but were all different inside" (Taylor, 1992, p. 59). Secondly, in other instances GM compromised variety in order to share parts (as opposed to maximizing shared parts keeping variety fixed), foregoing the extra revenue variety brings: "GM's aggressive pursuit of commonality of floor panels and other body parts as a way of holding down the enormous cost of developing a series of fuel–efficient models during the 1970s and 1980s seriously hurt its product differentiation" (Clark and Fujimoto, 1991, p. 149).

GM seems to have reversed the principle of modular equipment design as well: "As recently as the 1980s stamping was split among several divisions that each produced

their own metal parts using different presses and dies. This segmented approach resulted in some press systems that ran only 20 hours a week. Pieces stamped by Pontiac for a certain model wouldn't fit on a nearly identical car made by Buick. GM is now spending $850 million to standardize the die production at 13 plants and reduce the number of press–line setups from 57 to six. That's progress, though it won't send GM to the head of the class. GM will spend about $2700 per ton of stamping, vs. $2200 to $2300 at Toyota" (Taylor, 1997 (a), p. 62). The same source reports on GM's recent effort to adopt modular product design: "GM is thinking of ways to integrate future Chevies, Pontiacs and Saturns with German–made Opels to achieve greater economies of scale. ... a compact Pontiac Sunfire is designed alongside four other brands as part of a global small–car program" (ibid., p. 65). This paper discusses design for inexpensive variety in Section 7.

Quality, in the sense of a defect–free product, can be attained in several distinct ways. The first is to build redundant components into the product, so that if one component is defective, others will perform its function. It makes the product heavy and expensive to build. The alternative is to eliminate waste from the product in the form of redundant components, so that testing and fault–diagnosis become easier. (The same principle applied to production is the well–known lean technique of removing inventory to expose defects.) An example is provided by De Micheli (1993, p. 33): "Microelectronic circuits are tested after manufacturing to screen fabrication errors. Circuit testability affects its quality. A circuit that is not fully testable is less valuable than another one that is. For some kind of fault models, increasing the fault coverage is related to removing redundancies in the circuit." In the sense, then, that quality is a byproduct of design for cost reduction, quality is free. GM's quality strategy, on the other hand, was not based on waste elimination and was bound to produce cost–quality tradeoffs. Alex Mair, then head of GM's Vehicle Assessment center, made this point to GM engineers in a speech given in 1986, quoted by Ingrassia and White (pp. 89-93). Mair first reminded the audience that GM had invented the automatic transmission in the 1930s. He then played tapes of a GM Hydramatic transmission (noisy); and a Toyota Camry gearbox (quiet). Finally, he made the point: "People say, 'I don't believe that. I just drove a Cadillac and it was very quiet.' That's true. Because we spend a lot of money and a lot of engineering talent to mask that noise — by packing insulation under the hood" (ibid., p. 92). This paper discusses design for inexpensive quality in Section 6.

The fourth point the literature makes is that product design is an information–processing activity; its cost and duration depend on product complexity. This view informs the studies of design undertaken by Clark and Fujimoto (1991) and Simon (1981).

The former, for example, state that "Throughout the book we look at the development process as a total information system and identify important problems from the perspective of information processing" (p. 18). They also have separate chapters on the management of complexity (ch. 6) and on problem–solving (ch. 8). Complexity of design can be the binding constraint on a firm's ability to improve. Lengauer (1990, p. 938) in his survey of VLSI theory, states: "It is a generally accepted fact that the design problem dominates the fabrication problem. Put differently, the fabrication technology provides us with means to produce circuits that are so complex that we do not know how to design them effectively. This is the reason why circuit design is one of the most critical areas in computer science today." Design was also one of the binding constraints that resulted in GM's market share loss in the 1980s: "In 1987, GM's share of car sales skidded all the way down to 36.6%, a drop of nearly five points in a single year. Unlike Chrysler and Ford, wealthy GM had redesigned almost its entire car line in the early 1980s. It had changed four rear–wheel drive to front–wheel drive to reduce weight and improve fuel economy. In the rush to get the redesigned cars to market, GM neglected to remove all the bugs. The new models weren't nearly as good as those they replaced. ... GM had taken on too much — and done it badly. Switching the drive wheels of a car from the rear to the front requires entirely new mechanical systems and changes the dynamics of the car. Once the project got under way, corporate momentum demanded that it be completed on time" (Taylor, 1992, pp. 76, 77). Design was also the binding constraint on the timely completion of the GM–10 program; when the new models were ready, demand was no longer there: "it would be the largest new model program ever, the ultimate expression of GM's ability to capitalize on its enormous economies of scale. But GM couldn't pull it off. The world's largest corporaton choked. ... Eight years after the project began, the final GM–10 car came to the market in 1990 — but the market had moved. James Womack calls GM-10 the biggest catastrophe in American industrial history" (ibid., p. 731). It is complexity that prevents a firm from taking full advantage of the opportunities afforded by its equipment, knowledge, and people, i.e. from eliminating waste.

The fifth, and probably most important, point the literature makes is that complexity can be managed; and that the way it is managed matters for the cost and timeliness of design, and hence for the extent of waste elimination. Toyota's and Honda's ability, despite their smaller scale, to offer lower cost, higher quality, more variety and faster model replacement than the Big Three, led to studies of their design processes. The surprising result was that their advantage was not due to greater engineering effort: "a totally new Japanese car required 1.7 million hours of engineering effort on average and took forty–six months from first design to customer deliveries. By contrast, the average

US and European projects of comparable complexity and with the same fraction of carryover and shared parts took 3 million engineering hours and consumed sixty months. This, then, is the true magnitude of the performance difference between lean and mass production: nearly a two–to–one difference in engineering effort and a saving of one–third in development time" (Womack et al., 1990, p. 111). These firms had developed distinct ways of information processing and problem–solving. Clark and Fujimoto (1991) describe concurrent engineering; Hauser and Clausing (1988) quality function deployment; and Ward et al. (1995) concurrent set–based design. The Big Three understood the design advantages of these rivals and based their turnaround efforts on changing their own design methods. Naughton (1995, p. 58) comments on the appointment of J. Nasser as head of Ford's product development: "it puts him on the spot to fix Ford's biggest problem: spending too much money and time to bring out new cars. ... By 1999 Nasser expects to cut development time from 37 months to 24 months, equal to industry leader Toyota. Key steps will be reducing cars' complexity and eliminating redundant parts." Taylor (1997 (a), pp. 61, 65) describes his visit to a GM site where new product designs are kept: "No place is more vital than Rigorous Tracking Room. It contains a wall chart 45 feet long that plots 42 new vehicle programs — the very lifeblood of GM — through their three–year gestation. The programs are measured for timeliness, quality, and financial performance, and color–coded by complexity. ... It is one thing to design a car [AN: here "design" is used to mean "specify"] and quite another to engineer it so that customers get it quickly. A chart in the Rigorous Tracking Room shows GM's progress. In 1992 the company needed 42 months to start producton on a new model once the final design had been set, vs. 31 months for Toyota. Now GM can do it all in 31 months, Toyota in 26." Finally, Ingrassia and White (1994, ch. 19) document how Chrysler studied the design methods of Honda and Mitsubishi, and then adapted them to achieve significant gains in the cost and speed of model replacement.

2 Design Problems

Design aims at achieving desired product behavior at reasonable cost by eliminating waste. Several ways of doing this are presented in this paper; they all provably eliminate waste, but differ in the time and effort they need to do so. By comparing them, our attention is directed towards the specific activities that make the difference. In this particular case, our attention is drawn to the many, rather unexpected ways problem representation, division of design labor, and product architecture matter for the timely elimination of waste. This is done in the simplest model of design that does some justice

to its complexity.

Let $D = \{0, 1\}$ be a two–element set, and D^n its n–fold Cartesian product. The Introduction motivates the definition of product behavior as a function f mapping D^n (the inputs) into D^m (the outputs). Behavior of electronic circuits, and of other products after coding their inputs and outputs as zero–one strings, can be thus described. When there is only one output ($m = 1$), as in this section, behavior f can also be described by the set $f^{-1}(1)$. The case $m \geq 1$ will be considered in the section on design for variety.

Definition 2.1 — Product behavior (functionality) is a function $f : D^n \to D$; or equivalently a subset B of D^n intended to represent $f^{-1}(1)$. □

Design consists in the search for components that, put together, define a product with the desired behavior. Components will be made of elementary pieces called gates.

Definition 2.2 — A gate is any function mapping D or D^2 into D. □

There are four gates mapping D into D: always 0, always 1, identity, and negation or NOT (mapping 0 into 1, and 1 into 0). There are sixteen gates mapping D^2 into D. This section considers only two: logical sum (OR), defined by $0+0 = 0$, $0+1 = 1+0 = 1+1 = 1$; and logical product (AND), defined by $1 \cdot 1 = 1$, $1 \cdot 0 = 0 \cdot 1 = 0 \cdot 0 = 0$. Any function mapping D^n into D can be obtained by combining AND, OR, and NOT gates. To see this, we need the identity $f(x_1 \cdots x_i \cdots x_n) = x_i f(x_1 \cdots 1 \cdots x_n) + x_i' f(x_1 \cdots 0 \cdots x_n)$, where x_i' denotes the negation of x_i: both sides of the identity equal $f(x_1 \cdots 1 \cdots x_n)$ when $x_i = 1$, and both sides equal $f(x_1 \cdots 0 \cdots x_n)$ when $x_i = 0$. Repeated application of this identity, each time for a different variable, yields a form, called a cover, involving only AND, OR, and NOT. For example, the function f defined by $f(1,0) = 0$, $f(0,0) = f(0,1) = f(1,1) = 1$, decomposes as follows: $f(x_1, x_2) = x_1\, f(1, x_2) + x_1'\, f(0, x_2) = x_1[x_2\, f(1,1) + x_2'\, f(1,0)] + x_1'[x_2\, f(0,1) + x_2'\, f(0,0)] = x_1x_2 + x_1'x_2 + x_1'x_2'$.

I now formalize this discussion.

Definition 2.3. A literal is a member of D, a variable, or a negated variable. Equivalently, a literal is a form x_i^A, where x_i is a variable and A a subset of D, where $x_i^\phi = 0$, $x_i^D = 1$, $x_i^1 = x_i$, $x_i^0 = x_i'$. Members of D are trivial literals. □

Definition 2.4. A cube is a product of literals, i.e. a form $\Pi_{i=1}^n x_i^{A_i}$. If all A_i are singleton, the cube is called a minterm, and we say that all variables are present in it. A cover is a sum of cubes.

Definition 2.5. The behavior $b(F)$ of cover F is a subset of D^n inductively defined by

- $b\left(x_i^A\right) = D \times \ldots \times A \times \ldots \times D$, where A is in the i–th position;
- $b\left(\Pi_{i=1}^n x_i^{A_i}\right) = \cap_{i=1}^n b(x_i^{A_i}) = A_1 \times A_2 \times \ldots \times A_n$;
- $b\left(\sum_{i=1}^k Q^i\right) \doteq \cup_{i=1}^k b(Q^i)$, where $F = \sum_{i=1}^n Q^i$ is a sum of cubes.

Two covers F, G with the same behavior are called equivalent; equivalence is denoted by $F \sim G$. □

As an example $b(x_1x_2 + x_1'x_2 + x_1'x_2') = b(x_1x_2) \cup b(x_1'x_2) \cup b(x_1'x_2') = \{11, 01, 00\}$. This behavior, however, can also be realized by the less costly cover $x_1' + x_2$. Note also that each behavior B is realized by the maximally redundant cover $F_B = \sum_{\beta \in B} m_\beta$, where $m_\beta = \Pi_{i=1}^n x_i^{\beta_i}$; $b(F_B) = \cup_{\beta \in B} b(m_\beta) = \cup_{\beta \in B} \{\beta\} = B$. Any behavior, therefore, can be specified by a cover.

Definition 2.6. A design problem is a cover G. The cost of a cover is the number of cubes it contains. A solution to a design problem G is a minimum–cost cover F equivalent to G.

In terms of the discussion in the Introduction, the product is a cover and its components are the cubes in the cover. The search for minimum–cost covers can be focused if some of their properties are known in advance. □

Definition 2.7. Let F, G be covers; G is covered by $F (G \leq F)$ if $b(G) \subseteq b(F)$. □

For example, $G = x_1x_2 + x_1x_3$ is covered by $F = x_1$. When F, G are cubes, $G \leq F$ iff every nontrivial literal in F is also in G: $x_1x_2\bar{x}_3 \leq x_1\bar{x}_3 \leq x_1$. Equivalently if $G = \Pi_{i=1}^n x_i^{A_i}$, $F = \Pi_{i=1}^n x_i^{B_i}$ are cubes, then $G \leq F$ iff A_i is a subset of B_i for all i.

Definition 2.8. A cube p is an implicant of cover F if $p \leq F$. A cube p is a prime implicant of F if $p \leq F$ and, in addition, $p \leq q \leq F$ implies $p = q$; in other words, if any literal is dropped from p, it ceases to be an implicant of F. □

For example, $x_1x_2x_3 \leq x_1x_2 \leq F = x_1x_2x_3' + x_1x_2x_3 + x_1'x_2'x_3'$; $x_i \not\leq F$, $i = 1, 2$; $x_1x_2x_3$ is an implicant of F, but not a prime one; x_1x_2 is a prime implicant of F; neither x_i is an implicant of F. Note also that equivalent covers have the same set of primes, because $F \sim G$ implies that $p \leq F$ iff $p \leq G$.

Definition 2.9. A cover F is prime if every cube in F is a prime implicant of F. □

The covers $F = x_1' + x_1x_2$, $G = x_1' + x_2$ are equivalent, but only G is prime: x_1x_2 is not a prime of F since $x_1x_2 \leq x_2 \leq F$, $x_1x_2 \neq x_2$.

Primes are interesting because of

Theorem 2.1. Every design problem G has at least one prime solution. □

Proof. Let $F = \sum_{i \in I} q^i$ solve G. Then

$$b(G) = b(F) = \cup_{i \in I} b(q^i). \tag{1}$$

Let I_1 be the set of all i in I such that q^i is not a prime of G, and I_2 its complement. For each i in I_1, drop literals from q^i until it becomes a prime $\hat{q}^i$ of G. Then

$$b(q^i) \subseteq b(\hat{q}^i) \subseteq b(G). \tag{2}$$

The cover $\hat{F} = \sum_{i \in I_1} \hat{q}^i + \sum_{i \in I_2} q^i$ has the same cost as F does; it is prime by construction; and it is equivalent to G by (1) and (2). Hence $\hat{F}$ is a prime solution of G. □

Note that if cost was also increasing in the number of literals in a cover, then all solutions of design problems would be prime. In any case, the search for solutions can be restricted to prime covers. Altough a prime solution of G consists of primes, it does not necessarily consist of all primes of G. For example, $G = x_1x_2 + x_1'x_3 + x_2x_3$ is a prime cover, but only the first two cubes constitute a solution of G: the third cube, x_2x_3, is prime but redundant. One obvious division of design labor, therefore, is to first compute all primes of the design problem and then search for, and eliminate, redundant primes.

Definition 2.8. The Quine–McCluskey (QM) procedural division of labor comprises two steps:

1. For each design problem G, compute its set of primes $\pi(G)$.
2. Find the smallest subset F of $\pi(G)$ that covers G $(G \leq F \subseteq \pi(G))$.

3 Computation of Primes

Primes are the components out of which a product exhibiting the desired behavior will be built. The set of all primes of a design problem is uniquely characterized by two properties, maximality and compactness. Every method of computing primes will have, therefore, to transform the design problem G into a compact, maximal cover with the same behavior. It will turn out that maximality is achieved by removing cubes while preserving behavior; while compactness is achieved by adding cubes while preserving behavior. The properties of the cubes to be added or removed will delimit the possible

divisions of the labor of computing primes, and will allow comparisons of their efficiency properties.

Definition 3.1. A cover F is compact if every cube p covered by F is covered by some cube q in F. A cover F is maximal if for any two cubes p, q in F, $p \leq q$ implies $p = q$.

The cover $F_1 = x_1' + x_2$ is compact; the equivalent cover $F_2 = x_1'x_2' + x_2$ is not, because $x_1' \leq F_2$ but $x_1' \not\leq x_1'x_2'$ and $x_1' \not\leq x_2$. The cover $F_3 = x_1' + x_2 + x_2x_3$ is compact and nonmaximal; F_2 is maximal but not compact; while F_1 is both maximal and compact. Definition 3.1 is interesting because of

Theorem 3.1. A cover F consists of its primes ($F = \pi(F)$) iff it is both compact and maximal. □

Behavior–preserving maximality is easy to achieve, because $p \leq q$ implies $p+q \sim q$: If p, q are in F and $p \leq q$, then remove p from F. This gives rise to

Definition 3.2. The maximal equivalent $M(F)$ of F is what remains of F after all nonmaximal cubes are removed.

Behavior–preserving compactification of F involves adding cubes to F. For example $F = x_1'x_2' + x_2x_3$ is noncompact because $x_1'x_3 \leq F$ but $x_1'x_3 \not\leq x_1'x_2'$ and $x_1'x_3 \not\leq x_2x_3$: hence any compactification of F must add to F a cube covering $x_1'x_3$, namely either x_1', or x_3, or $x_1'x_3$ itself. Since neither x_1' nor x_3 are implicants of F, the only behavior–preserving compactification of F is $F + x_1'x_3$. To understand the general case, we need the next two definitions.

Definition 3.3. The distance $d(p, q)$ of two nonzero cubes $p = \Pi_{i=1}^n x_i^{A_i}$, $q = \Pi_{i=1}^n x_i^{B_i}$ is the number of their opposed literals, i.e. $d(p, q) = \#\{i : A_i \cap B_i = \phi\}$. □

For example, $d(x_1x_2,\, x_1x_3) = 0$, $d(x_1,\, x_1'x_2) = 1$, $d(x_1x_2,\, x_1'x_2'x_3) = 2$, $d(x_1x_2x_3,\, x_1'x_2'x_3') = 3$.

Definition 3.4. Let $dpq = 1$, and let x_i be the unique variable such that $A_i \cap B_i = \phi$. Then the consensus $c(p,\, q)$ of the two cubes is obtained by multiplying p and q after removing the opposing literals $x_i^{A_i}$ and $x_i^{B_i}$, i.e. $c(p,\, q) = \Pi_{j \neq i}\, x_j^{A_j \cap B_j}$. □

For example, $c(x_1,\, x_1'x_2) = x_2$, $c(x_1x_2,\, x_2'x_3) = x_1x_3$.

Theorem 3.2. Let p, q be two cubes in F at distance one from each other. Then $F \sim F + c(p, q)$. □

Proof. Since $dpq = 1$, $p = xu$, $q = x'v$ for some variable x not in u or v. Then

$F = p + q + G$, while $F + c(p,q) = p + q + uv + G$. Hence, it suffices to show that $xu + x'v + uv \sim xu + x'v$. In fact, both sides equal u when $x = 1$, and both sides equal v when $x = 0$. □

Theorem 3.2 shows that adding to F the consensus of any two cubes in F is behavior–preserving. The next theorem suggests that adding consensus cubes is necessary for compactification.

Theorem 3.3. A cover F is compact iff the consensus of any two cubes in F is covered by some cube in F. □

The last three theorems sugggest how to compute primes.

Theorem 3.4. The primes of design problem G can be obtained by repeated application of the following two rules, until neither applies:

1. If p, q are in G, and $p \leq q$, then remove p from G;
2. if p, q are in G, but their consensus $c(p,q)$ is not covered by any cube in G, then add $c(p,q)$ to G. □

Proof. By Theorem 3.2, application of rule 2 preserves behavior. By the fact that $p \leq q$ implies $b(p + q) = b(q)$, application of rule 1 preserves behavior. Hence, all the covers derived from G by application of those rules behave as G does. If neither rule applies to a cover, then this cover is maximal and, by Theorem 3.3, compact. Hence, if the process terminates, the resulting cover is maximal, compact, and equivalent to G, i.e., by Theorem 3.1, it is the set of primes of G. To prove termination, note that rule 2 can add to G at most all cubes formed out of $x_1 \ldots x_n$; that rule 2 can be applied only once to each pair of cubes; and that rule 1 can only reduce the number of cubes. Hence, both rules will cease to apply after a finite number of applications.

The division of prime–generating labor can thus be based exclusively on efficiency grounds, since by Theorem3.4 the order of cube additions and removals will affect neither correctness nor termination. The earliest such division, due to Quine and McCluskey, is based on the observation that rule2 can be simplified if the original cover G consists of minterms only. If p, q are minterms at distance one from each other, then $p = xu$, $q = x'u$ for some variable x, i.e. their nonopposed parts have to be equal. Rule 2 replaces $xu + x'u$ by $xu + x'u + u$; rule 1 then replaces the latter, by u. Hence, the two rules can be combined into one: replace any instance of $xu + x'u$ by u.

For example, when $n = 3$, the cover $F_1 = x'_1x'_2x'_3 + x_1x'_2x'_3 + x'_1x_2x'_3 + x_1x_2x'_3 +$

$x_1x_2x_3$ reduces to $F_2 = x_2'x_3' + x_1'x_3' + x_1x_3' + x_2x_3' + x_1x_2$ after application of $xu + x'u = u$ to all possible pairs in F_1. It is remarkable that although F_2 consists no longer of minterms, the same rule applied to F_2 suffices to generate a prime cover. It is not necessary, for instance, to add to F_2 the consensus of x_1x_2 and $x_2'x_3'$, namely x_1x_3', because this term is already in F_2. It suffices to apply the rule $xu + x'u = u$ to F_2 to obtain the prime cover $F_3 = x_1x_2 + x_3'$. The next definition and theorem establish the correctness of the procedure just outlined.

Definition 3.4. Let F be a cover; then
$A(F) = \{t$: there exists a literal ℓ such that ℓt, $\ell' t \in F\}$.
$S(F) = \{p$: there exists a literal ℓ and a cube t such that $p = \ell t \in F$, $\ell' t \in F\}$.

The set $A(F)$ represents the new cubes generated from applying the rule $\ell t + \ell' t \longrightarrow t$ on F; while $S(F)$ represents the cubes deleted from F as a result of the application of the same rule. The next theorem shows that the labor of computing primes can be split into $n + 1$ phases, where n is the number of variables in F.

Theorem 3.5. Let G be a cover, and $b(G)$ the set of its minterms. For each $t = n, n - 1, \ldots, 1$, let $F^n = b(G)$, $F^{t-1} = A(F^t)$, $S^t = S(F^t)$, $\pi^t = F^t \backslash S^t$. Then the set of primes of G is $\pi(G) = \cup_{t=1}^{n} \pi^t$.

The search for pairs $(xp, x'p)$ does not have to consider all pairs of cubes in F: note that xp contains one more positive, nontrivial literal than $x'p$ does. Hence, cubes in F can be sorted according to the number of such literals they contain; and application of the rule $xp + x'p = p$ restricted to cubes differing by one in this measure.

Definition 3.5. The number $\lambda(p)$ of positive nontrivial literals in cube $p = \Pi_{j=1}^{n} x_j^{A_j}$ is $\lambda(p) = \#\{j : A_j = \{1\}\}$. □

For example, $\lambda(x') = 0, \quad \lambda(x'y) = 1, \quad \lambda(x'yz) = 2.$

Definition 3.6. Iterative prime generation method (Quine–McCluskey). Given an arbitrary design problem G

1. Change problem representation, i.e. compute the behavior $b(G)$ of G (Def. 2.5) and replace G by the equivalent cover F^n consisting of all minterms in $b(G)$.
2. Sort the minterms in F^n into groups $F_0^n, F_1^n, \ldots, F_n^n$, where $F_i^n = \{p \in F^n : \lambda(p) = i\}$.
3. Divide the labor of computing $F^{t-1} = A(F^t)$. For each $i = 0, 1, \ldots, n$, compute $F_i^{t-1} = A(F_i^t \cup F_{i+1}^t)$; then set $F^{t-1} = \cup_{i=0}^{n} F_i^{t-1}$.

4. Divide the labor of computing $S^t = S(F^t)$. For each $i = 0, 1, \ldots, t, \quad S_i^t = S(F_i^t \cup F_{i+1}^t), \quad S^t = \cup_{i=0}^t S_i^t$.

5. Set $\pi^t = F^t \backslash S^t, \quad \pi(G) = \cup_{t=0}^n \pi^t$. □

The next example shows that step 1 of the QM method cannot be omitted.

Example 3.1. $G = x_1' + x_1 x_2$ is not a cover of minterms, since x_1' is not a minterm. Although G is not prime ($x_2 \leq G$ but $x_2 \not\leq x_1'$, $x_2 \not\leq x_1 x_2$), neither A nor S apply on G; hence, the iterative method without step 1 cannot find the primes of G. Applying step 1 on G yields $F^2 = x_1' x_2 + x_1' x_2' + x_1 x_2, \quad A(F^2) = \{x_1', x_2\}, \quad S(F^2) = F^2$. Hence, $F^1 = A(F^2) = x_1' + x_2 =$ the primes of G. □

The cost of step 1 of the iterative prime generation method is the cost of generating and storing the minterms in the behavior of G. For example, when $G = x_1 + x_1' x_2 x_3$, then $F^3 = x_1(x_2 x_3 + x_2 x_3' + x_2' x_3 + x_2' x_3') + x_1' x_2 x_3$. Each $p \in G$ that contains $k = k(p)$ variables has to be replaced by an equivalent cover consisting of 2^{n-k} minterms. Hence F^n can contain up to $\sum_{p \in G} 2^{n-k(p)}$ minterms, i.e. is an exponential function of the descriptional economy coefficients $n - k(p)$. The next example exhibits a family $< G_n >$ of covers with each G_n containing exactly two cubes, but with $|b(G_n)| = 2^{n-1} + 1$.

Example 3.2. $G_n = x_1 + x_1' x_2 x_3 \ldots x_n, \quad F_n^n = x_1 K_n + x_1' x_2 x_3 \ldots x_n, \quad n \geq 2$, where $K_n \sim 1$ is a cover recursively defined by $K_2 = x_2 + x_2', \quad K_n = x_n K_{n-1} + x_n' K_{n-1}$. K_n contains all minterms built from $x_2, \ldots, x_n$ and is thus a cover of 2^{n-1} minterms.

The cost of step 2 (sorting) of the QM prime generation method is proportional to $n N \log N$, following standard sorting algorithms described in Cormen et al. (1990, chapter 9.1). Step 2 has to be performed only once. Step 3, however, has to be repeated T times; T is bounded from above by the number of variables n, since each application of A adds cubes with one variable less. Step 3 requires, for each $i = 0, \ldots, n$ checking each pair (p, q) in $F_i^t \times F_{i+1}^t$ for the pattern $p = x'w, \quad q = xw$. There are $|F_i^t| \cdot |F_{i+1}^t|$ such pairs, and each check takes up to n comparisons. Hence step 3 takes in all $n \sum_{i=0}^n |F_i^t| \, |F_{i+1}^t|$ comparisons. Recalling that $F^t = \cup_{i=0}^n F_i^t$, this number is bounded by $n|F^t|^2$. In many cases $|F^t| \leq N$ for all t, so total cost is $n T N^2 \leq n^2 N^2$. In these cases, the strategy of the QM method, namely trading off a maximally explicit problem representation for a better division of search labor, pays only if the descriptional economy coefficients, $n - k(p)$ are small, i.e. if the original cover G is already nearly maximally redundant. There are other cases, however, where F^n has more primes than minterms (hence for some $t \quad |F^t| > N$), as shown in Example 3.3 below. In such cases, the only upper bound on $|F^t|$ is 3^n, namely the number of cubes formed out of n variables. The labor of step

1 is then wasted, since it is inevitable to write down the primes of F^n.

Example 3.3. Let B_n be the set of all $0-1$ vectors x in D^n such that the (arithmetic) sum $x_1 + \ldots + x_n$ is not divisible by 3. Let π_n be the number of primes of B_n. Then $\lim_{n\to\infty} \frac{\pi_n}{2^n} = \infty$ along the subsequence $n = 6k + 2$. □

This statement (proven in the appendix) shows that there are covers that have many more primes than minterms, since the number of minterms is bounded above by 2^n. □

QM's iterative prime generation method illustrates a design philosophy: Represent a problem in a maximally explicit way in order to apply better divisions of search labor (only cubes in adjacent groups F_i^t, F_{i+1}^t need to be compared). A different design philosophy is to maintain the descriptional economy of cubes relative to minterms, divide the design problem itself into subproblems, compute the primes of subproblems, and then combine them to form the primes of the original problem. This philosophy, divide–and–conquer, is different from the iterative one of QM: QM never divides the design problem itself; each iteration produces either primes of the original problem or cubes to be used by later iterations, not primes of subproblems. Divide–and–conquer expresses a design problem as a product of two simpler ones; it then computes primes for each element of the product separetely, and finally combines them into primes of the original problem.

Definition 3.7. The product of two cubes $p = \Pi_{i=1}^n x_i^{A_i}, \quad q = \Pi_{i=1}^n x_i^{B_i}$ is the cube $pq = \Pi_{i=1}^n x_1^{A_i \cap B_i}$. The product of two covers $F = \sum_{i\in I} p^i$ and $G = \sum_{j\in J} q^i$ is the cover $FG = \sum_{i\in I} \sum_{j\in J} p^i q^j$. □

The product of $F = x_1 + x_2x_3, \quad G = x_1'x_4 + x_3x_4'x_5$, for instance, is $FG = x_1x_3x_4'x_5 + x_2x_3x_1'x_4 + x_2x_3x_4'x_5$ (zero cubes and duplicate literals deleted). Note that $pq \neq 0$ only if $dpq = 0$.

Definition 3.8. Let F be a cover and x a variable. The cofactor F_x of F with respect to x is F with every instance of x deleted, and every cube containing $\bar{x}$ deleted. Similarly, $F_{\bar{x}}$ is F with every instance of $\bar{x}$ deleted, and every cube containing x deleted. □

If $F = x_1x_2' + x_1'x_3 + x_2x_3$, for instance, then $F_{x_1} = x_2' + x_2x_3, \quad F_{x_1'} = x_2x_3 + x_3$, $F_{x_3} = x_2 + x_1' + x_1x_2', \quad F_{x_3'} = x_1x_2'$.

Recall that $M(F)$ is the maximal equivalent of F (all nonmaximal cubes deleted); and that $\pi(F)$ is the cover consisting of all primes of F. The result that allows a divide–and–conquer computation of primes is

Theorem 3.6. Let x be any variable occurring in cover F. Then $\pi(F) = M((x' + \pi(F_x))(x + \pi(F_{x'})))$.

Example 3.4. Let $G = x' + xy$. Then $G_x = y, \quad G_{x'} = 1; \quad \pi(G_x) = y, \quad \pi(G_{x'}) = 1;$ $x' + \pi(G_x) = x' + y; \quad x + \pi(G_{x'}) = x + 1 \sim 1$. Hence $\pi(G) = M(x' + y) = x' + y$. Note that step 1 of QM was avoided. □

Straightforward application of Theorem 3.6 may require computation of all 2^n cofactors of a design problem in n variables. It is thus important to split the design problem along variables x such that cofactors with special properties, requiring no further decomposition, are obtained as early as possible.

Definition 3.9. A cover F is monotone in x if all instances of x in F have the same sign, i.e. they are either all primed or all unprimed. F is monotone if it is monotone in each variable. □

The cover $F = x_1x_3 + x_2'x_3' + x_1x_2'$ is monotone (increasing) in x_1, monotone (decreasing) in x_2, and nonmonotone in x_3; it is not monotone. The cover $G = x_1x_3 + x_1x_2'$ is monotone. The next theorem suggests a good division of the design problem in order to compute primes.

Theorem 3.7. If F is a monotone cover, then $\pi(F) = M(F)$. □

The cover $G = x_1x_3 + x_1x_2'$, for instance, satisfies $G = \pi(G)$ since G is monotone and contains only maximal cubes. The cover $G + x_1x_2'x_3$ satisfies $\pi(G + x_1x_2'x_3) = G$, since it is monotone and $x_1x_2'x_3$ is covered by x_1x_3.

Theorem 3.7 suggests that design problems should be divided along their nonmonotonic variables, so that subproblems become monotone after the fewest possible decompositions. This helps avoid exponential blowup.

Another problem of divide–and–conquer prime computation is that F_x and $F_{x'}$ may share many cubes, resulting in unnecessary duplication of effort. The cover $F = x_1x_2 + x_1'x_3x_2 + x_2'x_3$ contains two nonmonotonic variables, x_1 and x_2. The cofactors with respect to x_1, namely, $F_{x_1} = x_2 + x_2'x_3$, $F_{x_1'} = x_2x_3 + x_2'x_3$, share the cube $x_2'x_3$, and would share any cube independent of x_1. The cofactors with respect to x_2, namely $F_{x_2} = x_1 + x_1'x_3$, $F_{x_2'} = x_3$, do not share any cubes because all cubes in F depend on x_2. It pays, therefore, to divide the design problem along the nonmonotonic variable that appears in most cubes. In case of a tie, it pays to choose a variable x that minimizes the difference between positive and negative instances, so that size differences between F_x and $F_{x'}$ are minimized. This is because detection of non–maximal cubes in F requires

checking each pair (p, q) in $F \times F$ for the pattern $p \leq q$, and there are $|F|^2$ such pairs, i.e. the cost of constructing $M(F)$ is a convex function of $|F|$. A variable that satisfies these requirements (nonmonotonic, most instances in F, most balanced instances) will be called "appropriate" in the next definition.

Definition 3.10. The divide–and–conquer prime computation method. Let F be a cover. Then

1. If $F = 0$, or F contains exactly one cube, then $\pi(F) = F$.
2. If $1 \in F$, then $\pi(F) = 1$.
3. If F is monotone, then $\pi(F) = M(F)$.
4. If F is not monotone, then pick an appropriate variable x and divide the problem as follows
$$\pi(F) = M((x' + \pi(F_x))\,(x + \pi(F_{x'}))).$$ □

The algorithm will terminate, since decomposition will eventually reduce each cofactor to one of the three first cases. By Theorems 3.6 and 3.7, the algorithm will compute the primes of F. Its cost is determined by the number of decompositions required to arrive at cofactors that satisfy one of the three first cases; and on the number of primes of F. Note that if only one decomposition is required, say along x, then the algorithm will perform one multiplication of covers, namely $\pi(F_x) \times \pi(F_{x'})$, since $(x' + \pi(F_x))\,(x + \pi(F_{x'})) = x\pi(F_x) + x'\pi(F_{x'}) + \pi(F_x) \times \pi(F_{x'})$. If two decompositions are needed, say along x and y, then the algorithm will perform three cover multiplications, namely $\pi(F_{xy})\,\pi(F_{xy'})$, $\pi(F_{x'y})\,\pi(F_{x'y'})$, and $\pi(F_x) \times \pi(F_{x'})$. In general, if decomposition along T variables is needed, $2^T - 1$ cover multiplications need to be performed.

An upper bound on T is the number of nonmonotonic variables of F: Another upper bound is $\max_{p \in F}\, k(p)$, where $k(p)$ is the number of variables in cube p, since at least one of cofactors F_x, $F_{\bar{x}}$ contains a cube of F with one variable less. Hence, divide–and–conquer will save labor relative to QM on covers that contain many monotonic variables, and/or have high descriptional economy coefficients $n - k(p)$. Labor will be wasted, on the other hand, on covers that are nearly compact but have few monotonic variables. This is because divide–and–conquer, unlike QM, does not seek cubes at distance one from each other in order to form their consensus; it avoids this search by subdividing the design problem in search of monotonic, or at worst single–cube, subcovers. It will thus fail to recognize a cover that is already (nearly) compact, as the next example shows.

Example 3.5. Let the family of covers $< F^n >$ be inductively defined by $F^2 = x_1 x_2'$, $F^n = F^{n-1} x_n'$ if n is even, $F^n = F^{n-1}\, x_n' + x_1 x_2, \ldots, x_n$ if n is odd. F^n is the cover that contains, for each odd $k \leq n$, a minterm with exactly k positive literals. For example $F^5 = F^4\, x_5' + x_1 \ldots x_5 = F^3 x_4' x_5' + x_1 \ldots x_5 = (F^2 x_3' + x_1 x_2 x_3)\, x_4' x_5' + x_1 x_2 \ldots x_5 = x_1 x_2' x_3' x_4' x_5' + x_1 x_2 x_3 x_4' x_5' + x_1 x_2 x_3 x_4 x_5$. Each F^n is already prime, since any two of its cubes are at distance two or more, and all are maximal. Each F^n contains $N_n = \frac{n}{2}$ cubes. QM will sort F^n into groups $F_1^n, F_3^n, F_5^n, \ldots$, each consisting of one cube, and then stop; it will thus spend only sorting labor $nN \log N = n\frac{n}{2} \log \frac{n}{2}$. Divide–and–conquer on the other hand will divide each F^n along x_n if n is odd, or along x_{n-1} if n is even. Hence, recursively, it will split the design problem exactly $T_n = \frac{n}{2}$ times, and will then perform $2^{T_n} - 1$ cover multiplications. It will thus be exponentially more costly than QM on this cover. □

Ruddel and Sangiovanni–Vincentelli (1987, p. 739) briefly describe a third prime generation method that, like QM but unlike divide–and–conquer, derives a more explicit problem representation before starting generating primes; and, like divide–and–conquer but unlike QM, it seeks to exploit properties of monotonic covers. The design problem G is first rendered more explicit by deriving a complementary cover G', i.e. a cover whose behavior $b(G')$ equals $D^n - b(G)$. Then a new monotonic cover I_G is derived, such that $\pi(I_G) = M(I_G) = \pi(G)$. As the next example will show, the construction of I_G is based on a different division of labor, inspired from inductive generalization. The goal of deriving primes of G is (roughly) split into the subgoals of generalizing each implicant p of G (by dropping literals) until any further generalization would cause the behavior $b(p)$ of p to intersect $b(G')$, namely the (forbidden) behavior of G's complement.

Example 3.6. Let $G = x_1 x_2 x_3 + x_1' x_2 x_3 + x_1 x_2' x_3$. A complementary cover of G is $G' = x_3' + x_1' x_2'$. It is obvious (and shown in the Appendix) that any implicant p of G must be at distance one or greater from each cube in G', since $b(p) \subseteq b(G)$, $b(G) \cap b(G') = \phi$. To express this, let z_{kj}, $k = 0, 1$, $j = 1, 2, 3$ be new binary variables with the following interpretation: $z_{kj} = 1$ if there is an implicant of G that contains x_j^k, $z_{kj} = 0$ if no implicant of G contains x_j^k (recall that $x_j^0 = x_j'$, $x_j^1 = x_j$). For each j we must have $z_{0j} + z_{1j} = 1$, since each x_j appears in G, either primed or unprimed, or both. By inspection of G' in this example, $z_{03} = 0$ because no implicant of G can contain x_3'; and either $z_{01} = 0$ or $z_{02} = 0$, because no implicant of G can contain $x_1' x_2'$. The formula that expresses these two conditions is $I_G = \bar{z}_{03}(\bar{z}_{01} + \bar{z}_{02}) = \bar{z}_{03}\bar{z}_{01} + \bar{z}_{03}\bar{z}_{02}$. I_G is by construction monotonic decreasing, and in this example prime (all its cubes are maximal). To obtain the primes of G, take each prime of I_G and replace $\bar{z}_{kj}$ by x_j^{1-k}. To see why this makes sense, take $\bar{z}_{03}\, \bar{z}_{01}$, the first prime of I_G; it is equal to 1

iff $z_{03} = z_{01} = 0$, i.e., from the equations $z_{0j} + z_{1j} = 1$, iff $z_{13} = z_{11} = 1$, i.e. from the interpretation of the z_{kj}, iff there is an implicant of G that contains x_3 and x_1, namely x_3x_1. This is also the result of replacing $\bar{z}_{03}$ by $x_3^{1-0} = x_3$ and $\bar{z}_{01}$ by $x_1^{1-0} = x_1$.

To see why x_1x_3 is a prime of G, note that $d(x_3, x_1'x_2') = 0$ and $d(x_1, x_3') = 0$, i.e. any cube covering x_1x_3 is at distance zero from some cube in the complement of G. In this example, then, $\pi(G) = x_1x_3 + x_2x_3$; this can be verified by applying QM on G. □

The next two definitions describe a systematic way of constructing I_G out of G'.

Definition 3.11. Positional notation for subsets of $D : \{0\}$ is represented by the vector 10, i.e. the first element of $D = \{0, 1\}$ is present and the second is absent; $\{1\}$ is represented by 01; and D by 11 (ϕ is not represented).

Definition 3.12. For each $j = 1, \ldots, n$ and each cube $q \neq 0$, $\alpha_j(q)$ is positional notation for the exponent of x_j in q, i.e. $\alpha_j(q) = 10$ if q contains x_j'; 01 if it contains x_j; and 11 if q does not depend on x_j. The first element of $\alpha_j(q)$ is $\alpha_{0j}(q)$, and the second $\alpha_{1j}(q)$.

Definition 3.13. The "inductive generalization" prime generation method. Let G, G' be implementary covers. First construct the (monotonic) cover I_G in four stages:

1. $H_j(q) = (\alpha'_{0j}(q) + z'_{0j})(\alpha'_{1j}(q) + z'_{1j})$ for each $q \in G'$, $j = 1, \ldots, n$.
2. $H(q) = \sum_{j=1}^{n} H_j(q)$.
3. $I = \Pi_{q \in G'} H(q)$.
4. I_G is obtained from I by performing the multiplications in I's definition and deleting any cube that contains a $z'_{0j}z'_{1j}$ term (this enforces the constraint $z_{0j} + z_{1j} = 1$).

□

Secondly, extract the primes of I_G and G:

5. $\pi(I_G)$ is obtained from I_G by eliminating nonmaximal cubes.
6. $\pi(G)$ is obtained from $\pi(I_G)$ by replacing every instance of z'_{kj} by x_j^{1-k}. □

The correctness of this procedure is guaranteed by

Theorem 3.8. For any design problem G, $\pi(G)$ and $\pi(I_G) = M(I_G)$ are isomorphic. Each prime of I_G gives rise to a prime of G by replacing every instance of z'_{kj} by x_j^{1-k}.

The benefit of generating primes by "inductive generalization" is that fewer cubes are first generated and then discarded during prime generation, because excessive cube generation is checked by our explicit knowledge of G'. To see this, recall that in Example 3.6, every cube of I_G gives rise to a prime of G, i.e. waste in the prime generation process has been eliminated. The following example shows that his is not the case with divide–and–conquer.

Example 3.6 (... continued). We compute the primes of $G = xyz + x'yz + xy'z$ by divide–and–conquer: $G_x = yz + y'z$, $G_{x'} = yz$, $G_{xy} = z = G_{xy'}$. Hence $\pi(G_{x'}) = yz$, $\pi(G_x) = M(y' + \pi(G_{xy}))(y + \pi(G_{xy'})) = M((y' + z)(y + z)) = M(y'z + yz + zz) = z$. Finally, $\pi(G) = M((x'+\pi(G_x))(x+\pi(G_{x'})) = M((x'+z)(x+yz)) = M(x'yz+xz+yzz) = xz + yz$. The cubes first generated and then discarded are $y'z$, yz (in the computation of $\pi(G_x)$), and $x'yz$ (in the computation of $\pi(G)$).

An example of a more explicit problem representation developed by Toyota is described in Ward et al. (1995). Engineers have to pick parts specifications out of "engineering check sheets" (or lessons–learned books) that describe explicitly those specifications that are likely to be manufacturable. At GM, on the other hand, "designers were encouraged to draw the cars unencumbered by technical specifications that were believed to inhibit creativity" (Peters, 1993, p. 730). Lessons–learned books render the design problem more explicit, and thus guide search in the same way that a complementary cover guides the search for primes, namely by providing an easy check (no derivations) of what is feasible and what is not. These books were being developed for the last 15 years (ibid., p. 52). An idea of the design effort involved is given by Okino (1995, p. 82): "Toyota has as many as 300.000 specifications relating to quality standards for its parts. Specs for structural components cover materials, processing methods, precision levels, strengthen factors, and so on. One could say this book represents the bible of Japanese auto quality."

The cost of generating primes by "inductive generalization" is the cost of two backroom operations, namely multiplication and complementation. Note that multiplying two covers F, G using only the definition of product cover takes $|F|\ |G|$ operations. It follows that to multiply N covers, $F_1 \ldots F_N$, each of size $|F^i| = M$, takes $|F_1||F_2| \ldots |F_N| = M^N$ operations. Exponential cost is unavoidable when, for instance, the covers multiplied do not share variables. To reduce the cost of this backroom operation, therefore, good design has to exploit the special structure of the covers multiplied. First note that each $H_j(q)$ term, after performing the (four) multiplications involved in its definition and eliminating the $z'_{0j}\, z'_{1j}$ term, contains at most one literal, namely $H_j(q) = z'_{0j}$ if $\alpha_j(q) = 10$; $H_j(q) = z'_{1j}$ if $\alpha_j(q) = 01$; $H_j(q) = 0$ if $\alpha_j(q) = 11$. Covers

such as $H(q)$ that are sums of literals will be called atomic.

Definition 3.14. The monotonic–atomic (MA) cover multiplication problem consists of N covers $F^1 \dots F^k \dots F^N$ such that:

(a) Each cover is a sum of literals, i.e. for each $k = 1, \dots, N$, $F^k = \sum_{j=1}^{n} x_j^{A_{kj}}$; $A_{kj} \neq D$ for all k, j.

(b) Each variable j has the same sign in all covers F^k, i.e. if $A_{kj} \neq \phi$, $A_{tj} \neq \phi$, then $A_{kj} = A_{tj}$. □

The covers $H(q)$, for instance, are such that $A_{qj} \neq \phi$ implies $A_{qj} = \{0\}$ (all variables are primed).

The examples that follow show some important factors to be taken into account when dividing cover multiplication labor. The end result of the discussion will be a tree whose leaves are the covers to be multiplied. This tree determines exactly which covers are to be multiplied first, second, etc., to save labor; and is thus a "good" division of such labor.

Example 3.7. Order of multiplications. Let $< F_n >$ be a sequence of MA covers with $|F_n| = n$ and such that if $n \neq m$ then F_n, F_m have no variables in common. An example of such a sequence is $F_1 = x_1$, $F_2 = x_2 + x_3$, $F_3 = x_4 + x_5 + x_6, \dots$. When $N = 4$, for instance, multiplying in the order $((F_1F_2)F_3)F_4$ takes $1 \times 2 + 1 \times 2 \times 3 + 1 \times 2 \times 3 \times 4 = 2! + 3! + 4! = 32$ operations; while multiplying in the order $F_1(F_2(F_3F_4))$ takes $3 \times 4 + 2 \times 3 \times 4 + 1 \times 2 \times 3 \times 4 = \frac{4!}{2!} + \frac{4!}{1!} + \frac{4!}{0!} = 60$ operations. For general N, the corresponding cost figures are $A_N = \sum_{k=2}^{N} k!$ and $B_N = \sum_{k=2}^{N} \frac{n!}{(n-k)!} = \sum_{k=2}^{N} \binom{n}{k} k!$, respectively. The ratio B_N/A_N is always larger than 2^{N-2}, as shown by a simple inductive argument.

Example 3.8. Shared cubes. Covers $F = p + A$, $G = p + B$ share cube p. Their product $FG = p + AB$ is smaller in size than $|F||G|$; and requires only $1 + |A||B|$ operations, i.e. fewer than $|F||G|$.

Example 3.9. Covered cubes. Let $p \leq q$, and let $F = p + A$, $G = q + B$. Then $FG = p + qA + AB$ is smaller in size than $|F||G|$; and requires only $|A|(1 + |B|)$ operations, i.e. fewer than $|F||G|$.

Example 3.10. Elimination of nonmaximal cubes. Let $F_1 = x_1 + x_2$, $F_2 = x_3 + x_4$, $F_3 = x_1 + x_3$, $F_4 = x_2 + x_4$, $F_5 = x_5 + x_6$, and consider the multiplication $((F_1F_2)(F_3F_4))F_5$. Letting i stand for x_i, $F_1F_2 = 13+14+23+24$; $F_3F_4 = 12+14+23+34$; $(F_1F_2)(F_3F_4) = 14 + 23 + (13 + 24)(12 + 34) = 14 + 23 + 123 + 134 + 124 + 234$. At this point (the last

four) nonmaximal cubes can be eliminated, and multiplication with F_5 will take only four operations, as opposed to 12 if nonmaximal cubes are not eliminated.

Definition 3.15. The cost $c(F,G)$ of multiplying two MA covers, and also the size of the resulting cover, is given by

$$c(F,G) = |F \cap G| + |F \backslash G| \cdot |G \backslash F|. \qquad \square$$

For example, if $F = x + y, \quad G = x + u + v, \quad$ then $F \cap G = \{x\}, \quad F \backslash G = \{y\}, \quad G \backslash F = \{u, v\}, \quad c(F,G) = 1 + 1 \times 2 = 3$. The product cover $FG = x + uy + vy$ is of size 3.

The information contained in $c(F,G)$ can be used to order the covers $F' \ldots F^N$ of an MA multiplication problem: Covers that cost less to multiply should be closer together in the ordering.

Definition 3.16. Let $M =< F^1, \ldots, F^N >$ be an MA multiplication problem. To order its elements by the least–cost principle, pick any $G_1 \in M$ and set $T^1 = \{G_1\}$. Then, for each t, $2 \leq t \leq N-2$, pick $G \in M \backslash T^t$ to be the cover least costly to multiply with any of the covers in T^t, i.e. G is a solution of $\min_{H \in M \backslash T^t} \min_{1 \leq i \leq t} c(G_i, H)$. Finally, $T^{t+1} = T^t \cup \{G\}$; and G is ordered immediately after the cover G_i of T^t that, together with G, solves this minimization problem. Rename the covers in T^{t+1} to reflect the new order. $\square$

A least–cost ordering of the covers in example 3.10 for instance is F_1, F_3, F_2, F_4, F_5. Creating a least–cost order takes $N-2$ steps; at each step, the minimum of at most N^2 numbers $c(F^i, F^j)$ is chosen. Choosing the minimum of L numbers takes exactly $L-1$ comparisons (Cormen et al., 1990, p. 186). Hence, the total cost of creating a least–cost order cannot exceed $(N-2)(N^2-1)$, i.e. it is cubic in N.

A least–cost order, while excluding most options, does not completely determine how to multiply N covers. For example, if $F_1 < F_2 < F_3$, we know that $F_1 F_3$ will not be performed; but we don't know whether to perform $(F_1 F_2) F_3$ or $F_1 (F_2 F_3)$. In what follows, b_{ij} is an estimate of the least cost of multiplying $F_i F_{i+1} \ldots F_j$, and s_{ij} an estimate of the size of the resulting cover. $F_1 \ldots F_N$ are assumed to be in least–cost order.

Set $b_{ii} = 0$, $s_{ii} = |F_i|$, $\quad b_{i,i+1} = s_{i,i+1} = c(F_i, F_{i+1})$. For $i < j$, we compute recursively:

$$\begin{aligned} b_{ij} &= \min_{i \leq k < j} \{b_{ik} + b_{k+1,j} + s_{ik} s_{k+1,j}\}; \\ k_{ij} &= \arg\min_{i \leq k < j} \{b_{ik} + b_{k+1,j} + s_{ik} s_{k+1,j}\}; \\ s_{ij} &= s_{ik} s_{k+1,j}, \qquad\qquad k = k_{ij}. \end{aligned}$$

The computation ends when b_{1N} has been obtained. The recursive equations have to be invoked $N-2$ times, in order to decompose b_{1N} into parts that involve only the known quantities b_{ii}, s_{ii}, $b_{i,i+1}$, $s_{i,i+1}$. The discussion of divide–and–conquer showed that, in general, the cost of solving such equations is exponential in the depth of recursion $(N-2)$. In this particular problem, however, the exponential cost is avoidable, because it is due to solving the same subproblems repeatedly. The estimate b_{46}, for instance, will be computed every time b_{ij} is computed, $i \leq 4 < 6 \leq j$, namely $3(n-6)$ times. To avoid this we solve the recursive equations bottom–up (by dynamic programming). We first compute $b_{i,i+2}$, $s_{i,i+2}$ for each $i = 1, \ldots, N-2$ using the recursion equations and $b_{i,i+1}$, $s_{i,i+1}$; then we compute $b_{i,i+3}$, $s_{i,i+3}$ for each $i = 1, \ldots, N-3$ in the same way. In each round we use data from previous rounds, but we don't recompute them.

Example 3.11. Let $F_1 = x_1 + x_2$, $F_2 = x_1 + x_3$, $F_3 = x_3 + x_4$, $F_4 = x_2 + x_4$, $F_5 = x_5 + x_6$, be an MA multiplication problem in least–cost order. Clearly, $s_{ii} = 2$, $b_{ii} = 0$, $b_{12} = b_{23} = b_{34} = 2$, $b_{45} = 4$. We first compute $b_{i,i+2}$, $i = 1, 2, 3$. For example, $b_{13} = \min_{1 \leq k < 3} \{b_{1k} + b_{k+1,3} + s_{1k}s_{k+1,3}\} = \min\{2 + 2 \times 2, 2 + 2 \times 2\} = 6$; $k_{13} \in \{1, 2\}$, say $k_{13} = 1$; $s_{13} = s_{11}s_{23} = 2 \times 2 = 4$. Similarly, $b_{24} = 6$, $k_{24} = 2$, $s_{24} = 4$; $b_{35} = 6$, $k_{35} = 4$, $s_{35} = 4$. In the next round, we compute $b_{i,i+3}$, $i = 1, 2$. For example, $b_{25} = \min_{2 \leq k < 5} \{b_{2k} + b_{k+1,5} + s_{2k}s_{k+1,5}\} = \min\{b_{35} + s_{22}s_{35},\, b_{23} + b_{45} + s_{23}s_{45},\, b_{24} + s_{24}s_{55}\} = \min\{6+2\times4,\, 2+4+2\times4,\, 6+4\times2\} = 14$; $k_{25} \in \{2, 3, 4\}$, say $k_{25} = 2$; $s_{25} = s_{22}s_{35} = 8$. Similarly, $b_{14} = 8$, $k_{14} = 2$, $s_{14} = 4$. Finally, in the last round we compute $b_{i,i+4}$ for $i = 1$, i.e. b_{15}. By the recursion equation $b_{15} = \min_{1 \leq k < 5}\{b_{1k} + b_{k+1,5} + s_{1k}s_{k+1,5}\} = \min\{14 + 2 \times 8, 2 + 6 + 2 \times 4, 6 + 4 + 4 \times 4, 8 + 4 \times 2\} = 2 + 6 + 2 \times 4 = 16$; $k_{15} = 2$; $s_{15} = s_{12}s_{35} = 8$. The information contained in k_{ij} is now used to determine the order of multiplications: $F_1 \ldots F_5 = (F_1F_2)(F_3F_4F_5) = (F_1F_2)((F_3F_4)F_5)$, because, respectively, $k_{15} = 2$, $k_{35} = 4$. □

The dynamic programming algorithm goes through $N-2$ rounds. At each round $t = 2, 3, \ldots, N-1$, we compute b_{ij}, s_{ij}, k_{ij} for $i = 1, \ldots, N-t$, $j = i+t$. For each i, $1 \leq i \leq N-t$, we compute t numbers $b_{ik} + b_{k+1,j} + s_{ik}s_{k+1,j}$, where $j = i+t$, and we take their minimum; these operations take time proportional to t. Hence each round requires time $(N-t)t$, namely one t for each $i = 1, \ldots, N-t$. Summing over all t, the algorithm takes time proportional to $\sum_{t=2}^{N-1}(N-t)t \leq N^3$. It also needs memory space N^2 to store the values of b_{ij}, s_{ij}, k_{ij}. Given that a good division of multiplication labor can generate exponential–size savings (Example 3.7), these costs are reasonable. The discussion so far motivates

Definition 3.17. Multiplication of N MA covers $F_1 \ldots F_N$.

1. Create a least–cost total order on $\{F_1, \ldots, F_N\}$.

2. Determine exactly the order of multiplications using the values k_{ij} provided by the dynamic programming algorithm.

3. Perform the multiplications in this order. After each multiplication, eliminate nonmaximal cubes from the resulting product cover. □

The use of dynamic programming might create the impression that finding a good division of multiplication labor has been reduced to solving an ordinary optimization problem. This is not the case; there is not, for instance, an objective function that attaches to each way of multiplying N covers its true cost. What is more, such an objective function cannot be defined. There are several reasons for this. First, given a total order on $\{F_1, \ldots, F_N\}$, there are $C_N = \frac{1}{N+1}\binom{2N}{N}$ ways to multiply them while respecting the order (Cormen et al., 1990, p. 504). C_N, the N–th Catalan number, is of the same order as $\frac{4^N}{6N^{3/2}}$, a number that exceeds the age of the universe even for moderate values of N. Secondly, it is not always possible to find the true cost of each way of multiplying N covers; because it is not possible to predict in advance how many nonmaximal cubes will be created from partial multiplications. Thirdly, the number of options is even greater if we allow for different total orders on $\{F_1, \ldots, F_N\}$. "Optimal" cover multiplication is therefore an ill–structured problem (Simon, 1973), that has to be "solved" by considerations other than minimization of a given objective function subject to given constraints.

Complementation of covers, the other backroom operation involved in prime generation by "inductive generalization", provides an example of the more obvious division of labor being less efficient. Recall that, given a design problem G, and a complement of it G', we multiply $|G'|$ covers to obtain the primes of G (Definition 3.13). We are thus interested in deriving a short complement of G in reasonable time. The obvious way to complement covers is to use De Morgan's laws, namely $(x+y)' = x'y'$, $(xy)' = x' + y'$. If $G = xy + uwv$, for instance, then $G' = (xy)'(uwv)' = (x'+y')(u'+w'+v') = x'u'+x'w'+x'v'+y'u'+y'w'+y'v'$. The next definition describes this particular division of complementation labor.

Definition 3.18. Complementation reduced to multiplication. Given a cover G, the labor of deriving a complement G' is divided as follows:

1. For each cube $p = \Pi_{j=1}^{n} x_j^{A_j}$ in G, obtain its complement $p' = \sum_{j=1}^{n} x_j^{A'_j}$.

2. Multiply these cube complements to obtain $G' = \Pi_{p \in G} p'$. □

Note that the covers multiplied are all atomic, but this is not an MA multiplication problem, since the same variable can appear unprimed in one cover and primed in another. The discussion that follows will show that, when G is nonmonotonic, this division of complementation labor will generate excessively long complementary covers.

Example 3.12 (a). In this example variable x_i is represented by i, x_i' by i'. Let $G = 1'2+3'4+12'4'5$. Then $G' = (1+2')(3+4')(1'+2+4+5') = 1'2'3+1'2'4'+123+124'+134+2'34+135'+14'5'+2'35'+2'4'5'$; all cubes are maximal. We call G'_a this complement of G, for future reference.

Another, less obvious, division of complementation labor is based on the identity $G' = x(G_x)' + x'(G_{x'})'$, proven in Theorem 3.9 in the Appendix. The same proof shows that $(G_x)' = (G')_x$, so both can be denoted by G'_x.

Example 3.12 (b). The complement G'_b of G obtained by Boolean decomposition: $G'_b = 1G'_1+1'G'_{1'} = 1[4G'_{14}+4'G'_{14'}]+ 1'[2G'_{1'2}+2'G'_{1'2'}] = 1[4(3')'+4'(2'5)']+ 1'[20+2'(3'4)'] = 143 + 14'2 + 14'5'+ 1'2'3 + 1'2'4'$. Note that while $G'_a \sim G'_b$, G'_b contains only half as many cubes as G'_a does. To see why, we obtain G'_b from G'_a by applying the rule $xp+x'q+pq \sim xp+x'q$ (the rule itself holds because both sides are equivalent to p when $x = 1$, and to q when $x = 0$).

$G'_a = 1(23+24'+34+35'+4'5')+ 2'(1'3+1'4'+34+35'+4'5') \sim 1(24'+34+4'5')+ 2'(1'3 + 1'4' + 34 + 4'5') = 3(14 + 2'4 + 1'2')+ 4'(12 + 15' + 1'2' + 2'5') \sim 3(14 + 1'2')+ 4'(12 + 15' + 1'2') = G'_b$.

It is no accident that Boolean decomposition obtains a shorter complement of G without the extra labor of applying the rule $xp + x'q + pq = xp + x'q$. To see why, let $G = xp+x'q+pq$. Then $G'_a = (x'+p')(x+q')(p'+q') = (x'+p')(q'+xp') = x'q'+xp'+p'q'$; while $G'_b = x(G_x)' + x'(G_{x'})' = x(p + pq)'+ x'(q + pq)' \sim xp' + x'q'$. Hence, the essential difference between the two methods is that the Boolean decomposition method, in the process of eliminating nonmaximal cubes, will also apply (without any extra effort) this rule; while the multiplication method, simply because it divides the problem differently, will not. For these reasons, for nonmonotonic covers, only complementation by Boolean decomposition will be further analyzed.

An obvious point is that labor should be divided first along nonmonotonic variables, so that the rule $xp+x'p+pq = xp+x'p$ gets a chance of being costlessly applied. Another obvious point is that, among nonmonotonic variables, labor should be divided first along the variable appearing in most cubes, so that cofactors share as few cubes as possible, and unnecessary duplication of labor is avoided. Finally, in case of a tie, the variable

that minimizes the difference between positive and negative instances in G should be preferred, to produce cofactors with the least size difference. This is to minimize the labor of eliminating nonmaximal cubes from each cofactor. A variable that satisfies these three requirements (nonmonotonic, most instances in G, most balanced instances) will be called "appropriate" in the next definition.

How should monotonic covers be complemented? The method based on multiplication can take advantage of the MA multiplication techniques contained in Definition 3.17. The method based on Boolean decomposition can take advantage of the identities $G' = xG'_x + G'_{x'}$ if G is monotonic decreasing in x; and $G' = x'G'_{x'} + G'_x$ if G is monotonic increasing in x (see Theorem 3.10 in the Appendix). The examples that follow show why the multiplication method is likely to take less work on monotonic covers.

Example 3.13. Let $G = x'y' + u'v'w'$ (cubes share no variables). Using the multiplication method $G' = (x + y)(u + v + w) = xu + xv + xw + yu + yv + yw$. Using the Boolean decomposition method, $G' = xG'_x + G'_{x'} = x(u'v'w')' + (y' + u'v'w')' = x(u + v + w) + H'$. $H' = yH'_y + H'_y = y(u'v'w')' + (1 + u'v'w')' = y(u + v + w)$. Boolean decomposition is less efficient than multiplication because it generates needlessly the term $(1 + u'v'w')'$ that has to be detected and eliminated. Any other choice of decomposition variables will also generate superfluous terms.

Example 3.14. Let $G = x'y'z' + y'w'$ (cubes share some variables). Then, using the multiplication method, $G' = (x + y + z)(y + w) = y + xw + zw$. Boolean decomposition, instead, works as follows: $G' = yG'_y + G'_{y'} = y0' + (x'z' + w') = y + H' = y + wH'_w + H'_{w'} = y + w(x'z')' + (x'z' + 1)' = y + w(x + z) = y + wx + wz$. The term $(x'z' + 1)'$ is again superfluous, and any choice of decomposition variables will generate superfluous terms.

□

Definition 3.19. Complementation reduced to Boolean decomposition and multiplication. Let G be an cover, and G' its complement. Then,

1. If $1 \in G$, then $G' = 0$.
   If $G = 0$, then $G' = 1$.
2. If G contains exactly one cube $p = \Pi_{j=1}^{n} x_j^{A_j}$, then $G' = p' = \sum_{j=1}^{n} x_j^{A'_j}$.
3. If G is monotonic, then $G' = \Pi_{p \in G} p'$, where multiplications are performed as indicated in Definition 3.17.
4. If G is nonmonotonic, then for some appropriate variable x, $G' = xG'_x + x'G'_{x'}$.

□

European University Institute
Economics Department

EUI Working Paper **ECO** No. 97/35

Managing Design Complexity
to Improve on Cost, Quality, Variety,
and Time–to–Market Performance Variables

Spyros Vassilakis

Part B — pp. 30-54

Printed in Italy in December 1997
European University Institute
Badia Fiesolana
I – 50016 San Domenico (FI)
Italy

The discussion so far illustrates some points relevant to design for cost reduction. Problem representation is important. The most compact, implicit representations (covers) require more search to generate the desired result (primes), because they "hide" information from the algorithm. The less compact, more explicit representations (cover of minterms, complementary covers) are costly to produce and store; but admit more specialized algorithms that spend less of their time in unproductive search (exploring blind alleys).

Division of labor (iterative, divide–and–conquer, "inductive generalization", dynamic programming, Boolean decomposition) is important for efficiency. A good division of labor will avoid solving the same problems twice; generating partial results that will be discarded later; using a general method when a more efficient, specialized method applies due to special properties of the problem at hand; ordering decisions in such a way that partial results are unnecessarily complicated, or not as informative as possible.

Finding good problem representations and good divisions of labor are ill–structured problems, in the sense that they cannot be usefully formulated as optimization problems.

The performance difference between good and not so good problem representations and divisions of labor is quantitatively important, and can be exponential in some parameter of the design problem. It seems, though, that no problem representation and division of labor is uniformly better on all design problems. For example, when the design problem represents a symmetric, nonmonotonic function, methods that exploit special properties of symmetry rather than monotonicity will do better.

4 Elimination of Redundant Primes

In this section, we are given a compact, maximal cover F and we look for a minimum cardinality subcover H of F such that $H \sim F$; it is implicitly understood, but irrelevant for this section, that $F = \pi(G)$ for some design problem G. Large saving in product cost can be realized at this stage, as evidenced by Example 3.3. These savings, however, can easily be exceeded by the cost of searching for redundant components; efficient search is thus important. All methods described will first represent in some way the covering relationships among cubes in F, i.e. the functions fulfilled by each component and any redundancies present (functions fulfilled by more than one component). They will then try to extract the smallest set of components that fulfills all functions of F. We first discuss the Quine–McCluskey (QM) problem representation, that is conceptually simple

but not as efficient as possible.

Example 4.1. Let $F = xyz+x'uv+yzuv$. QM represents the functions of each component in the most explicit way, namely by listing the minterms covered by each cube. Using the notation $K_x = x + x'$, $K_{xy} = xK_y + x'K_y$, $K_{xyz} = xK_{yz} + x'K_{yz}$, etc., for the sets of minterms of one, two, three, ..., variables, respectively, we have, in this example: $M(xyz) = xyzK_{uv} = xyz(uK_v + u'K_v) = xyz(uv + uv' + u'v + u'v')$; $M(x'uv) = x'uvK_{yz}$; $M(yzuv) = yzuvK_x$. Note that the first two sets contain four minterms each, while the last contains two. QM then eliminate redundant components by solving a set–covering problem, namely they find a minimum cardinality subset H of F such that $\cup_{p\in H} M(p) = \cup_{p\in F} M(p)$. This is done by representing the problem in matrix form. Let N be the number of minterms in $b(F)$ (8 in this example), and let K be the number of cubes in F (3 in this example). Form the $N \times K$ matrix A by setting $A_{mp} = 1$ if $m \in M(p)$, i.e. if minterm m is covered by cube p, $A_{mp} = 0$ otherwise. Let x_p be a binary variable with the following interpretation: $x_p = 1$ means that component $p \in F$ will be part of the final product, while $x_p = 0$ means that component $p \in F$ will be discarded. Then, any solution x^* of $\min \sum_{p=1}^{K} x_p$, subject to, for each $m = 1, \ldots, N$, $\sum_{p=1}^{K} A_{mp}x_p \geq 1$, is a solution of the set–covering problem, with $H = \{p : x_p^* = 1\}$. This is because each constraint guarantees that the corresponding minterm is covered by some cube in H, so that $H \sim F$; and minimization guarantees that H is of minimum cardinality. In this example, the (three) columns of A are 1111000000, 0000111100, 10001000; the unique solution is $H = \{xyz, x'uv\}$. □

It is well known that the set–covering problem (and the equivalent zero–one linear programming minimization problem) is NP–complete; a proof of this can be found in Wegener (1987, Theorem 5.1, p. 35). An implication of this is that all known methods for solving such problems take time exponential in the size of the set–covering problem, namely in Nk. It is thus important to find problem representations that reduce the number of rows and columns of A, i.e. that represent redundancies in a more compact, implicit way. The main idea is that covering relationships between cubes in F can be discovered and recorded directly, without listing the minterms covered by each cube. To do this, we need the following definitions and results.

Definition 4.1. A cover F in n variables is a tautology if $b(F) = D^n$. □

The covers $F_0 = x'$, $F_1 = x$, for instance, are not tautologies, since $b(x') = \{0\}$, $b(x) = \{1\}$; the cover $F = x + x'$ is a tautology since $b(F) = b(x) \cup b(x') = \{0, 1\} = D$.

Definition 4.2. Let $p = \Pi_{j=1}^{n} x_j^{A_j}$, $q = \Pi_{j=1}^{n} x_j^{B_j}$ be two cubes. The restriction of p on q,

is the cube p_q obtained from p by setting $x_j = t$ if $B_j = \{t\}$, $t = 0, 1$. Equivalently, if $d(p, q) \geq 1$, then $p_q = 0$; while if $d(p, q) = 0$, then $p_q = \Pi_{j=1}^{n} x_j^{A_j \cup B'_j}$. □

If $p = xy$, $q = uvw$, $s = xu$, $t = x'vw$, for instance, then $p_q = p, q_p = q$; $p_s = y$, $s_p = u$; $p_t = 0 = t_p$; $q_s = vw$, $s_q = x$; $q_t = u$, $t_q = x'$; $s_t = 0 = t_s$.

Definition 4.3. Let F be a cover and q a cube. Then the restriction of F on q, F_q, is the cover $\sum_{p \in F} p_q$. □

If $F = xy + uvw + xu$, and $t = x'vw$, for instance, then $F_t = u$.

Theorem 4.1. Let F be a cover and q a cube. Then $q \leq F$ iff F_q is a tautology. □

In the previous example, F_t is not a tautology; to see that $t \not\leq F$, note that $xyuvw = 00011$ belongs to $b(t)$ but not to $b(F)$. For $q = xu \leq F$, however, $F_q = y + vw + 1 \sim 1$.

We can now describe the construction of a reduced–size set–covering problem. This will be first done in Examples 4.2 and 4.3.

Example 4.2. Let $F = xyz + x'uv + yzuv$; name the cubes $1, 2, 3$ in the order they appear. The notation $F \backslash p$ will denote F without p; for example $F \backslash xyz = x'uv + yzuv$. We test whether each cube p in F is covered by the other cubes in F, by testing whether $(F \backslash p)_p$ is a tautology. For example $(F \backslash 1)_1 = uv$ is not a tautology, i.e. xyz is not covered by the remaining cubes, and has to be part of any solution of the set–covering problem. Similarly, $(F \backslash 2)_2 = yz$ is not a tautology, and $x'uv$ has to be part of any solution. Finally, $(F \backslash 3)_3 = x + x'$ is a tautology, i.e. $yzuv$ is covered by the other cubes in F. It is the case in this example that the cubes covering $yzuv$, namely xyz and $x'uv$, will be part of any solution, so $yzuv$ can safely be discarded. Note that we did not have to generate, store, or process the minterms covered by each cube (ten in all in this example).

In Example 4.2, a solution was found without representing the problem in matrix form. In general, the matrix form is unavoidable as a way to record mutual covering relationships. The next example illustrates the construction of such a (reduced–size) covering matrix.

Example 4.3. Let $F = xu + yu + zu + x'y + y'z + z'x + xv + uv' + vy$; the cubes of F are named $1, 2, \ldots, 9$ in the order they appear. We first compute, for each p in F, $(F \backslash p)_p$. If $(F \backslash p)_p$ is not a tautology, then p is called relatively essential; recall that in this case p is not covered by the remaining cubes in F, and will thus be part of any solution. In this example, cubes $4, 5, 6, 7, 8$ are relatively essential; we call this set E. If, on the other hand, $(F \backslash p)_p$ is a tautology, then we record the minimal sets of cubes covering

p. For example $(F\backslash 1)_1 = y + z + yz + z' + v + v' + yv$. After eliminating nonmaximal cubes, we have $(F\backslash 1)_1 = z + z' + v + v'$; we also record the source of each term in this expression. For example, $z = 3_1$, $z' = 6_1$, $v = 7_1$, $v' = 8_1$, where $3_1 = zu|_{xu}$, etc. Since $z + z' \sim 1$, $v + v' \sim 1$, cube 1 is covered either by cubes $\{3, 6\}$, or by cubes $\{7, 8\}$, or both. We record this information by writing $\varphi_0(1) = \{36, 78\}$. We obtain similarly $\varphi_0(2) = \{14, 89\}$, $\varphi_0(3) = \{25\}$, $\varphi_0(9) = \{47\}$. This is the end of the first stage in the construction of the reduced–size covering matrix. In the second stage, we discard from each $\varphi_0(p)$ all relatively essential cubes, to obtain $\varphi_1(p) = \varphi_0(p)\backslash E$. In this example, $\varphi_1(1) = \{3\}$, $\varphi_1(2) = \{1, 9\}$, $\varphi_1(3) = \{2\}$, $\varphi_1(9) = \emptyset$. All relatively essential cubes will be present in all solutions, so it is not useful to record cubes covered by them. We then form the set $T_1 = \{p : \varphi_1(p) = \emptyset\}$ of totally redundant cubes, i.e. of cubes that are covered solely by relatively essential cubes; such cubes can safely be discarded. Finally, we iteratively eliminate totally redundant cubes using the equations $\varphi_{t+1}(p) = \varphi_t(p)\backslash T_t$, $T_{t+1} = \{p : \varphi_{t+1}(p) = \emptyset\}$, until the first k, such that $T_k = \emptyset$. In this example, $k = 2$, $T_1 = \{9\}$, $\varphi_2(1) = \{3\}$, $\varphi_2(2) = \{1\}$, $\varphi_2(3) = \{2\}$, and $T_2 = \emptyset$. The set of totally redundant cubes is $T = \cup_{t=1}^{k} T_t = \{9\}$. The set $R = F\backslash(E \cup T)$ is called the set of partially redundant cubes, and forms the columns of the covering matrix A. In this example, $R = \{1, 2, 3\}$. While relatively essential cubes have to be in the product, and totally redundant cubes can safely be discarded, partially redundant cubes stand in mutually covering relationships with each other, given by functions $\varphi(p) = \varphi_k(p)$, and their removal has to take them into account. To do this, we take the graph of function φ, where graph $(\varphi) = \{(r, C) : C \in \varphi(r)\}$. In this example, graph $(\varphi) = \{(1, 3), (2, 1), (3, 2)\}$. We create a row of A for each element of this graph. We finally define $A_{rC,p} = 1$ if $p = r$ or $p \in C$, $A_{rC,p} = 0$ otherwise, where $r, C \in$ graph (φ), $p \in R$. In this example, the rows of A are 101, 110, 011. The first row, for instance, means that either cube 1 or cube 3 have to be in the product, because 1 is a required product function $(1 \in F)$, and if 1 is omitted, cube 3 (and the relatively essential cubes) will fulfill its function. The covering problem is then $\min x_1 + x_2 + x_3$ subject to $x_1 + x_3 \geq 1$, $x_1 + x_2 \geq 1$, $x_2 + x_3 \geq 1$, $x_j = 0, 1$. All solutions have value two; one solution is $x_1 = x_3 = 1$. The corresponding solution of the overall problem is $\{1, 3\} \cup E = \{1, 3, 4, 5, 6, 7, 8\}$; cubes 2 and 9 have been eliminated. Note that the covering matrix is of size 3×3; while the corresponding QM covering matrix is of size 28×9, where 28 is the number of minterms in $b(F)$ and 9 the number of cubes in F. □

The correctness of this method is established by

Theorem 4.2. Let F be a compact, maximal cover. Let A be its reduced–size covering matrix, E the set of its relatively essential cubes, and R the set of its partially redundant

cubes, as defined in Example 4.3. Let x^* solve the 0–1 linear programming problem $\min \sum_{p \in R} x_p$ subject to $Ax \geq 1$, where 1 is a vector of 1s. Then $H = E \cup \{p \in R : x_p^* = 1\}$ is a minimum–cost cover equivalent to F.

The 0–1 LP problem can be solved by standard methods that will not be covered here; see De Micheli (1993, p. 91, algorithm 2.5.4) for a branch and bound algorithm, and Balas and Ho (1980) for a cutting–plane algorithm.

In all the examples covered here, it was easy to check whether each cover F_q was a tautology or not, since each F_q was a short expression. In general, checking whether a cover is a tautology is an NP–complete problem. Recall that tautology checking is needed to avoid solving impossibly large set–covering problems of dimension $|b(F)| \times |F|$, where F is the given prime cover. What the reduced–size A problem representation does, then, is to avoid solving one very large NP–problem (set–covering of dimension $(b(F)) \times |F|$); by solving several smaller–size NP–complete problems (tautology checking for each F_q; setcovering for the reduced–size covering matrix A). This strategy is likely to generate large cost savings, precisely because tautology–checking and set–covering are worst–case exponental: The cost of solving N problems of size n_i each, namely $\sum_{i=1}^{N} \exp(n_i)$, is much smaller than the cost of solving one problem of size $\sum_{i=1}^{N} n_i$, namely $\exp(\sum_{i=1}^{N} n_i)$. Since this strategy depends on performing $|F|$ tautology checks, it is important that tautology–checking is done efficiently, even if it is a backroom operation for product design.

Efficient tautology checking will be based on two pieces of knowledge. First, on our knowledge of special properties of covers that make tautology checking easy. Secondly, on divisions of labor that first split the original cover into smaller covers until covers with these special properties are obtained; and then put the parts back together for the purposes of tautology checking.

In what follows, the notation $F \rightarrow G$ means that for the purposes of tautology checking F can be replaced by G; $F \rightarrow 1$ means that F is a tautology, and thus can be replaced by 1; $F \rightarrow 0$ means that F is not a tautology, and thus can be replaced by 0.

Theorem 4.3. Covers easy to check for tautology.

B_1. $F + 1 \rightarrow 1$.

B_2. $\emptyset \rightarrow 0$, where $\emptyset$ is the empty cover.

M. $F \rightarrow m(F)$, if F has at least one monotonic variable; $m(F)$ is the subcover of F consisting of all cubes that do not contain monotonic variables.

Proof. The first two are obvious. For the third, assume that F has monotonic variables. We show that F is a tautology iff $m(F)$ is. Since $m(F)$ is a subcover of F, one direction is clear. Suppose now, that F is a tautology and that, for contradiction, $m(F)$ is not. Then there exists a vector u of values of nonmonotonic variables such that $m(F)(u) = 0$. Let v be a vector of values for monotonic variables, defined as follows: $v_x = 0$ if x appears uncomplemented in F; $v_x = 1$ if x appears complemented in F. Finally, let $K(F)$ be the cubes of F that contain monotonic variables. Then $F(u, v) = K(F)(u, v) + m(F)(u) = K(F)(u, v) + 0 = K(F)(u, v) = 0$, a contradiction. The last step follows from the fact that, at v, every cube containing monotonic variables evaluates to 0. □

The cover $G = xy' + xuv + y'u'v'$, for instance, contains two monotonic variables, namely x and y; all its cubes contain x and y, hence $m(G) = \emptyset$. Then, by Theorem 4.3, $G \stackrel{M}{\rightarrow} m(G) = \emptyset \stackrel{B}{\longrightarrow} 0$, i.e. G is not a tautology; in fact, $G(xy = 01) = 0$.

In this paper I will examine three kinds of division of tautology–checking labor. The first one, disjunctive division of labor, is based on the following result.

Theorem 4.4. Let $F = G + H$, where G and H do not share variables. Then F is a tautology iff at least one of G, H is a tautology.

Proof. Suppose that F is a tautology but, for contradiction, neither G nor H are tautologies. Let $S(F)$, $S(G)$, $S(H)$ be the variables in F, G, H, respectively. By assumption $S(F) = S(G) \cup S(H)$, $S(G) \cap S(H) = \emptyset$. Let $u \in D^{S(G)}$, $v \in D^{S(H)}$ be such that $G(u) = 0$, $H(v) = 0$. Then $(u, v) \in D^{S(F)}$ and $F(u, v) = G(u) + H(v) = 0$, a contradiction.

Definition 4.4. The graph of cover F has the cubes of F as nodes; there is an edge between two cubes iff they share variables.

The graph of $F = xy + x'z + uv$, for instance, has one edge, namely $(xy, x'z)$. □

Theorem 4.4 and Definition 4.4 suggest the following division of labor: If the graph of F has more than one connected components, check each component for tautology separately. Then declare F a tautology if at least one component is a tautology; and declare F a nontautology if no component is a tautology.

Definition 4.5. Disjunctive division of labor.

D. $F \longrightarrow O_K(F_1, \ldots, F_K)$, if $F_1, \ldots, F_K$, $K \geq 2$, are the connected components of the graph of F.

O. For each $K \geq 2$

$O_K(s_1, \ldots s_K) \longrightarrow 1 \quad$ if $\quad s_i = 1 \quad$ for some $\quad i = 1, \ldots, K$

$O_K(s_1, \ldots s_K) \longrightarrow 0 \quad$ if $\quad s_i = 0 \quad$ for all $\quad i = 1, \ldots, K.$ □

The cover $F = x + y$, for instance, has two connected components, namely $F_1 = x$ and $F_2 = y$. Each F_i is monotonic, so $m(F_i) = \emptyset$. Hence, $F \xrightarrow{D} O_2(x, y) \xrightarrow{M} O_2(\emptyset, \emptyset) \xrightarrow{B} O_2(0, 0) \xrightarrow{O} 0$, i.e. F is not a tautology. □

Conjunctive division of labor is based on the fact that a cover is a tautology iff both its cofactors are.

Theorem 4.5. A cover F is a tautology iff, for any variable x, both F_x and $F_{x'}$ are tautologies.

Proof. Let F be a tautology. Let u be a vector of values for all variables except x. Then $F_x(u) = F(1, u) = 1$; $F_{x'}(u) = F(0, u) = 1$. Hence, both F_x and $F_{x'}$ are tautologies.

For the converse, let both F_x and $F_{x'}$ be tautologies. Then for each $w \in D^n$, the identity $F = xF_x + \bar{x}F_{\bar{x}}$ yields $F(w) = F_x(w_{-x}) = 1$ if $w_x = 1$; $F(w) = F_{\bar{x}}(w_{-x}) = 1$ if $w_x = 0$. Hence F is a tautology.

Recall, for the next definition, that an appropriate variable of F is a nonmonotonic variable that appears in most cubes of F; in case of a tie, a variable that minimizes the difference between positive and negative instances in F.

Definition 4.6. Conjunctive division of labor.

C. $\quad F \longrightarrow A(F, F_x, F_{x'})$, where x is an appropriate variable.

A. $\quad A(F, 1, G) \longrightarrow G, \qquad A(F, G, 1) \longrightarrow G.$

$A(F, 0, G) \longrightarrow 0, \qquad A(F, G, 0) \longrightarrow 0.$ □

The cover $F = y + z + y'z'$, for instance, contains no monotonic variables; and its graph is connected. Hence, none of the rules B, M, D, O, apply. Using conjunctive division of labor, however, we obtain $F \xrightarrow{C} A(F, 1 + z, z + z') \xrightarrow{B} A(F, 1, z + z') \xrightarrow{A} z + z' \xrightarrow{C} A(z + z', 1, 1) \xrightarrow{A} 1$. Hence, F is a tautology.

The last division of labor to be presented here is motivated by covers like $F = xy + xy' + x'y + x'y' + x'yu'v + xyuv'$. Only conjunctive divisions of labor apply to F. Any conjunctive division of labor is wasteful, because F is a tautology by virtue of its first four cubes only; while any cofactor of F involves the irrelevant last two cubes of F. What is needed is a criterion that isolates subcovers like $G = xy + xy' + x'y + x'y'$. To visualize it, consider the bipartite graph $BG(F)$ of F, with nodes $F \cup S(F)$; and edges

(p, x) for any $p \in F$, $x \in S(p)$. Its incidence matrix is given by

	x	y	u	v
xy	1	1	0	0
xy'	1	1	0	0
$x'y$	1	1	0	0
$x'y'$	1	1	0	0
$x'yu'v$	1	1	1	1
$xyuv'$	1	1	1	1

Note that G is associated with the largest block of zeros in the incidence matrix. Equivalently, if $A_1 = G \cup S(G)$, $A_2 = (F \cup S(F)) \backslash A_1$, then (A_1, A_2) is a minimum cut of $BG(F)$; in the sense that (A_1, A_2) is a partition of $F \cup S(F)$ that minimizes the number of edges connecting nodes in A_1 to nodes in A_2. Minimum cuts can be efficiently computed by standard network flow algorithms (Cormen et al., 1990, ch. 27). Given a cut (A_1, A_2), let $\mu(A_i) = \{p \in A_i \cap F, s(p) \cap A_j = \emptyset\}$. Then the desired subcover $\mu(F)$ of F is the largest of $\mu(A_1)$, $\mu(A_2)$. In the example, $\mu(A_1) = G$, $\mu(A_2) = \phi$, $\mu(F) = G$.

The theorem that follows shows how to use the subcovers $\mu(F)$, $\nu(F)$ for tautology checking.

Definition 4.7. Let F be a cover, and (A_1, A_2) a minimum cut of $BG(F)$. Then $\mu(F)$ is the largest of $\mu(A_1)$, $\mu(A_2)$. The cover $\nu(F)$ consists of all cubes obtained from cubes in $F \backslash \mu(F)$ by dropping all literals involving variables in $\mu(F)$.

The value of F, a measure of F's decomposability, is given by $v(F) = \frac{|\mu(F)|}{|F|} \times \frac{(|S(F)| - |S(\mu(F))|)}{|S(F)|}$. □

In the example, $\nu(F) = uv' + u'v$, $v(F) = \frac{4 \times 2}{6 \times 4} = \frac{1}{3}$. □

Theorem 4.7. Let F be a cover. Then

(a) F is a tautology if $\mu(F)$ is a tautology.

(b) F is not a tautology if neither $\mu(F)$ nor $\nu(F)$ are tautologies.

(c) If $\mu(F)$ is not a tautology and $\nu(F)$ is a tautology, then F may or may not be a tautology.

Proof.

(a) Obvious, because $\mu(F) \leq F$.

(b) Since neither $\mu(F)$ nor $\nu(F)$ are tautologies, there exist vectors u, v, $u \in D^{S(\mu(F))}$, $v \in D^{S(\nu(F))}$ such that $\mu(F)(u) = 0 = \nu(F)(v)$. Since $S(\nu(F)) = S(F) \backslash S(\mu(F))$, $(u, v) \in D^{S(F)}$, and $F(u, v) = \mu(F)(u) + (F \backslash \mu(F))(u, v) = (F \backslash \mu(F))(u, v) \leq \nu(F)(v) = 0$, i.e. F is not a tautology. The last inequality follows from the fact that each cube in $\nu(F)$ is derived from a cube in $F \backslash \mu(F)$ by dropping all literals involving variables in $\mu(F)$.

(c) Consider the covers $F_1 = x + x'z + x'z'$, $F_2 = xu + yu' + xy$. Then $\mu(F_1) = x$, $\nu(F_1) = z + z' \sim 1$; $\mu(F_2) = xy$, $\nu(F_2) = u + u' \sim 1$. Hence, for both i, $\nu(F_i)$ is a tautology but $\mu(F_i)$ is not. Note that F_1 is a tautology, while F_2 is not.

Definition 4.8. Semidisjunctive division of labor.

S. $\quad F \longrightarrow R(F, \mu(F), \nu(F), \quad$ if $v(F) \geq \bar{v}$,
where $\bar{v}$ is a parameter between 0 and 1.

R. $\quad R(F, 1, G) \longrightarrow 1$; $\quad R(F, 0, 0) \longrightarrow 0$;
$R(F, 0, 1) \longrightarrow A(F, F_x, F_{x'})$ $\quad$ for some appropriate variable x.

Note that application of rule S is restricted by raising the value of $\bar{v}$. The rule $R(F, 0, 1) \longrightarrow A(F, F_x, F_{x'})$ changes division of labor from semidisjunctive to conjunctive when the former is inconclusive.

When $F = x + x' + xuv' + xu'v$, for instance, then $\mu(F) = x + x'$, $\nu(F) = uv' + u'v$. Hence we obtain the following derivation: $F \xrightarrow{S} R(F, x + x', uv' + u'v) \xrightarrow{C} R(F, A(x + x', 1, 1), uv' + u'v) \xrightarrow{A} R(F, 1, uv' + u'v) \xrightarrow{R} 1$.

The tautology–checking algorithm will simply order the rules presented so far, and will put some restrictions on their application. First note that each division of labor appears as a pair of rules, namely $< D, O_k >$, $< S, R >$, and $< C, A >$. The first element of each pair is a rule that decomposes covers; while the second element of each pair is a rule that combines the parts for the purposes of tautology–checking. We call D, S, C analytic rules, and O_k, R, A synthetic rules. Synthetic rules may not require evaluation of all their arguments: for example, $A(F, 0, H) \longrightarrow 0$, whatever the value of H is. Hence, to prevent useless decompositions, synthetic rules should be applied before analytic rules, and analytic rules should be applied when synthetic rules no longer apply. Secondly, no rule should apply on F when F appears as the first argument in some O_k, R, or A term, i.e. as $O_k(F, G, H)$, $R(F, G, H)$, or $A(F, G, H)$. This prevents useless rule applications, and guarantees termination. The notation $A(F, G, H)$, for instance, means that F has been conjunctively decomposed into G and H; all further rule applications

should be on G and H. Thirdly, rules B should precede all others, to avoid useless decompositions (D, C, S) or reductions (M). Fourthly, application of rules B and M should precede all others, to avoid useless decompositions. Disjunctive decomposition should precede semidisjunctive decomposition, as easier to apply; and semidisjunctive decomposition should precede conjunctive decomposition, to exploit any block structure before calculating cofactors.

Definition 4.9. Tautology–checking algorithm. Given a cover F, apply the following rules on F and on any resulting cover, in the order they appear: B, M, O, R, A, D, S, C. Do not apply any rules on a cover that is the first argument of an O, R, or A term. Do not apply any rule unless all rules preceding it have ceased to apply. Stop when either 1 (tautology) or 0 (nontautology) have been derived. □

Example 4.4. Let $F = x'y'+x'y+xy'+xyu'+xyuv'+xyuv$. F does not contain monotonic variables, and its graph is connected; $\mu(F) = x'y'+x'y+xy'$, $v(F) = 6/24 = 1/4$, $\nu(F) = u'+uv'+uv$. For $\bar{v} \leq 1/4$, we apply first semidisjunctive decomposition, to obtain $F \xrightarrow{S} R(F, \mu(F), \mu(F)) \xrightarrow{C} R(F, A(\mu(F), y', y+y'),\ A(\nu(F), v+v', 1)) \xrightarrow{A} R(F, A(\mu(F), y+y', y'), v+v') \xrightarrow{C} R(F, A(\mu(F), y+y', y'),\ A(v+v', 1, 1)) \xrightarrow{A} R(F, A(\mu(F), y+y', y'), 1)$, $\xrightarrow{C} R(F, A(\mu(F), A(y+y', 1, 1),\ A(y', 0, 1)), 1) \xrightarrow{A} R(F, A(\mu(F), 1, 0), 1) \xrightarrow{A} R(F, 0, 1)$ $\xrightarrow{R} A(F, F_x, F_{x'}) = A(F, y' + yu' + yuv' + yuv, y+y') \xrightarrow{C} A(F, F_x, A(y+y', 1, 1)) \xrightarrow{A}$ $A(F, F_x, 1) \xrightarrow{A} F_x \xrightarrow{C} A(F_x, F_{xy}, F_{xy'}) = A(F_x, uv' + uv + u', 1) \xrightarrow{A} H = uv' + uv + u'$ $\xrightarrow{C} A(H, H_u, H_{u'}) = A(H, v+v', 1) \xrightarrow{A} v+v' \xrightarrow{C} A(v+v', 1, 1) \xrightarrow{A} 1$.

For $\bar{v} > 1/4$, on the other hand, disjunctive decomposition is not allowed, and the derivation starts from $F \xrightarrow{C} A(F, F_x, F_{x'}) \longrightarrow \ldots$, i.e. from the tenth step of the previous derivation. The example shows that even in a problem with substantial block structure, semidisjunctive decomposition may be wasteful. The cubes needed to establish tautology may not all belong to one of the subcovers created by such decomposition. The choice of $\bar{v}$ is another ill–structured problem: There is no objective function that assigns to each choice of $\bar{v}$ its true benefit (net number of steps saved).

The reduction of the derivation of a minimum–cost cover to standard problems (covering, connected components, minimum cut, tautology) allows a firm to benefit from continuing improvements in algorithms that solve such standard problems, simply by buying off–the–shelf. A firm that relies exclusively on buying, however, has to accept the existing division of labor. For example, a firm that uses QM's problem representation will demand only new and improved covering algorithms; but will have no use for new and improved tautology algorithms. Only a firm that has consciously examined the way it represents problems and divides labor will be able to achieve efficiency gains that

are not available to all. Whitney (1995, p. 116) emphasizes this point in his study of Nippondenso (NDCL in the quotation):

> "Many companies see the need to implement product design using computer–aided design (CAD) or to improve their ability to assemble their products efficiently using design for assembly (DFA). Fewer see the need to be able to manufacture their products in unique ways, much less to be able to build in–house the specialized equipment necessary to do so. Fewer yet are those who see the need to write their own CAD software to tie together their own carefully groomed product–process design methodology. Fewer of all are those who see the need to do all of these. NDCL is one of the most advanced in understanding that all these actions must be taken together systematically. ... Many of NDCL's make–buy choices in technology often seem uneconomical or indicative of a not–invented–here attitude. An engineer at another Japanese firm put it bluntly: 'You learn by trying, not by buying.'"

5 Product Architecture

All products up to this point in the paper were assumed to consist of components (cubes) linked together by OR gates. The objective of design was to minimize the number of components taking this architecture as given. This section will show that choice of product architecture, i.e. of the way components are linked, is an important determinant of cost. It will also describe a way to choose a good architecture. The importance of this issue was stressed by Henderson and Clark (1990, p. 10):

> "Xerox was confronted in the 1970s with competitors offering copiers that were much smaller and more reliable than the traditional product. The new products required little new scientific or engineering knowledge, but despite the fact that Xerox had invented the core technologies and had enormous experience in the industry, it took the company almost eight years of missteps and false starts to introduce a competitive product into the market. In that time Xerox lost half of its market share and suffered serious financial problems. In the mid–1950s engineers at RCA's corporate R& D center developed a prototype of a portable, transistorized radio receiver. The new product used technology in which RCA was accomplished, but RCA saw little reason to pursue such an apparently inferior technology. In contrast, Sony,

a small, relatively new company, used the small transistorized radio to gain entry into the US market. Even after Sony's success was apparent, RCA remained a follower in the market as by introduced successive models with improved sound quality and FM capability. ... for many years Sony's radios were produced with technology licensed from RCA, yet RCA had great difficulty matching Sony's product in the marketplace. ... we define innovations that change the way in which the components of a product are linked together, while leaving the core design concepts (and thus the basic knowledge underlying the components) untouched, as architectural innovation. This is the kind of innovation that confronted Xerox and RCA. It destroys the usefulness of a firm's architectural knowledge but preserves the usefulness of its knowledge about the product's components."

Architectural issues would be of less interest if all product architectures for a given design problem G resulted in roughly the same number of components, and hence the same cost. The next well–known example shows that this is not the case.

Example 5.1. The behavior $P_n \subseteq D^n$ consists of all $0-1$ vectors that contain an odd number of ones. For example, $P_2 = \{01, 10\}$, $P_3 = \{001, 010, 100, 111\}$. It will be shown that (a) the product architecture that links cubes with OR gates requires 2^{n-1} cubes to represent P_n; (b) there is a product architecture that requires only n cubes to represent P_n. Hence, the right product architecture can result in exponential–size cost savings. I start with (b): Consider the exclusive–or (XOR) gate $\oplus$ defined by $0 \oplus 0 = 0 = 1 \oplus 1$, $1 \oplus 0 = 0 \oplus 1 = 1$, and the covers $F_n = x_1 \oplus \ldots \oplus x_n$. Let $u \in D^n$ contain an odd number of ones. Then $F_n(u) = 1$. To see this, suppose without loss of generality ($\oplus$ is commutative–associative) that the first $m = 2k+1$ components of u equal 1, and the remaining $n - 2k - 1$ components equal zero. Then

$$
\begin{aligned}
&F_n(u) = (u_1 \oplus \ldots \oplus u_m) \oplus (u_{m+1} \oplus \ldots \oplus u_n) && = \\
&(1 \oplus \ldots \oplus 1) \oplus (0 \oplus \ldots \oplus 0) && = \\
&(1 \oplus \ldots \oplus 1) \oplus 0 && = \\
&\underbrace{(1 \oplus 1) \oplus (1 \oplus 1) \ldots \oplus (1 \oplus 1)}_{k-\text{times}} \oplus (1 \oplus 0) && = \\
&(0 \oplus \ldots \oplus 0) \oplus 1 && = \\
&0 \oplus 1 = 1.
\end{aligned}
$$

Let $u \in D^n$ contain an even number of ones. Then a similar argument establishes $F_n(u) = 0$. Hence, $b(F_n) = P_n$.

To prove (a), note that each P_n contains all minterms with $1, 3, 5, \ldots, n$ ones when n is odd; and all minterms with $1, 3, 5, \ldots, n-1$ ones when n is even. Hence each P_n is already compact, because any two cubes in P_n are at distance two or greater. At the same time, each P_n is maximal, since no minterm can cover another minterm. It follows that $P_n = \pi(P_n)$ for all n. What is more, no prime is redundant, in the sense of being covered by the (logical) sum of the other primes. This is because if p, q are distinct minterms, then $d(p,q) \geq 1$ and therefore cofactors p_q and q_p equal 0. Hence, $(P_n \backslash q)_q = 0$ for all q in P_n, i.e. by Theorem 4.1, q is not covered by $P_n \backslash q$. Hence, the minimum–cost cover with behavior P_n is P_n itself. P_n, however, contains 2^{n-1} minterms, since there are 2^n minterms in n variables, and half of them contain an odd number of ones. It follows that the OR–based product architecture results in a product with 2^{n-1} components; while the XOR–based product architecture in an equivalent product with n components. □

A design problem G may include parts that can best be realized by a XOR–architecture, and other parts best realized by an OR–architecture. Detection of such parts can be done by spectral decomposition methods (Hurst et al., 1985), illustrated in the next example.

Example 5.2. Consider the design problem $G = x_1x_2' + x_1x_3' + x_1'x_2x_3$; G is a minimum–cost cover in the OR–architecture, since it is compact, maximal, and irredundant. A less expensive cover equivalent to G is $H = x_1 \oplus x_2x_3$; this can be verified by direct substitution (for example, $G(111) = H(111) = 0$). To derive H, we first derive the spectrum of G, i.e. a vector of correlation coefficients of the output values of G with constant, projection, and parity functions.

Step 1. Compute the value of G at each minterm and record it in a vector Y of length $2^n = 2^3 = 8$. Minterms are written in the form $x_3x_2x_1$, and ordered lexicographically, i.e. $x_3x_2x_1 < y_3y_2y_1$ if (a) $x_3 < y_3$ or (b) $x_3 = y_3$ and $x_2 < y_2$ or (c) $x_3 = y_3$, $x_2 = y_2$ and $x_1 < y_1$. In this example, $000 < 001 < 010 < 011 < 100 < 101 < 110 < 111$, and $y = (0, 1, 0, 1, 0, 1, 1, 0)$.

Step 2. Rewrite Y by replacing 1 by -1, and 0 by 1, to obtain $Y = (1, -1, 1, -1, 1, -1, -1, 1)$.

Step 3. Form the Hadamard matrix T^n for $n = 3$; T^1 is a 2×2 matrix with rows $(1, 1)$, $(1, -1)$; T^{n+1} is computed inductively using the formula

$$T^{n+1} = \begin{bmatrix} T^n & T^n \\ T^n & -T^n \end{bmatrix}. \qquad \text{Each } T^n \text{ is of dimension } 2^n \times 2^n.$$

The rows of T^3 are 11111111, 1−11−11−11−1, 11−1−111−1−1, 1−1−111−1−11, 1111−1−1−1−1, 1−11−1−11−11, 11−1−1−1−111, 1−1−11−111−1. Each row represents the vector of output values of a function: constant, x_1, x_2, $x_1 \oplus x_2$, x_3, $x_1 \oplus x_3$, $x_2 \oplus x_3$, $x_1 \oplus x_2 \oplus x_3$. This can be verified by direct computation, remembering the change of notation in Step 2.

Step 4. Compute the spectrum $S = T^n Y$ of Y. In this case, $S = 0404040 - 4 = (s_0, s_1, s_2, s_{12}, s_3, s_{13}, s_{23}, s_{123})$. The components of S are correlation coefficients: $s_0 = 0$ means there is no correlation between Y and a constant function, $s_{123} = -4$ means that Y and $x_1 \oplus x_2 \oplus x_3$ are negatively correlated, etc.

Step 5. Maximize the values of the zero and first–order coefficients s_0, s_1, s_2, s_3 by permuting the values of S. To explain this, we need two facts. First, if every instance of x_i in a cover G is replaced by $x_i \oplus x_j$, $j \neq i$, then the spectrum of G changes in a precise way: the values of s_i and s_{ij}, s_{ik} and s_{ijk}, s_{ikt} and $s_{ijkt}, \ldots$, are interchanged. Secondly, the spectrum of $G \oplus x_i$ is obtained from that of G by interchanging the values of s_o and s_i, s_{ij} and s_j, s_{ijk} and $s_{jk}, \ldots$. The objective of maximizing the values of zero and first–order coefficients by these two kinds of permutations, then, fulfills our original objective of splitting the design problem G into a core part best realized with an OR–architecture (the cover corresponding to the spectrum with s_o, s_i maximized); and a part best realized with an XOR-architecture (the XOR transformations corresponding to these permutations that, when applied on the core cover, make it equivalent to the original cover). In this example, performing the second kind of transformation for $i = 1$, we obtain a new spectrum $S_1 = 404040 - 40$: the values of s_1 and s_0, s_{12} and s_2, s_{13} and s_3, s_{123} and s_{23} have been interchanged. We transform Y_1 back into the original domain by multiplying it with the inverse $(T^n)^{-1}$ of Hadamard's matrix, where $(T^n)^{-1} = 2^{-n}T^n$. We obtain $Y_1 = 2^{-3}T^3S_1 = 111111 - 1 - 1$. Remembering the notation of Step 2, Y_1 is the vector of output values of the cover $x_3x_2x_1' + x_3x_2x_1 \sim x_3x_2$. We finally apply on this cover the XOR transformations associated with the permutations necessary to obtain S_1, to obtain $x_1 \oplus x_2x_3$.

The example shows that spectral methods are expensive, involving as they are the creation and manipulation of $2^n \times 2^n$ matrices. At this point, no cheaper methods are known. It follows that existing product architectures are not likely to be close to optimal, even if they are part of a dominant industry design and are universally accepted. Dominant firms, as Henderson and Clark (1990) document, can make the mistake of considering their current product architecture the best possible. Writing about photolithographic alignment technology, they state: "... the technology has seen four

waves of architectural innovation. In each case the core technologies remained largely untouched. ... Yet in each case the industry leader was unable to make the transition ... the established firm invested heavily in the next generation of equipment, ... with very little success. Our analysis of the industry's history suggests that a reliance on architectural knowledge derived from experience with the previous generation blinded the incumbent firms to critical aspects of the new technology" (ibid., pp. 23–24). In fact, belief in the optimality of the current product architecture can reduce a firm's ability to recognize why a competitor has a better product: "... GCA first pronounced the Nikon stepper a 'copy' of the GCA design. Even after GCA had fully recognized the threat posed by the second–generation stepper, its historical experience handicapped the company in its attempt to develop a competitive machine. GCA's engineers were organized by component, and cross–departmental communication channels were all structured around the architecture of the first–generation system. While GCA engineers were able to push the limits of the component technology, they had great difficulty understanding what Nikon had done to achieve its superior performance" (ibid., p. 27).

Product architecture also affects cost predictability. To understand this, consider a firm that supplies a product with behavior B. A customer calls and asks for product B'; the order–handling department has to give this customer a price quotation and an expected delivery date. It seems natural to quote the price of B plus the price of a NOT gate; and to promise delivery equal to B's delivery time plus the time it takes to assemble the extra NOT gate. This rule seems natural because it is based on the assumption that product cost is predictable from product description. To see that this can be misleading, consider a product with behavior $B = x_1'y_1' + x_2'y_2'$. Then $B' = (x_1 + y_1)(x_2 + y_2) = x_1x_2 + x_1y_2 + x_2y_1 + x_2y_2$. In general, if $B = \sum_{i=1}^{n} x_i'y_i'$, then B' requires 2^n components x_iy_j in the sum of products architecture, but only n components $x_i + y_i$ in the product of sums architecture. A firm that wants to provide fast and reliable customer service must discover off–line where product architecture matters. An example of this is provided by Stalk and Hout (1990). They discuss heavy vehicles, where ""... it takes 45 days to prepare an order for assembly, but only 16 hours to assemble each vehicle" (ibid., p. 76). In the 1980s, new companies (Freightliner and Paccar) took market share away from GM, Ford, and Mack because "... they delivered faster and handled product variety better than traditional producers could. In fact, many traditional firms gave price incentives to customers who would limit the custom features they ordered." (ibid., p. 174). These authors then stress the importance of predictability in providing fast and reliable price and time–of–delivery information to customers: "The most important improvement in support systems has been the pre–engineering of a variety of truck combinations. Before the streamlining effort, truck

assemblers had custom–engineered most of the orders after they received them. Some orders demanded more engineering than others, causing a lumpy flow of on–line work. As a result, the custom engineering was hasty, which led to errors and rework. Freightliner decided to invest heavily in pre–engineering hundreds of combinations of components and truck styles so that nearly all orders would be from a pretested menu. They were able to eliminate lumpy and hasty work. This dramatically collapsed the processing time on the order before it got to the assembly plant. In recent years, nearly all heavy–duty, on–highway truck producers have followed Freightliner's and Paccar's changes."

6 Product Quality

Recall from the Introduction that the main puzzle about quality is that some firms seem able to offer products with fewer defects and lower cost than their competitors: elimination of waste both reduces product cost and renders products easier to test. This section provides some modelling of this issue.

Definition 6.1. Let G be a cover. A fault in G can be either a variable in some cube that is stuck at some value; or a cube whose output is stuck at some value; or G itself, when its output is stuck at some value. □

Example 6.1. Let $F = xy + xy' + xz + x'y$. Suppose first that x is stuck at 1 in xy'. The faulty F is then $F_f = xy + y' + xz + x'y$. The behavior of faulty F on $xyz = 000$, ($F_f(000) = 1$) is different from desired behavior ($F(000) = 0$). The vector $xyz = 000$ is called a test for this fault. Similarly, if the output of cube xy is stuck at zero, the corresponding faulty cover is $F_f = xy' + xz + x'y$; $xyz = 110$ is a test vector for this fault. Finally, if the output of F is stuck at 1, any vector in the complement of its behavior is a test; while if F is stuck at 0, any vector in $b(F)$ is a test.

Consider now the faulty cover $F_f = xy + xy' + x'y$ resulting from xz being stuck at zero; F_f is equivalent to F, because xz is covered by $xy + xy'$. Hence, no test vector exists for this fault, because xz is redundant. Similarly, if x' is stuck at one in cube $x'y$, then the resulting faulty cover $F_f = xy + xy' + xz + y$ is equivalent to F, because y is covered by F. Again, no test vector exists for this fault, because $x'y$ is not a prime of F. □

Definition 6.2. Let F be a cover in n variables and F_f the faulty cover corresponding to some faults in F. A test vector for these faults is a vector u in D^n such that $F(u) \neq F_f(u)$. A cover F is single–fault testable if for any single fault in F there is a test vector. A

cover F is multiple–fault testable if, for any combination of faults in F, there is a test vector. □

Recall that a cover in n variables accepts 2^n inputs (the number of vectors in D^n). One could test a cover by comparing its actual behavior to desired behavior at each u in D^n. This is too costly even for moderate values of n; for example, when $n = 117$ and $2^{17} \sim 131,000$ tests per second are performed, it would take 2^{67} millennia to perform these tests. The main task of design is to generate the smallest possible set of test vectors that can alert us to the presence of any kind of fault. Hill and Peterson (1993, p. 456) state this concisely: "Testing is part of manufacturing. Test generation is properly part of the design process." We first characterize the covers that are single–fault testable. Recall that a cover is irredundant if none of its cubes is covered by the rest of its cubes, i.e. if no $(F \backslash p)_p$ is a tautology.

Theorem 6.1. A cover F is single–fault testable if, and only if, it is prime and irredundant. In particular, any minimum–cost cover is single–fault testable.

Proof. The second statement of the theorem follows from the fact that minimum–cost covers are prime (they are constructed to be so) and irredundant (this is a necessary condition to achieve minimum cost).

Suppose F is prime and irredundant. Faults in the output of F are easily testable: $v \in b(F)$ tests a stuck at 0 fault, and $v \notin b(F)$ tests a stuck at 1 fault. Consider a fault in literal ℓ in cube p, $p \in F$. The fault "ℓ in p is stuck at zero" is as testable as the fault "p's output is stuck at zero", so we will consider it later.

Suppose, for contradiction, that the fault "ℓ in p is stuck at one" is not testable: let $p = \ell q$, $F = p + G$, $F_f = q + G$. By the contradiction hypothesis, $F_f \sim F$. Hence, $q \leq q + G \sim F$, i.e. $q \leq F$; this contradicts the primality of $p = \ell q$.

Consider now faults in cube outputs. If p is stuck at one, then F is stuck at one, so this fault is testable by any $v \notin b(F)$. If p is stuck at zero, then $F_f = G$. Suppose, for contradiction, that this fault is not testable; then $G = F_f \sim F$. Hence, $p \leq F \sim G$, i.e. $p \leq G$, a contradiction to the irredundancy of F.

For the converse, let F be single–fault testable. Suppose, for contradiction, that $p \in F$ is not prime. Then there exists a literal ℓ such that $p = \ell q$, $q \leq F$. The fault "ℓ is stuck at one" generates the faulty cover $F_f = q + G$. Since $F = p + G$ and $q \leq F$, we have $F_f = q + G \leq F + G = F$; on the other hand, $p = \ell q \leq q$, so $F = p + G \leq q + G = F_f$. Hence, $F_f \sim F$, i.e. F is not single–fault testable, a contradiction.

Suppose, for contradiction, that F is redundant. Then there exists $p \in F$, $F = p + G$, such that $p \leq G$ (and hence $F \sim G$). Consider the fault "p's output is stuck at zero". The resulting faulty cover is $F_f = G$. Hence, $F_f \sim F$, i.e. F is not single–fault testable, a contradiction.

This theorem is valuable because it allows the construction of a complete set of test vectors much smaller in size than 2^n; these vectors test each cube for primality and irredundancy.

Theorem 6.2. A prime and irredundant cover F in n variables has a complete set of test vectors for single faults, of cardinality $(n+1)|F| + 2$.

Proof. The faults "F is stuck at d", $d = 0, 1$ are detectable by any $v \in b(F)$, for $d = 0$, and by any $v \notin b(F)$, for $d = 1$. Hence, we need two test vectors for these faults.

The fault "ℓ is stuck at one in p", where $F = p + G$, $p = \ell q$, is detectable by any vector $v \in b(q) \backslash b(F)$ (the primality of p ensures that $q \not\leq F$, i.e. $b(q) \not\leq b(F)$. To see this, note that $F_f = q + G$, and that $F_f(v) = 1$ because $v \in b(q)$, while $F(v) = 0$ because $v \notin b(F)$. Hence, we need $n|F|$ test vectors for this type of fault, one for each cube and variable).

The fault "p's output is stuck at zero" is detectable by any $v \in b(p) \backslash b(G)$, where $F = p + G$. (Since p is not redundant, $p \not\leq G$, i.e. $b(p) \not\leq b(G)$.) To see this, note that $F_f = G$, $F_f(v) = 0$ because $v \notin b(G)$, while $F(v) = 1$ because $v \in b(p)$. There are $|F|$ such tests, one for each cube in F. □

It has been established so far that cost reduction in the form of waste elimination has as a byproduct full testability with respect to single fault. It is also true that, in covers that are not prime and irredundant, generating test vectors is more difficult (for those faults that are actually detectable). To see this we will need the concept of Boolean derivative. The significance of this is that quality is harder to obtain the more redundancies a cover contains. Hence, even partial elimination of redundant components reduces the cost of quality.

Definition 6.3. Let $F = p + G$, $p = \ell q$, ℓ a literal. The Boolean derivative of p with respect to ℓ is $\frac{dp}{d\ell} = p_\ell \oplus p_{\ell'} = q \oplus 0 = q$. The Boolean derivative of F with respect to p is $\frac{dF}{dp} = (1 + G) \oplus (0 + G) = 1 \oplus G = G'$. Finally, the Boolean derivative of F with respect to ℓ in p is

$$\left(\frac{dF}{d\ell}\right)_p = \frac{dF}{dp}\frac{dp}{d\ell} = G'q.$$

□

Boolean derivatives of F identically equal to zero imply that F is independent from the variable of differentiation, and hence that a fault in this variable is undetectable; this is because $x \oplus y = 0$ if, and only if, $x = y$. Hence $\frac{dF}{dp} = 0$, for instance, means that $1 + G = 0 + G$, i.e. that F is independent of the value of p. If, on the other hand, a Boolean derivative is not identically zero, it can be used to construct test vectors for faults in the variable of differentiation.

Theorem 6.3. Let $F = p + G$, $p = \ell q$, ℓ a literal. Fault "p is stuck at zero" can be detected by any vector v that satisfies $\left(\frac{dF}{dp}p\right)(v) = 1$. Fault "$\ell$ is stuck at one in p" can be detected by any vector v that satisfies $\left(\left(\frac{dF}{d\ell}\right)_p \ell'\right)(v) = 1$. If either equation has no solution, the corresponding fault is not detectable.

Proof. Let $\left(\frac{dF}{dp}p\right)(v) = 1$. Then $\frac{dF}{dp}(v) = 1 = p(v)$, i.e. $G'(v) = 1 = p(v)$, i.e. $G(v) = 0$, $p(v) = 1$. The faulty cover corresponding to p being stuck at zero is $F_f = G$. Hence, $F_f(v) = 0$, $F(v) = p(v) + G(v) = 1$. It follows that "p is stuck at zero" is detectable by v. If, on the other hand, $\left(\frac{dF}{dp}p\right)(v) = 0$ for all v, then for each v either $p(v) = 0$, or $\frac{dF}{dp}(v) = 0$, or both. If $\frac{dF}{dp}(v) = 0$, then $G'(v) = 0$, i.e. $G(v) = 1$, i.e. $F_f(v) = 1 = p(v) + 1 = p(v) + G(v) = F(v)$, i.e. "$p$ is stuck at zero" is undetectable. If $\frac{dF}{dp}(v) = 1$, then $p(v) = 0$ and $G'(v) = 1$, i.e. $G(v) = 0$. Hence $F_f(v) = G(v) = 0 = 0 + 0 = p(v) + G(v) = F(v)$, i.e. "$p$ is stuck at zero" is again undetectable.

For the other part, let $\left(\left(\frac{dF}{d\ell}\right)_p \ell'\right)(v) = 1$, i.e. $\frac{dF}{dp}(v) = 1$, $\frac{dp}{d\ell}(v) = 1$, $\ell'(v) = 1$. It follows that $G(v) = 0$, $q(v) = 1$, $v_x = \ell'$, where x is the variable involved in literal ℓ. The faulty cover corresponding to "ℓ is stuck at one in p" is $F_f = q + G$. Hence $F_f(v) = 1$, $F(v) = (lq)(v) + G(v) = \ell(v)q(v) + G(v) = 01 + 0 = 0$, i.e. v detects this fault. If, on the other hand, $\left(\left(\frac{dF}{d\ell}\right)_p \ell'\right)(v) = 0$ for all v, then for each v either $\frac{dF}{dp}(v) = 0$, or $\frac{dp}{d\ell}(v) = 0$, or $\ell'(v) = 0$, i.e. either $G(v) = 1$, or $q(v) = 0$, or $\ell(v) = 1$. Hence, if $G(v) = 0$ then either $q(v) = 0$ or $\ell(v) = 1$, so $F_f(v) = q(v) + G(v) = q(v)$, $F(v) = p(v) + G(v) = \ell(v)q(v)$. Then, if $F(v) \neq F_f(v)$, we must have $q(v) = 1$, $\ell(v) = 0$, which is not possible. Hence $F(v) = F_f(v)$, i.e. any v with $G(v) = 0$ cannot detect that ℓ in p is stuck at one. Consider now v such that $G(v) = 1$; $F_f(v) = q(v) + G(v) = 1$, while $F(v) = p(v) + G(v) = 1$, i.e. again v cannot detect this fault. Hence, this fault is not detectable. □

We can now explain why, in covers that are not prime and irredundant, the more redundancy there is, the harder it is to compute a test vector for the subset of faults that are detectable. Quite simply, redundant terms make the calculation of Boolean derivatives harder.

Example 6.3. Let $F = xy + xy' + xz + x'y$, $p = xy'$, $l = x$, $q = y'$, $G = xy + xz + x'y$. The

test vectors for "ℓ is stuck at one in p" are given by $\frac{dF}{dp}\frac{dp}{d\ell}\ell' = G'q\ell' = (xy+xz+x'y)'y'x' = (x'+y')(x'+z')(x+y')y'x' = (x'+y'z')(x+y')y'x' = (x'y'+y'z')y'x' = y'x'+y'x'z' \sim y'x'$. Solving the equation $y'x' = 1$ we obtain $x = y = 0$, i.e. $xyz = 000$, 001 are the test vectors for this fault. If, on the other hand, we had eliminated waste from F we would get $H = x + y$. ($xy + xy' \sim x$, $x + xz \sim x$, $x + x'y \sim x + y$.) Checking for "x is stuck at 1" involves computing $\frac{dH}{dx}\frac{dx}{dx}x' = y'x'$. We have obtained the same test vectors with a much shorter computation.

The next theorem shows that a prime and irredundant cover is also multifault–testable, by the same test vectors that form a complete test set for single faults. This generates a further economy in testing, since the number of multifaults is much larger than the number of single faults.

Theorem 6.4. A cover F is multifault–testable if, and only if, it is prime and irredundant.

Proof. If F is multifault–testable, it is also single–fault testable; hence, by Theorem 6.1, prime and irredundant.

For the converse, let F be prime and irredundant. A collection of faults (i.e., a multifault) splits F into three disjoint sets, $F = L + H + G$; L is the set of cubes that lose some of their literals due to stuck–at–one faults; H is the set of cubes that disappear due to stuck–at–zero faults; and G is the set of cubes not affected by the faults in this collection. The faulty cover corresponding to this multifault is $F_f = \sum_{p \supseteq L} q(p)+G$, where $q(p) \geq p$, $q(p) \neq p$, is the cube resulting from p by dropping all stuck–at–one literals in p. If $L = \phi$, then any vector $v \in b(p)\backslash b(F\backslash p)$, $p \in H$, is a test vector for this multifault, since $F_f(v) = (F\backslash p)(v) = 0$, $F(v) = H(v) + G(v) = 1 + 0 = 1$; $b(p)\backslash b(F\backslash p)$ is nonempty by irredundancy of F. If, on the other hand, $L \neq \phi$, then any $v \in b(q(p))\backslash b(F)$, $p \in L$, is a test vector for this multifault, because $F_f(v) = q(p)(v) + \ldots = 1$, while $F(v) = 0$ because $v \notin b(F)$; $b(q(p))\backslash b(F)$ is nonempty for any $p \in L$ due to primality, which implies $q(p) \not\leq F$. □

To conclude, this section has shown that cost reduction by eliminating waste in the form of redundant components renders a cover fully testable, both for single faults and multifaults; the number of tests needed is $(n+1)|F|+2$; the precise test vectors can be generated using Boolean derivatives, whose calculation becomes easier in the absence of redundancies. Products that contain redundancies, on the contrary, are not fully testable; and the calculation of test vectors for those faults that are testable is harder, due to the presence of redundancies. Quality may not be free, but its cost decreases as a free byproduct of cost reduction through waste elimination.

7 Design for Product Variety

Recall that the literature surveyed in the Introduction claims that inexpensive variety can be achieved through product design; and that GM's inability to achieve it was due to its misdiagnosis of a design (waste–elimination) problem for an investment problem (more robots and flexible manufacturing systems). Nippondenso (NDCL) is cited in this literature as a pioneer in design for variety. Whitney (1995, pp. 117-118) describes Nippondenso's approach, and also motivates the treatment of variety in this section:

> "An important feature of NDCL's approach is avoiding complex assembly technology such as 'intelligent dexterous' robots. Instead, NDCL put as much as possible of the 'intelligence' into the product itself, by focusing the design process on supporting high–volume mixed–model JIT automated assembly. Large numbers of robots are indeed used at NDCL, but they and other complex technology are not the core of the approach. ... The difficulty of achieving high–volume model–mix JIT automated production can be put in the context of a generic, long–standing conflict in manufacturing: the flexibility–efficiency tradeoff. ... Although the flexibility–efficiency tradeoff appears alive and well in most factories, it can be beaten in two basic ways: by designing equipment so that 'wasters' are small (see Shingo, 1989) and by designing products so that 'wasters' are not needed. NDCL has used the second method: embedding flexibility in the product during the design process. ... Imagine the phone ringing each day at NDCL and a voice from Toyota demanding, 'we want 4.316 of motor type A, 301 of type B, 1.633 of type C, and 4 of type D, tomorrow morning'. The next day, totally different distributions might be ordered. One cannot possibly respond to this kind of customer by order–picking from a warehouse or by adjusting fabrication patterns. This customer at one point, however, accounted for 90% of the business and still commands more than 50%."

To understand how product design lowers the cost of variety, imagine that tomorrow Toyota will ask NDCL to supply a product in the set $\{f_0, f_1, f_2\}$, but does not know yet which. For the sake of the argument, let $f_o = x + y'z'$, $f_1 = xy + xz$, $f_2 = xy + z$. Each f_i is a minimum–cost cover; hence, as long as products are designed separately, no design activity need take place. NDCL now faces two unpleasant alternatives, namely either to manufacture in advance components x, $y'z'$, xy, xz, z and supply Toyota quickly with the product demanded, say f_0, bearing the inventory cost of the unused components

xy, xz, z; or to make components x and $y'z'$ in response to Toyota's order, keeping Toyota waiting in the meantime. There is, however, a third alternative that combines speedy response to Toyota's demands at moderate cost. It involves joint design of the possible set of products $\{f_0, f_1, f_2\}$ to maximize shared parts. The outcome of such design in this case is $f_0 = xy + xz + y'z'$, $f_1 = xy + xz$, $f_2 = xy + xz + z$. (Note the redundancies in f_0, f_2.) NDCL can now build in advance $xy + xz$, and respond quickly to Toyota's orders building $y'z'$ or z ex–post. Inventory cost is zero ($xy + xz$ is needed by all three products). The cost of modular design, or component standardization (the standardized components are xy, xz), is the redundancy introduced in f_0, f_2: "There are some circumstances under which the use of a standard component may incur higher unit costs than the use of a special component. Sometimes in an effort to standardize, firms will use a component with excess capability for a particular application." (Ulrich, 1995, p. 431). Note that this is not a necessary cost, as evidenced by the family of products $f_i = x + y_i$, $i = 1, \ldots, k$.

Product design for variety does not require any more apparatus than that developed in Section 4 for cost reduction. It is only the cover to be minimized that changes, to take into account all potential products together.

Example 7.1. Let $f_0 = x + y'z'$, $f_1 = xy + xz$, $f_2 = xy + z$. Create a new variable u that can take three values, namely 0, 1, or 2. By analogy with binary variables, for each subset A of $\{0, 1, 2\}$, u^A is a literal whose value is one if, and only if, the value of u belongs to A. The cover to be minimized, then, is $F = f_0u^0 + f_1u^1 + f_2u^2 = (x + y'z')u^0 + (xy + xz)u^1 + (xy + z)u^2 = xu^0 + y'z'u^0 + xyu^1 + xzu^1 + xyu^2 + zu^2$. Note that, by definition, $u^0 + u^1 + u^2 = 1$, $u^iu^j = 0$ if $i \neq j$. Minimization of F, exactly as in Section 4, starts with finding primes.

We first compute cofactors $F_z \sim u^3 + xu^1 + xu^2$, $F_{z'} = xu^1 + y'u^1 + xyu^2 + xyu^3$, $F_{zx} = u^1 + u^2 + u^3 \sim 1$, $F_{zx'} = u^3$, $F_{z'y} = xu^1 + xu^2 + xu^3 \sim x$, $F_{z'y'} = xu^1 + u^1 \sim u^1$. Then we use the divide–and–conquer formula to compute

$$\begin{aligned}
\pi(F_z) &= M[(x' + \pi(F_{zx}))(x + \pi(F_{zx'})] = M[x + u^3] = x + u^3;\\
\pi(F_{z'}) &= M[(y' + \pi(F_{z'y}))(y + \pi(F_{z'y'}))] = M[(y' + x)(y + u^1)] = y'u^1 + xy + xu^1;\\
\pi(F) &= M[(z' + \pi(F_z))(z + \pi(F_{z'}))] = M[(z' + x + u^3)(z + y'u^1 + xy + xu^1)] =\\
&= xy + xu^1 + xz + z'y'u^1 + zu^3.
\end{aligned}$$

We then discover, by applying the tautology–based algorithm in Section 4, that only xu^1 is redundant. (In this simple example, this can be done directly: If $xu^1 = 1$, for instance, then $x = u^1 = 1$, and the sum of the remaining cubes in $\pi(F)$ is $y + z + y'z' \sim 1$. Since $y + z + y'z'$ is identically 1, xu^1 is covered by $\pi(F)\backslash xu^1$.) We finally extract from the

minimized cover expressions for each of f_0, f_1, f_2: cubes that do not contain any u^j term will appear in all f_j, while a cube that contains u^j will appear in f_j with u^j dropped. Hence, $f_0 = xy + xz + y'z'$, $f_1 = xy + xz$, $f_2 = xy + xz + z$. □

Toyota was traditionally weak in parts sharing. Womack and Jones (1996, p. 238) report that in 1992 Toyota introduced a new division of labor in its product development system in order to "... focus on product families which share components rather than on stand–alone products". A similar division of labor had been introduced earlier by Chrysler, under the names of "platform teams" and "value engineering". *The Economist* (1995) reports that "The RAV4 ... was designed in a novel way. ... Toyota has copied value engineering techniques from Chrysler and Ford: these minimize the number of parts in a new model. Nearly half the parts in the RAV4 were already knocking around in other Toyota models. The aim now is to have each new Toyota model 70% built from parts common to its predecessor." This was achieved, as Taylor (1997 (b), p. 42) reports, by joint design of several models simultaneously: "Most auto companies develop models sequentially. First you design a Camry sedan; then you design a Camry coupe. That lightens the engineering load and ensures that problems on one model get resolved before the next one is started. But Toyota has begun developing similar models simultaneously, so that engineering tasks overlap. MIT's Cusumano believes that Toyota can save 15% in lead time and 50% in engineering hours by overlapping projects. Under this new system, Toyota's product fecundity has been unrivaled. In the past two years it has introduced 18 new or redesigned models. Several Japanese models went into production as little as 14.5 months after their designs were approved — probably an industry record. Overall, Toyota has doubled its engineering output over the past four years, while increasing its budget by only 20% — an astounding achievement."

In conclusion, design for variety is design for cost reduction applied to the set of all potential products. Variety may not be free, but its cost decreases as a free byproduct of waste elimination.

8 Summary and Conclusions

This paper has proposed circuit design as a model of waste elimination and its role in achieving simultaneous improvements in cost, quality, variety, and speed performance variables.

Waste elimination involves three steps: (a) definition of the product as a desired

behavior rather than as an historically given artifact; (b) mapping of desired behavior into physical components that realize it; and (c) elimination of redundant components. The activity that accomplishes these tasks is design. The key bottleneck in design is complexity. This has two effects. It makes design expensive; and/or it prevents waste elimination.

One way to manage complexity, outlined in Sections 3, 4, and 5 of this paper, is to look for better problem representations, divisions of labor, and product architectures. If such changes fit the product, they can yield exponential–size savings, revolutionizing an industry.

No currently known method of waste elimination works equally well on all design problems; all are worst–case exponential, although they differ in the type and frequency of worst cases. A necessary consequence of this is that all known methods have to be applied incrementally. After designers eliminate some, but not all waste, production takes place; at the same time, another design exercise begins, to eliminate some more waste. Given the complexity of design problems, each design effort can yield significant savings, even if no change in the underlying technology has taken place. Womack et al. (1990, ch. 6) report that Toyota and its suppliers, after joint analysis of costs, agree on an initial price for a part and on a schedule of continual future price decreases over the life of the part. Suppliers are expected to keep redesigning parts, eliminating some waste each time.

Improving design productivity involves discovery of better problem representations, divisions of labor, and product architectures. Each one of these constitutes an ill–structured problem, i.e. it cannot be usefully represented as an optimization problem with explicitly stated objective functions and constraints. While occasional discoveries are made, there is no systematic search procedure for improving design productivity. Drucker (1991) has associated this fact with the slow rate of increase of the productivity of knowledge work.

Given the complexity of waste elimination, and the lack of systematic procedures that improve design productivity, it is unlikely that any firm, or value stream, is producing near an optimum. Production functions, cost functions, learning curves and other such representations of past experience are unlikely to summarize all of the economically relevant aspects of technology. A substantial amount of research has suggested that firms "managed by the numbers" generated by such summaries of past experience forego large improvement opportunities. Hayes and Abernathy (1980, p. 74) made this point while discussing management by the numbers: "... its first doctrine is that neither industry

experience nor hands–on technological expertise counts for very much. ... it encourages the faithful to make decisions about technological matters simply as if they were adjuncts to finance or marketing decisions. Complex modern technology has its own inner logic and developmental imperatives." The belief that summaries of past experience describe a firm's possibility frontier implies that the only way to improve is to shift this frontier by investing in equipment, R&D, and training. Baldwin and Clark (1994, p. 73) summarize the consequences of internal control systems based on such beliefs: "... these systems obscured the value of investments in organizational capabilities, because such investments were hard to quantify — indeed, even to describe, within the financial models in use. As a result, companies often invested vigorously — but in the wrong things." Jensen (1993) has provided a well–known quantification of the costs of failure of internal control systems : "It is clear that GM's R&D and investment program produced massive losses. The company spent a total of $67.2 billion in excess of depreciation in the period [1980–1990] and produced a firm with total ending value of equity of $26.2 billion. ... the difference between the value of GM's actual strategy and the value of the equivalent–risk bank account strategy amounts to $-100.7 billion." (ibid., p. 858). In the light of these observations, this paper can be seen as an attempt to analyze some aspects of economic performance not visible through the standard apparatus of production functions, cost curves and learning curves.

European University Institute
Economics Department

EUI Working Paper **ECO** No. 97/35

Managing Design Complexity to Improve on Cost, Quality, Variety, and Time–to–Market Performance Variables

Spyros Vassilakis

Part C — pp. 55-74

Printed in Italy in December 1997
European University Institute
Badia Fiesolana
I – 50016 San Domenico (FI)
Italy

Appendix A

This Appendix contains proofs of Theorems 3.1 through 3.7, except for Theorems 3.2 and 3.4 proven in the main text. It also contains a proof of the statement in Example 3.3.

Theorem 3.1. F consists of its primes ($F = \pi(F)$) if, and only if, it is compact and maximal.

The proof is split into several lemmas.

Lemma A.1. F is compact if, and only if, it contains all primes of F(i.e., $\pi(F) \subseteq F$).

Proof. Let F be compact. To show $\pi(F) \subseteq F$, let $p \in \pi(F)$. Then, by definition of primality, $p \leq F$; and by compactness of F, there is a q in F such that $p \leq q$. Since $q \in F$, $q \leq F$, so $p \leq q \leq F$. By the definition of primality, $p = q$; hence $p \in F$.

For the converse, let F contain all its primes, i.e. let $\pi(F) \subseteq F$. To show that F is prime, let $p \leq F$; we need to show that there exists $q \in F$ such that $p \leq q$. If $p \in F$, there is nothing to show. If $p \notin F$, start dropping literals from p until a prime q of F is obtained. Then $p \leq q$, $q \in \pi(F) \subseteq F$.

Lemma A.2. Prime covers are maximal.

Proof. Let F be prime. To show maximality, we need to show that p, $q \in F$, $p \leq q$ imply $p = q$. If $p \neq q$, then $p < q \leq F$, a contradiction to the primality of p.

Lemma A.3. A compact, maximal cover F consists exclusively of the primes of F, i.e. $F = \pi(F)$.

Proof. By Lemma A.1, $\pi(F) \subseteq F$. To show that $F \subseteq \pi(F)$, let $p \in F \backslash \pi(F)$ (for contradiction). Since p is not a prime of F, there is a prime q such that $p < q \leq F$. Since $\pi(F) \subseteq F$, $q \in F$. The maximality of F then implies $p = q$, a contradiction.

Proof of Theorem 3.1. If F is compact and maximal, then $F = \pi(F)$ by Lemma A.3. If $F = \pi(F)$, then F is maximal by Lemma A.2 and compact by Lemma A.1.

Theorem 3.3. F is compact if, and only if, the consensus of any two cubes in F is covered by some cube in F.

The proof is split in several lemmas.

Lemma A.4. Let p, q be two cubes. Then $p \leq q$ if, and only if, $pq' = 0$.

Proof. Let $p = \Pi\, x_j^{A_j}$, $q = \Pi\, x_j^{B_j}$. Then $q' = \sum x_j^{B'_j}$, and $p\,q' = \sum (p\, x_j^{B'_j})$. For each j, $p\, x_j^{B'_j} = \left(\Pi_{i \neq j}\, x_i^{A_i}\right) \cdot x_j^{A_j \cap B'_j}$. If $p \leq q$, then $A_j \subseteq B_j$ for all j, so $A_j \cap B'_j = \phi$ for all j, i.e. $p\,q' = 0$. If $p\,q' = 0$, then either $p = 0 \leq q$; or $p \leq q = 1$; or $A_j \cap B'_j = \emptyset$ for all j, i.e. $A_j \subseteq B_j$ for all j, i.e. $p \leq q$.

Lemma A.5. Let p, q be two nonzero cubes. Then $p + q$ is compact if, and only if, $d(p, q) \neq 1$.

Proof. It is first shown that $d(p, q) = 0$ implies $p + q$ is compact. Suppose, for contradiction, that $p + q$ is not so. Then there is a cube $s \leq p + q$, $s \not\leq p$, $s \not\leq q$; $s \not\leq p$ means that p contains a literal ℓ not in s; $s \not\leq q$ means that q contains a literal m not in s. Hence, $p = \ell u$, $q = mv$, $\ell \notin u$, $m \notin v$, $\ell \notin s$, $m \notin s$. By Lemma A.4, $s \leq p + q$ implies $sp'q' = 0$, i.e. $s(\ell' + u')(m' + v') = 0$, i.e. $s\ell'm' = 0$. Since $\ell \notin s$, $m \notin s$, it must be that $\ell'm' = 0$, i.e. $l + m = 1$, i.e., for some variable x, $\ell = x$, $m = x'$. Hence, $p = xu$, $q = x'v$, i.e. $d(p, q) \geq 1$, a contradiction.

It is now shown that $d(p, q) \geq 2$ implies $p + q$ is compact. There exists a literal ℓ and cubes u, v such that $p = \ell u$, $q = \ell' v$, $uv = 0$. Let $s \leq p + q$, $s \neq 0$; it is to be shown that $s \leq p$ or $s \leq q$. By Lemma A.4, $sp'q' = 0$, i.e. $s(\ell' + u')(\ell + v') = 0$, i.e. $s\ell'v' = 0 = s\ell u'$. By Lemma A.4 again, $s\ell' \leq v$, $s\ell \leq u$. The last two inequalities imply that s must contain either ℓ or ℓ', for if it doesn't, $s\ell'v' = 0$ implies $sv' = 0$, and $s\ell u' = 0$ implies $su' = 0$; hence, $s \leq u$, $s \leq v$, i.e. $s \leq uv = 0$, a contradiction. Suppose that s contains ℓ, so that $s = s\ell$; then $s = s\ell \leq p$, Q.E.D. Suppose that s contains ℓ', so that $s = s\ell'$; then $s = s\ell' \leq q$, Q.E.D.

Finally, it is shown that $d(p, q) = 1$ implies $p + q$ is not compact. There exists a literal ℓ such that $p = \ell u$, $q = \ell' v$, $uv \neq 0$. It is to be shown that $uv \leq p + q$, but $uv \not\leq p$ and $uv \not\leq q$. First, $uv(p + q)' = uvp'q' = uv(\ell' + u')(\ell + v') = uv(\ell'v' + \ell u' + u'v') = 0$, hence, by Lemma A.4 $uv \leq p + q$. Secondly, $uvp' = uv(\ell' + v') = \ell' uv \neq 0$, since $uv \neq 0$ and $\ell \notin u$, $\ell \notin v$; hence $uv \not\leq p$. Finally, $uvq' = uv(\ell + v') = uv\ell \neq 0$, since $uv \neq 0$, $\ell' \notin u$, $\ell' \notin v$; hence $uv \not\leq q$.

Lemma A.6. If $p + q$ is not compact, then $p + q + c(p, q)$ is compact and equivalent to $p + q$.

Proof. Equivalence was shown in Theorem 3.2. Since $p + q$ is not compact, Lemma A.5 implies $d(p, q) = 1$, so $c(p, q)$ is well–defined. To show compactness of $p + q + c(p, q)$, it suffices to show that $s \leq p + q$, $s \not\leq p$, $s \not\leq q$ implies $s \leq c(p, q)$. Since $d(p, q) = 1$, there

exists a literal ℓ such that $p = \ell u$, $q = \ell' v$, $uv \neq 0$. By assumption and Lemma A.4, $sp'q' = 0$, i.e. $s\ell u' = s\ell' v' = su'v' = 0$. Neither ℓ nor ℓ' can belong to s. For if ℓ is in s, for instance, $s = s\ell$, $s\ell' = 0$ and therefore $sp' = s(\ell' + u') = su' = s\ell u' = 0$, i.e. by Lemma A.4, $s \leq p$, a contradiction. Similarly for ℓ'. Hence, $s\ell u' = 0$ implies $su' = 0$, because $\ell' \notin s$ and $\ell \notin u$; $su' = 0$, by A.4, implies $s \leq u$. Similarly, $s\ell' v' = 0$ implies $sv' = 0$, i.e. $s \leq v$. Hence $s \leq uv = c(p, q)$.

Proof of Theorem 3.3. Let F be compact, and p, q be cubes in F at distance one from each other. It is to be shown that $c(p, q)$ is covered by some cube in F. Note that $c(p, q) \leq p + q \leq F$, hence by F's compactness, there is a cube s in F such that $c(p, q) \leq s$.

For the converse, let F be a cover such that for any two p, q in F, $c(p, q)$ is covered by some cube in F. Suppose, for contradiction, F is not compact. Let n be the number of variables in F. Then any $t \leq F$ not covered by some cube in F must contain strictly less than n literals, i.e. it must not be a minterm. For if t is a minterm, $b(t)$ is a singleton, so $b(t) \subseteq b(F) = \cup_{p \in F} b(p)$ implies $b(t) \subseteq b(p)$ for some $p \in F$, i.e. $t \leq p$, a contradiction. Let $t \leq F$ contain the maximum number of literals among the implicants of F not covered by any single cube in F. This number is less than n, so there is a variable x appearing in F but not in t. Hence $xt < t \leq F$, $x't < t \leq F$. The maximality property of t implies that xt, $x't$ are each covered by single cubes in F, namely $xt \leq p$, $x't \leq q$, p, q in F. Then $t = xt + x't \leq p + q$. By assumption, $t \not\leq p$ and $t \not\leq q$, i.e. $p + q$ is not compact. By Lemma A.5, $d(p, q) = 1$, so $c(p, q)$ is defined. By our assumption on F, $c(p, q) \leq s$ for some s in F. By Lemma A.6, $p + q + c(p, q)$ is compact and equivalent to $p + q$. Then we have $t \leq p + q \sim p + q + c(p, q)$, $t \not\leq p$, $t \not\leq q$, and $p + q + c(p, q)$ is compact. It follows that $t \leq c(p, q) \leq s$, $s \in F$, a contradiction.

Theorem 3.5. Let $F^n = b(G)$, and for each $t = n, n-1, \ldots, 1$, $F^{t-1} = A(F^t)$, $S^t = S(F^t)$, $\pi^t = F^t \backslash S^t$. Then the set of primes of G is $\pi(G) = \cup_{t=1}^{n} \pi^t$. □

The proof is split into several lemmas.

Lemma A.7 (Wegener, 1987, p. 25). Let G be a cover and p a cube. Then $p \leq F$ if, and only if, for any variable x not in p, $xp \leq F$ and $x'p \leq F$.

Proof. Let $p \leq F$, x a variable not in p. If $xp \not\leq F$, then there exists a vector $w = (w_x, w_{-x})$ with $w_x = 1$ such that $p(w_{-x}) = 1$, $F(w_{-x}) = 0$; this contradicts $p \leq F$. Hence $xp \leq F$; similarly, $x'p \leq F$. For the converse, if $xp \leq F$, $x'p \leq F$, then $p = xp + x'p \leq F + F = F$.

Lemma A.8. For each $t = n, \ldots, 1$, F^t consists of all implicants of G that contain exactly t literals.

Proof. This is true by construction for $F^n = b(G)$. Suppose it holds for $t > 1$; then $F^{t-1} = A(F^t) = \{p$: there exists a variable x such that both xp and $x'p$ belong to $F^t\}$. By the induction hypothesis, xp and $x'p$ are implicants of G containing exactly t literals. Hence p contains exactly $t-1$ literals; and is an implicant of G by Lemma A.7. For the converse, let p be an implicant of G that contains exactly $t-1$ literals. By Lemma A.7, both xp and $x'p$ are implicants of G, for any x not in p. By the induction hypothesis, both xp and $x'p$ are in F^t, hence p is in F^{t-1}.

Lemma A.9. For each $t = n, \ldots, 1$, π^t consists of all primes of G containing exactly t literals.

Proof. $\pi^t \leq F^t$, so all members of π^t are implicants of G, by Lemma A.8. Suppose $p \in \pi^t$ is not a prime of G. Then there exists a literal ℓ and a cube q such that $p = \ell q$ and q is still an implicant of G.

By Lemma A.7, both ℓq and $\ell' q$ are implicants of G; and by Lemma A.8, both belong to F^t. Hence $p = \ell q$ is not in π^t, a contradiction. Hence every member of π^t is a prime of G containing exactly t literals.

For the converse, let p be a prime of G containing exactly t literals. By Lemma A.8, $p \in F^t$. Suppose, for contradiction, that $p \in S^t$, i.e. that for some literal ℓ and cube q, $p = \ell q$, and $\ell' q \in F^t$. By Lemma A.8, ℓq and $\ell' q$ are implicants of G, while by A.7 q is an implicant of G, a contradiction to p's primality. Hence $p \in F^t$, $p \notin S^t$, i.e. $p \in F^t \backslash S^t = \pi^t$.

Proof of Theorem 3.5. $\pi(G) = \cup_{t=1}^{n} \pi^t$ by Lemma A.9.

Example 3.3. Let B_n be the cover consisting of all minterms m in n variables whose number of positive literals $\lambda(m)$ is not divisible by 3. Let π_n be the number of primes of B_n. Then, along the subsequence $n = 6k + 2$, $\lim_{n\to\infty} 2^{-n}\, \pi_n = \infty$.

Proof. Let $\sigma = 12, 45, 78, \ldots$ be the sequence of numbers not divisible by 3, arranged in pairs. Let $B_{n,i} = \{m \in B_n,\, \lambda(m) = i\}$. Then $B_n = \cup\{B_{ni} \cup B_{n,i+1} : (i, i+1) \in \sigma,\, 1 \leq i \leq n-1\}$. Let $C_{ni} = \{c(p,q);\, p \in B_{ni},\, q \in B_{n,i+1},\, (i, i+1) \in \sigma\}$ be the set of consensus cubes formed by minterms in B_{ni}, $B_{n,i+1}$. Then the set of primes of B_n is $\pi(B_n) = \cup\{C_{n,i} : (i, i+1) \in \sigma,\, 1 \leq i \leq n-1\}$, because if $j \neq i+1$ and j is not divisible by 3, $|i - j| \geq 3$, hence no consensus forms can be built out of cubes in $C_{n,i}$, $C_{n,j}$: the QM method will stop after one iteration and deliver $\pi(B_n)$. Each C_{ni} contains $\binom{n}{i}(n-i)$

cubes, because B_{ni} contains $\binom{n}{i}$ cubes; and because each cube m in B_{ni} is at distance one from exactly $n-i$ cubes in $B_{n,i+1}$, namely those that contain an unprimed variable if m does. Hence the number of primes in B_n is $\pi_n = \sum_{(i,i+1)\in\sigma} \binom{n}{i}(n-i)$. Setting $n = 6k+2$, we obtain $\pi_n = \sum_{t=0}^{2k} \binom{6k+2}{3t+1}(6k-3t+1) > \binom{6k+2}{3k+1}(3k+1)$. By this inequality and Stirling's formula $2^{-n}\,\pi_n > 2^{-6k-2}\,(3k+1)\binom{6k+2}{3k+1} \sim 2^{-6k-2}\,(3k+1)\,2^{6k+2}\,\pi^{-\frac{1}{2}}\,(3k+1)^{-\frac{1}{2}} = (3k+1)^{\frac{1}{2}}\,\pi^{-\frac{1}{2}} = n^{\frac{1}{2}}\,(2\pi)^{-\frac{1}{2}} \to \infty$ as $n \to \infty$.

Theorem 3.6. Let x be any variable in cover F. Then $\pi(F) = M\,[(x' + \pi(F_x))\ (x + \pi(F_{x'}))]$.

The proof is split into several lemmas.

Lemma A.10. $b(FG) = b(F) \cap b(G)$.

Proof. Recall that $FG = \sum_{p\in F} \sum_{q\in G}\ pq$. Then

$$b(F) \cap b(G) = (\cup_{p\in F}\, b(p)) \cap (\cup_{q\in G}\, b(q)) = \cup_{p\in F} \cup_{q\in G}\ b(p) \cap b(q) =$$
$$= \cup_{p\in F} \cup_{q\in G}\ b(pq) = b\left(\sum_{p\in F} \sum_{q\in G}\ pq\right) = b(FG).$$

Lemma A.11. $F \sim (x' + F_x)(x + F_{x'})$.

Proof. Both sides equal F_x when $x = 1$; both sides equal $F_{x'}$ when $x = 0$.

Lemma A.12. The product of compact covers is also compact.

Proof. Let F, G be compact covers, $p \le FG$. Then $p \le F$, $p \le G$, since by Lemma A.10 $FG \le F$, $FG \le G$. By compactness, then, there exist $u \in F$, $v \in G$ such that $p \le u$, $p \le v$. Hence $p \le uv \in FG$, Q.E.D.

Lemma A.13. Let $\hat{F}$, $\hat{G}$ be compact covers equivalent to F, G, respectively. Then $\hat{F}\hat{G}$ is a compact cover equivalent to FG.

Proof. $\hat{F}\hat{G}$ is compact by A.12. By A.10, $b(\hat{F}\hat{G}) = b(\hat{F}) \cap b(\hat{G}) = b(F) \cap b(G) = b(FG)$, i.e. $\hat{F}\hat{G} \sim FG$.

Lemma A.14. If G is compact, then so is $M(G)$.

Proof. Let $p \le M(G)$. Since $G \sim M(G)$, $p \le G$. G's compactness implies $p \le q$, $q \in G$. If $q \notin M(G)$, then there exists $s \in M(G)$, $q \le s$; hence $p \le s$. Hence in all cases $p \le s$, $s \in M(G)$, and thus $M(G)$ is compact.

Lemma A.15. Let $\pi(F)$ be the set of primes of F. Then $\pi(F) = M\,[\pi(x' + F_x)\ \pi(x + F_{x'})]$.

Proof. By A.11, $F \sim (x' + F_x)(x + F_{x'})$. By Theorem 3.1, $\pi(x' + F_x)$, $\pi(x + F_{x'})$ are

compact covers equivalent to $x' + F_x$, $x + F_{x'}$, respectively. By A.13 then, $\pi(x' + F_x) \cdot \pi(x + F_{x'})$ is a compact cover equivalent to F. By A.14, $M[\pi(x' + F_x)\,\pi(x + F_{x'})]$ is a compact, maximal cover equivalent to F; by Theorem 3.1, it equals $\pi(F)$.

Lemma A.16. If G does not depend on x, then no prime of G depends on x.

Proof. Let $p \leq G$ be a prime of G. If p depends on x, then $p = xq$ or $p = x'q$. Suppose, for instance, that $p = xq$, and let $p(w) = 1$; then $w_x = 1$, $q(w_{-x}) = 1$. Since $p \leq G$, we have $G(w) = 1$, and since G does not depend on x, $G(w_{-x}) = 1$. Hence, we obtain: $q(w_{-x}) = 1 \Rightarrow p(1, w-x) = 1 \Rightarrow G(w_{-x}) = 1$, i.e. $q \leq G$, a contradiction to p's primality. Hence p does not depend on x.

Lemma A.17. Let ℓ be a literal, and G a cover independent of the variable in ℓ. Then $\pi(G) \subseteq \pi(\ell + G)$.

Proof. Let $p \in \pi(G)$. By A.16, p is independent of the variable in ℓ, say x. Suppose, for contradiction, that $p \notin \pi(\ell+G)$. Then there exists a literal m, $m \neq \ell$, $m \neq \ell'$, and a cube q independent of x, such that $p = mq$, $q \leq \ell + G$. Suppose without loss of generality, that $\ell = x$. Since q is independent of x, $q(w) = 1$ with $w_x = 1$ implies $q(w_{-x}, 0) = 1$. Hence $q(w) = 1 \Rightarrow q(w_{-x}, 0) = 1 \Rightarrow (\ell + G)(w_{-x}, 0) = 1 \Rightarrow G(w_{-x}) = 1 \Rightarrow G(w) = 1$, i.e. $q \leq G$, contradicting the primality of p. Hence $p \in \pi(\ell + G)$, Q.E.D.

Lemma A.18. Let ℓ be a literal and G be a cover independent of the variable in ℓ. Then any prime of $\ell + G$ is either ℓ or independent of the variable in ℓ.

Proof. Suppose, without loss of generality, that $\ell = x'$. Let $p \in \pi(\ell+G)$. If p contains x', i.e. $p = x'q$, then $p = x'$; for if not, $p = x'q < x' \leq x' + G$, i.e. p is not a prime of $\ell + G$, a contradiction. Now suppose, for contradiction, that $p \neq \ell$ but p depends on x; then $p = xq$ for some q independent of x. Since q is independent of x, $q(w) = 1 \Rightarrow q(w_{-x}, 1) = 1 \Rightarrow p(w_{-x}, 1) = 1 \Rightarrow (x' + G)(w_{-x}, 1) = 1 \Rightarrow G(w_{-x}) = 1$, i.e. $q \leq G \leq x' + G$, contradicting p's primality. Hence, any prime $p \neq \ell$ has to be independent of x.

Lemma A.19. Let ℓ be a literal and G a cover independent of the variable in ℓ. Then $\pi(\ell + G) \subseteq \ell + \pi(G)$.

Proof. Let $p \in \pi(\ell + G)$. By A.18, either $p = \ell$ or p is independent of the variable in ℓ. If $p = \ell$, there is nothing to prove. If p is independent of the variable in x, then (for $\ell = x$), $p(w) = 1 \Rightarrow p(w_{-x}, 0) = 1 \Rightarrow (\ell + G)(w_{-x}, 0) = 1 \Rightarrow G(w_{-x}) = 1 \Rightarrow G(w) = 1$, i.e. $p \leq G$, i.e. p is an implicant of G. Suppose, for contradiction, that p is not a prime of G; then $p = yq$, $q \leq G \leq \ell + G$, a contradiction to p's primality. Hence $p \in \pi(G)$, Q.E.D. The same proof works when $\ell = x'$, by setting $w_x = 1$.

Lemma A.20. For any cover G, and any literal ℓ whose variable is not in G, $\pi(\ell + G) = \ell + \pi(G)$.

Proof. We show $\ell + \pi(G) \subseteq \pi(\ell + G)$, since the other half is Lemma A.19. By A.17, we need only show $\ell \in \pi(\ell + G)$. If $\ell + G$ is not identically one, then $\ell \leq \ell + G$ and $1 \not\leq \ell + G$, hence $\ell \in \pi(\ell + G)$, Q.E.D.

If $\ell + G \sim 1$, then $G \sim 1$ (to see this, let $\ell = x$; if $G \not\sim 1$, then $G(w) = 0$ for some w; since G is independent of x, w_x can be set to zero. But then $(\ell + G)(w) = 0$. Since $\pi(G) \sim G$, both sides of the equality to be proven equal 1.

Proof of Theorem 3.6. By Lemmas A.15 and A.20, and the fact that F_x, $F_{x'}$ are independent of x.

Theorem 3.7. If F is a monotone cover, then $\pi(F) = M(F)$.

Proof. $M(F)$ is maximal and equivalent to F. To show compactness, note that for any two cubes p, q in F, $d(p, q) = 0$, since each variable appears with the same sign in all cubes. By Theorem 3.3, F is compact; by A.14, so is $M(F)$; by Theorem 3.1, $M(F) = \pi(F)$.

Appendix B

This Appendix provides a proof of Theorems 3.8, 3.9, and 3.10.

Theorem 3.8. For any design problem, $\pi(G)$ and $\pi(I_G) = M(I_G)$ are isomorphic. Each prime of I_G gives rise to a prime of G by replacing every instance of z'_{kj} by x_j^{1-k}. □

The proof splits into several lemmas.

Lemma B.1. Let F, G be complementary covers, p a cube. Then p is an implicant of F if, and only if, for each q in G, $d(p,q) \geq 1$.

Proof. $p \leq F \leftrightarrow b(p) \subseteq b(F) \leftrightarrow b(p) \cap b(G) = \phi \leftrightarrow b(p) \cap \cup_{q \in G} b(q) = \phi \leftrightarrow \forall q \in G$, $b(p) \cap b(q) = \phi \leftrightarrow \forall q \in G$, $d(p,q) \geq 1$.

Lemma B.2. Let F, G be complementary covers. An implicant $p = \Pi_{j=1}^n x_j^{A_j}$ of F is prime if, and only if, it satisfies the following condition C:

(C) If A_j is singleton, then there is a $q = \Pi_{i=1}^n x_i^{B_i}$ in G such that $A_j \cap B_j = \phi$, $A_i \cap B_i \neq \phi$ for all $i \neq j$.

Proof. Let $p \leq F$ be a prime of F. Let A_j be singleton; without loss of generality, set $A_j = \{1\}$. Suppose, for contradiction, that for every $q = \Pi_{j=1}^n x_j^{B_j^q}$ in G, either $A_j \cap B_j^q \neq \phi$ or $\exists i \neq j$ such that $A_i \cap B_i^q = \phi$; equivalently, either $1 \in B_j^q$ or $\exists i \neq j$ such that $A_i \cap B_i^q = \phi$. Let $G_1 = \{q \in G : 1 \in B_j^q\}$, $G_2 = \{q \in G : B_j^q = \{0\}\}$ be a partition of G. If $q \in G_1$, then $A_j \cap B_j^q \neq \phi$; since, by B.1, $d(p,q) \geq 1$, there is $i \neq j$ such that $A_i \cap B_i^q = \phi$. If $q \in G_2$, then by the contradictions hypothesis, $\exists i \neq j$, $A_i \cap B_i^q = \phi$. Hence, for each q in G, $\exists i \neq j$, $A_i \cap B_i^q = \phi$. It follows that the cube $s = \Pi\, x_j^{C_j}$ defined by $C_j = D$, $C_i = A_i$, $i \neq j$, satisfies $d(s,q) \geq 1$ for each q in G. By B.1, $s \leq F$; by the definition of s, $p < s$. Hence $p < s \leq F$, a contradiction to p's primality.

For the converse, let $p = \Pi_{j=1}^n x_j^{A_j}$ be an implicant of F that satisfies condition C. Suppose, for contradiction, that p is not a prime of F. Then there exists a cube $s = \Pi_{i=1}^n x_i^{C_i}$ such that $p < s \leq F$. Hence $A_i \subseteq C_i$ for all i; and there exists a j such that A_j is singleton and $C_j = D$. By property C of p, there is a $q = \Pi_{i=1}^n x_i^{B_i}$ in G such that $A_j \cap B_j = \phi$, $A_i \cap B_i \neq \phi$ $\forall i \neq j$. Hence $B_j \cap C_j \neq \phi$, $B_i \cap C_i \supseteq B_i \cap A_i \neq \phi$, i.e. $d(s,q) = 0$. This contradicts, by B.1, the fact that $s \leq F$. □

The next two definitions establish some useful notation.

Definition B.1. Let $E = \{\{0\}, \{1\}, D\}$ be the set of nonempty subsets of D, ordered

by set inclusion; E^n is ordered componentwise, and is isomorphic to the set of cubes $p = \Pi_{j=1}^n x_j^{A_j}$ in n variables, since each p can be identified with $(A_1 \ldots A_n) \in E^n$. Let $D_0 = \{10, 01, 11\}$ be ordered as follows: $10 < 11, 01 < 11$; D_0^n is ordered complementwise.

Definition B.2. The function $\hat{\alpha}_j : E \to D_0$ maps each nonempty subset of D into its positional notation, namely $\{0\} \to 10$, $\{1\} \to 01$, $D \to 11$; the function $\hat{\beta}_j : D_0 \to E$, given by $10 \to \{0\}$, $01 \to \{1\}$, $11 \to D$, is its inverse. Given a cube $p = (A_1, \ldots, A_n) \in E^n$, $\alpha_j(p) = \hat{\alpha}_j(A_j)$; given a vector $d = (d^1, \ldots, d^n) \in D_0^n$, $\beta_j(d) = \hat{\beta}_j(d^j)$. The functions $\alpha : E^n \to D_0^n$, $\beta : D_0^n \to E^n$, are defined by $\alpha(p) = (\alpha_1(p), \ldots, \alpha_n(p))$, $\beta(d) = (\beta_1(d), \ldots, \beta_n(d))$.

Lemma B.3. The pairs $(\hat{\alpha}_j, \hat{\beta}_j)$, (α, β) are strictly increasing, inverse functions.

Proof. Obvious from Definition B.2.

Lemma B.4. Let $p = (A_1, \ldots, A_n)$, $q = (B_1, \ldots, B_n)$ be two cubes. Then $A_j \cap B_j = \phi$ if, and only if, $\alpha_j(p)\,\alpha_j(q) = \sum_{k=0}^{1} \alpha_{kj}(p)\,\alpha_{kj}(q)$ equals zero.

Proof. Note that $A_j \cap B_j = \phi$ if, and only if, $A_j = \{0\}$ and $B_j = \{1\}$, or viceversa; $\alpha_j(p) = \hat{\alpha}_j(A_j) = \hat{\alpha}_j(\{0\}) = 10$; $\alpha_j(q) = \hat{\alpha}_j(B_j) = \hat{\alpha}_j(\{1\}) = 01$; and $\alpha_j(p)\,\alpha_j(q) = 1 \cdot 0 + 0 \cdot 1 = 0$. The converse follows from the fact that $\alpha_j(p)\,\alpha_j(q) = 0$ iff $\alpha_j(p) = 01$ and $\alpha_j(q) = 10$, or $\alpha_j(p) = 10$ and $\alpha_j(q) = 01$.

Lemma B.5. Let F, G be complementary covers. If p is an implicant of F, there exists a vector z that satisfies condition S:

(S) $\forall q \in G\ \exists j = j_q$ such that $\alpha_j(q)\, z^j = 0$, $z \in D_0^n$.

Conversely, if z satisfies S, then $\beta(z)$ is an implicant of F.

Proof. Let $p = \Pi_{j=1}^n x_j^{A_j}$ be an implicant of F. By Lemma B.1, $\forall q \in G\ d(p, q) \geq 1$; i.e. $\forall q \in G\ \exists j = j_q$ such that $A_j \cap B_j^q = \phi$; i.e. by Lemma B.4, $\forall q \in G\ \exists j = j_q$ such that $\alpha_j(q)\,\alpha_j(p) = 0$. Hence, set $z^j = \alpha_j(p)$, $z = (z^1, \ldots, z^n) = \alpha(p)$. Conversely, let z satisfy S. Let $p = \beta(z) = \Pi_{j=1}^n x_j^{A_j}$. By Lemma B.3, $\alpha(p) = \alpha(\beta(z)) = z$, i.e. $z^j = \alpha_j(p)$. By this and condition S, $\forall q \in G\ \exists j = j_q$ such that $\alpha_j(q) \cdot \alpha_j(p) = 0$. Lemma B.4 then implies that $\forall q \in G\ \exists j = j_q$ such that $A_j \cap B_j^q = \phi$, hence $d(p, q) \geq 1$. Lemma B.1 then shows $p \leq F$, Q.E.D.

Lemma B.6. Let F, G be complementary covers. If p is a prime of F, then $\alpha(p)$ is a maximal solution of S (if $\alpha(p) \leq w$ and if w satisfies S, then $w = \alpha(p)$). Conversely, if z is a maximal solution of S, then $\beta(z)$ is a prime of F.

Proof. Let p be a prime of F. By Lemma B.5, $z = \alpha(p)$ satisfies S. Let $z \leq w$, with w also satisfying S. By Lemma B.5, $\beta(w) \leq F$. Hence, by Lemma B.3, $p = \beta(\alpha(p)) = \beta(z) \leq \beta(w) \leq F$; by primality of p, $p = \beta(w)$, i.e. $\beta(\alpha(p)) = \beta(w)$; by B.3 again, $\alpha(p) = w$. Q.E.D.

For the converse, let z be a maximal solution of S, and $p = \beta(z)$. By Lemma B.5, $p \leq F$. Suppose $p \leq s \leq F$. Then, by Lemma B.5, both $\alpha(p)$, $\alpha(s)$ satisfy S, and by Lemma B.3 $\alpha(p) \leq \alpha(s)$. We obtain, then, $z \stackrel{B.3}{=} \alpha(\beta(z)) = \alpha(p) \stackrel{B.3}{\leq} \alpha(s)$; since both z and $\alpha(s)$ satisfy S, and z is maximal, $z = \alpha(s)$. Hence $\alpha(p) = \alpha(\beta(z)) = z = \alpha(s)$, i.e. by B.3, $p = s$. Hence p is a prime of F. □

Note that (α, β) is an isomorphism pair between primes of F and maximal solutions of S. The next definition will introduce transformations that will turn out to be an isomorphism pair between maximal solutions of S and primes of I_G. Note that $(E^2)^n$ is (isomorphic to) the set of cubes in variables (z_{0j}, z_{1j}), $j = 1, \ldots, n$, since each such cube can be identified with a point $(A_{0j}, A_{1j})_{j=1}^n \in (E^2)^n$.

Definition B.3. The function $\gamma : (E^2)^n \to D_0^n$ maps cubes formed out of variables $(z_{0j}, z_{1j}), j = 1, \ldots, n$, into vectors in D_0^n, hence candidate solutions of S. If $e = e^1 \ldots e^n$, $e^j \in E^2$, then $\gamma(e) = (\hat{\gamma}_1(e^1) \ldots \hat{\gamma}_n(e_n))$, where $\hat{\gamma}_j : E^2 \to D_0$ is given by $\hat{\gamma}_j(e^j) = (\zeta(e_0^j), \zeta(e_1^j))$; and $\zeta : E \to D$ is given by $\zeta(\{0\}) = 0$, $\zeta(\{1\}) = 1 = \zeta(D)$.

Definition B.4. The function $\delta : D_0^n \to (E^2)^n$ maps candidate solutions of S into cubes formed out of variables (z_{0j}, z_{1j}), $j = 1, \ldots, n$. If $d = (d^1, \ldots, d^n)$, then $\delta(d) = (\hat{\delta}_1(d^1), \ldots, \hat{\delta}_n(d^n))$, where $\hat{\delta}_j : D_0 \to E^2$ is given by $\hat{\delta}_j(d^j) = (\theta(d_0^j), \theta(d_1^j))$; and $\theta : D \to E$ is given by $\theta(0) = \{0\}$, $\theta(1) = D$.

Definition B.5. Let X, Y be partially ordered sets, and $f : X \to Y$, $g : Y \to X$ increasing functions. The pair (f, g) is a projection–embedding pair if

(a) $x \leq g(f(x))$, all x in X,

(b) $y = f(g(y))$, all y in Y.

Lemma B.7. The pairs (ζ, θ), $(\hat{\gamma}_j, \hat{\delta}_j)$, (γ, δ) are all projection–embedding pairs.

Proof. Consider first the pairs $\zeta : E \to D$, $\theta : D \to E$. ζ is increasing; and θ is increasing and one–to–one. To show (a), we need to show $e \leq \theta(\zeta(e))$ for each $e \in E$. If $e = \{0\}$, then $\theta(\zeta(e)) = \theta(0) = \{0\} = e$; if $e = \{1\}$, then $\theta(\zeta(e)) = \theta(1) = D \supseteq \{1\} = e$; and if $e = D$, then $\theta(\zeta(e)) = \theta(1) = D = e$.

To show (b), we need to show $d = \zeta(\theta(d))$ for all $d \in D$. If $d = 0$, then $\zeta(\theta(d)) = \zeta(\{0\}) = 0 = d$; if $d = 1$, then $\zeta(\theta(d)) = \zeta(D) = 1 = d$.

Pairs $(\hat{\gamma}_j, \hat{\delta}_j)$, (γ, δ) simply inherit properties (a) and (b) from (ζ, θ).

For example, $\hat{\delta}_j(\hat{\gamma}_j(e^j)) = \hat{\delta}_j\left(\zeta(e_0^j), \zeta(e_1^j)\right) = \left(\theta(\zeta(e_0^j)), \theta(\zeta(e_1^j))\right) \geq (e_0^j, e_1^j) = e^j$. □

Lemma B.8. Let I_G be the cover defined in Def. 3.13. Then its behavior is $b(I_G) = \{d \in D_0^n :$ there exists a z that solves S and $d \leq z\} =$ all vectors dominated by some solution of S.

Proof. Recall that $I = \Pi_{q \in G'} \sum_{j=1}^{n} H_j(q)$, and that $H_j(q) = (\alpha'_{0j}(q) + z'_{0j})(\alpha'_{1j}(q) + z'_{1j})$. Performing the multiplications involved in I's definition, we obtain

$$I = \sum_{j_1=1}^{n} \ldots \sum_{j_q=1}^{n} \ldots \sum_{j_N=1}^{n} \Pi_{q \in G'} H_{j_q}(q), \qquad N = |G'|. \tag{1}$$

By the definition of $H_j(q)$, $H_j(q) = z'_{0j}$ if $\alpha_j(q) = 10$; $H_j(q) = z'_{1j}$ if $\alpha_j(q) = 01$; and $H_j(q) = z'_{0j}\, z'_{1j}$ if $\alpha_j(q) = 11$. We express these equalities compactly

$$H_j(q) = z_{0j}^{A_{0j}^q}\, z_{1j}^{A_{1j}^q} \tag{2}$$

$$A_{kj} = \theta(\alpha'_{kj}(q)), \qquad k = 0, 1 \tag{3}$$

By (1), we observe that to remove all cubes with $z'_{0j}\, z'_{1j}$ terms from I, we need to delete all cubes $\Pi_{q \in G} H_{j_q}(q)$ such that $\alpha_{j_q}(q) = 11$ for some $q \in G'$, because such a q gives rise, by (2) and (3), to a $z'_{0j}\, z'_{1j}$ term.

Let $J = \{(j_1, \ldots, j_N) : \forall q \in G',\ \alpha_{j_q}(q) \neq 11\}$. Then

$$I_G = \sum_{(j_1, \ldots, j_N) \in J} \Pi_{q \in G'} H_{j_q}(q). \tag{4}$$

It is now shown that each $d \in b(I_G)$ is dominated by a solution z of S. If $d \in b(I_G)$, there exists a set of indices $(j_1 \ldots j_N) \in J$ such that, for each $q \in G'$ and $j = j_q$, $H_j(q)(d^j) = 1$. Hence by (2) and the definition of J,

$$\forall q \in G',\ \exists j = j_q \text{ such that } d_k^j \in A_{kj}^q, \qquad k = 0, 1 \tag{5}$$

$$j = j_q,\ q \in G' \text{ imply } \alpha_j(q) \neq 11. \tag{6}$$

Given this information, we can define a z that solves S. First, if j is a variable such that $j \neq j_q$ for all $q \in G'$, set $z^j = 11$. Secondly, if j is a variable such that $j = j_q$

for some $q \in G'$, there are, by (6), two possible cases: $\alpha_j(q) = 01$, or $\alpha_j(q) = 10$. If $\alpha_j(q) = 01$, then set $z^j = 10$; and if $\alpha_j(q) = 10$, then set $z^j = 01$. Obviously, then, if $j = j_q$ then $\alpha_j(q)z^j = 0$, and $z \in D_0^n$. Hence we obtain

$$\forall q \in G' \ \exists j = j_q, \quad \text{such that} \quad \alpha_j(q)\, z^j = 0, z \in D_0^n, \tag{7}$$

i.e. z solves S. We now show that $d \leq z$. In fact, if $j = j_q$ and $\alpha_j(q) = 10$, then by (5), (3), $d_0^j \in A_{0j}^q = \theta(\alpha'_{0j}(q)) = \theta(1') = \theta(0) = \{0\}$, i.e. $d_0^j = 0$. Hence $d^j \leq z^j = 01$. If, on the other hand, $j = j_q$ and $\alpha_j(q) = 01$, then by (5), (3), $d_1^j \in A_{1j}^q = \theta(\alpha'_{1j}(q)) = \theta(1') = \theta(0) = \{0\}$, i.e. $d_1^j = 0$, i.e. $d^j \leq z^j = 10$. Finally, if $j \neq j_q$ for all $q \in G'$, then $d^j \leq 11 = z^j$. Hence $d \leq z$.

For the converse, let z satisfy S (i.e. (7)) and $d \leq z$. We show that $d \in b(I_G)$. If $j = j_q$ and $\alpha_j(q) = 10$, then (7) implies $z^j = 01$, and $d^j \leq z^j$ implies $d_0^j = 0$; hence $d_{kj} \in A_{kj}^q$, $k = 0, 1$, i.e. $H_j(q)(d^i) = 1$. If $j = j_q$ and $\alpha_j(q) = 01$, then (7) implies $z^j = 10$, and $d^j \leq z^j$ implies $d_1^j = 0$; hence $d_k^j \in A_{kj}^q$, i.e. $H_j(q)(d^j) = 1$. Finally, if $j \neq j_q$ for all $q \in G'$, $d^j \leq z^j$ does not imply anything definite about d^j. By (7) and these results, then, $\forall q \in G' \ \exists j = j_q$ such that $H_j(q)(d^j) = 1$; hence $\Pi_{q \in G'} H_{j_q}(q)\,(d) = 1$, i.e. $d \in b(I_G)$.

Lemma B.9. For each $z^j \in D_0$, $b(\hat{\delta}_j(z^j)) = \{d^j : d^j \leq z^j\}$; for each $z \in D_0^n$, $b(\delta(z)) = \{d : d \leq z\}$.

Proof. If $z^j = 10$, then $\hat{\delta}_j(z^j) = (D, 0)$, $b(\hat{\delta}^j(z^j)) = D \times \{0\} = \{d^j : d^j \leq z^j\}$. Similarly for $z^j = 01, 11$. Finally, $b(\delta(z)) = b(\hat{\delta}_1(z^1)) \times \ldots \times b(\hat{\delta}_n(z^n)) = \{d : d \leq z\}$.

Lemma B.10. If z satisfies S, then $\delta(z) \leq I_G$.

Proof. Let z satisfy S. Then $b(\delta(z)) = \{d : d \leq z\} \subseteq b(I_G)$, by Lemma B.8.

Lemma B.11. If p is a prime of a cover F that is decreasing in x, then p does not contain x.

Proof. Suppose, for contradiction, that $p = xq$. Then $p(w) = 1$ implies $F(w) = 1$; and $w_x = 1$, $q(w_{-x}) = 1$. Since F is decreasing in x, $F(0, w_{-x}) \geq F(1, w_{-x}) = F(w) = 1$. Hence $q(w_{-x}) = 1 \Rightarrow p(1, w_{-x}) = 1 \Rightarrow$. $F(0, w_{-x}) = 1 = F(1, w_{-x})$, i.e. $q \leq F$, a contradiction to the primality of p.

Lemma B.12. If e is a prime of I_G, then $e = \delta(\gamma(e))$; and $\gamma(e)$ is a maximal solution of S.

Proof. By Lemma B.11 and the fact that I_G is decreasing in all variables,

$$e_k^j \neq \{1\}, \qquad k = 0, 1, \qquad j = 1, \ldots, n. \tag{1}$$

By the definition of $\hat{\gamma}_j$, we obtain $\hat{\gamma}_j(\{0\}) = 0$, $\hat{\gamma}_j(D) = 1$; hence

$$\hat{\gamma}_j(e_k^j) = \max e_k^j. \tag{2}$$

By the definition of behavior, $b(e) = \{d : d_k^j \in e_k^j\} \overset{(1)}{=} \{d : d_k^j \leq \max e_k^j\} \overset{(2)}{=} \{d : d_k^j \leq \hat{\gamma}_j(e_k^j)\} = \{d : d \leq \gamma(e)\} \overset{B.9}{=} b(\delta(\gamma(e)))$, i.e.

$$b(e) = b(\delta(\gamma(e))) = \{d : d \leq \gamma(e)\}. \tag{3}$$

Since $e \leq I_G$ and by (3) $\gamma(e) \in b(e)$, $\gamma(e) \in b(I_G)$.

By Lemma B.8, then, there is a solution z of S such that $\gamma(e) \leq z$. Hence $e \overset{B.7}{\leq} \delta(\gamma(e)) \overset{B.7}{\leq} \delta(z) \overset{B.10}{\leq} I_G$. The primality of e then implies

$$e = \delta(\gamma(e)) = \delta(z), \tag{4}$$

while (4) and the fact that δ is one–to–one imply $z = \gamma(e)$, i.e. that $\gamma(e)$ solves S. To show that $\gamma(e)$ is a maximal solution of S, let $w \geq \gamma(e)$ be a solution of S. We show that $w = \gamma(e)$. We have $e \overset{(4)}{=} \delta(\gamma(e)) \overset{B.7}{\leq} \delta(w) \overset{B.10}{\leq} I_G$; the primality of e then implies $e = \delta(\gamma(e)) = \delta(w)$; and the fact that δ is one–to–one implies $w = \gamma(e)$. □

Lemma B.13. If z is a maximal solution of S, then $\delta(z)$ is a prime of I_G.

Proof. By B.10, $\delta(z) \leq I_G$. If $\delta(z)$ is not a prime of I_G, there exists a prime e such that $\delta(z) < e \leq I_G$. Hence $z \overset{B.7}{=} \gamma(\delta(z)) \overset{B.7}{\leq} \gamma(e)$; and $\gamma(e)$ is a maximal solution of S, by B.12. By maximality of z, $z = \gamma(e)$, hence $\delta(z) = \delta(\gamma(e)) \overset{B.7}{\geq} e > \delta(z)$, a contradiction.

Lemma B.14. $\pi(G)$ and $\pi(I_G)$ are isomorphic.

Proof. Let λ be defined by $\lambda = \delta \circ \alpha$. By Lemma B.6, if $p \in \pi(G)$, then $\alpha(p)$ is a maximal solution of S; and by Lemma B.13, $\lambda(p) = \delta(\alpha(p))$ is a prime of I_G. Hence λ maps $\pi(G)$ into $\pi(I_G)$. Let $\mu = \beta \circ \gamma$. By Lemma B.12, if $e \in \pi(I_G)$, then $\gamma(e)$ is a maximal solution of S; and by Lemma B.6, $\mu(e) = \beta(\gamma(e))$ is a prime of G. Hence μ maps $\pi(I_G)$ into $\pi(G)$. We now show that (λ, μ) are an isomorphism pair.

Let $e \in \pi(I_G)$; then $(\lambda \circ \mu)(e) = (\delta \circ \alpha) \circ (\beta \circ \gamma)(e) = \delta \circ (\alpha \circ \beta)(\gamma(e)) \overset{B.6}{=} \delta(\gamma(e)) \overset{B.12}{=} e$. Let $p \in \pi(G)$; then $(\mu \circ \lambda)(p) = ((\beta \circ \gamma) \circ (\delta \circ \alpha))(p) = (\beta \circ (\gamma \circ \delta))(\alpha(p)) \overset{B.7}{=} \beta(\alpha(p)) \overset{B.6}{=} p$. Hence $\pi(G)$ and $\pi(I_G)$ are isomorphic.

Proof of Theorem 3.8. Lemma B.14 has established that $\pi(G)$ and $\pi(I_G)$ are isomorphic; if e is a prime of I_G, then $p = \beta(\gamma(e))$ is a prime of G.

Let $p = \Pi_{j=1}^n x_j^{A_j}$. Then $A_j = \hat{\beta}_j(\hat{\gamma}_j(e^j))$, where $e = e^1 \ldots e^n$, and $e^j = (e_0^j, e_k^j)$, with e_k^j the exponent of z_{kj}. Suppose that z'_{0j} appears in e; then $e_0^j = \{0\}$, $e_1^j = D$, because

(a) $e^j_k \neq \{1\}$, and (b) $e^j_k = \{0\}$ would mean that $z'_{0j}\, z'_{1j}$ appears in e, a contradiction to the definition of I_G and the primality of e. Then $A_j = \hat{\beta}_j(\hat{\gamma}_j(\{0\}), D)) = \hat{\beta}_j(01) = \{1\} = \{1-0\}$, i.e. each z'_{0j} in e is replaced by $x_j = x_j^{1-0}$ in p, Q.E.D. Similarly, if z'_{1j} appears in e, $e^j = (D, \{0\})$, and $A_j = \hat{\beta}_j(\hat{\gamma}_j(D, \{0\})) = \hat{\beta}_j(10) = \{0\} = \{1-1\}$, i.e. z'_{1j} in e is replaced by $x'_j = x_j^{1-1}$ in p. Finally, if neither z'_{0j} nor z'_{1j} appear in e, then $e^j = (D, D)$, and $A_j = \hat{\beta}_j(\hat{\gamma}_j(D, D)) = \hat{\beta}_j(11) = D$, i.e. x_j does not appear in p either. Hence, in all cases, z'_{kj} appears in e if, and only if, x_j^{1-k} appears in p, Q.E.D.

Theorem 3.9 (Brayton et al., 1984, section 3.1). Let G be a cover, x a variable in G. Then (a) $G' \sim x(G')_x + x'(G')_{x'}$; (b) $(G'_x) \sim (G_x)'$.

Proof. For (a) we complement the identity $G = xG_x + x'Gx'$, to obtain $G' = [x' + (G_x)'][x + G_{x'})'] = x(G_x)' + x'(G_{x'})' + (G_x)'(G_{x'})' \sim x(G_x)' + x'(G_{x'})'$, where the last step follows from the identity $xp + x'q + pq \sim xp + x'q$. Hence

$$G' \sim x(G_x)' + x'(G_{x'})'. \tag{1}$$

At the same time, we have the identity

$$G' \sim x(G')_x + x'(G')_{x'}. \tag{2}$$

To show (b), let $u \in D^{n-1}$ be a vector with the x component missing. Then by (1), $G'(1, u) = (G_x)'(u)$, while by (2) $G'(1, u) = (G')_x(u)$. Hence for all $u \in D^{n-1}$, $(G_x)'(u) = (G')_x(u)$, i.e. $(G_x)' \sim (G')_x$. Similarly, $(G_{x'})' \sim (G')_{x'}$.

Theorem 3.10 (ibid.).

(a) If G is monotone increasing in x, then $G' \sim x'G'_{x'} + G'_x$.

(b) If G is monotone decreasing in x, then $G' \sim xG'_x + G'_{x'}$.

Proof. (a) If G is increasing in x, then no cube of G contains x'. We can thus write $G = H + R$, where H consists of all cubes that contain x, and R of those that don't. Note that H can be written as $H = xS$, since each cube in H contains x. Then $G_x = S + R$, $G_{x'} = R$. Hence $G = xG_x + x'G_{x'} = xS + xR + x'\,R = xS + R = xS + xR + RS + R = (x + R)(S + R) = (x + G_{x'})G_x$. Complementing this identity, we obtain $G' = x'G'_x + G'_x$. Similarly for (b).

Appendix C

This Appendix provides proofs of Theorems 4.1 and 4.2.

Theorem 4.1. Let F be a cover and q a cube. Then $q \leq F$ if, and only if, F_q is a tautology.

The proof splits into several lemmas.

Lemma C.1. Let $p = \Pi\, x_j^{A_j}$, $q = \Pi\, x_j^{B_j}$, $d(p,q) = 0$, $p_q = \Pi\, x_j^{E_j}$. Then $E_j = D$ if $B_j \neq D$, and $E_j = A_j$ if $B_j = D$.

Proof. Since $d(p,q) = 0$, $E_j = A_j \cup B_j'$. If $B_j \neq D$, then $A_j \cap B_j \neq \emptyset$ implies either $A_j = B_j$ or $A_j = D$; in both cases, $A_j \cup B_j' = D$. If $B_j = D$, then $C_j = A_j \cup B_j' = A_j \cup D' = A_j$.

Lemma C.2. Let p, q, r be cubes. Then $(pq)_r = p_r q_r$.

Proof.

Case 1: $d(p,r) \geq 1$. Then $p_r q_r = 0 q_r = 0$. Since $pq \leq p$, $d(pq,r) \geq d(p,r) \geq 1$, hence $(pq)_r = 0$.

Case 2: $d(q,r) \geq 1$. Similar to Case 1.

Case 3: $d(p,r) = d(q,r) = 0$; $d(p,q) \geq 1$. Then $pq = 0$, so $(pq)_r = 0$. Let A_j, B_j, $C_j \subseteq D$ be the exponents of x_j in p, q, r, respectively. Let j be such that $A_j \cap B_j = \emptyset$ (its existence follows from $d(p,q) \geq 1$); without loss of generality, let $A_j = \{0\}$, $B_j = \{1\}$. By Lemma C.1, $q_r^j = p_r^j = D$ if $C_j \neq D$; and if $C_j = D$, then $p_r^j = A_j = \{0\}$, $q_r^j = B_j = \{1\}$. Since $d(p,r) = 0$, $A_j \cap C_j \neq \emptyset$, i.e. $0 \in C_j$; since $d(q,r) = 0$, $B_j \cap C_j \neq \emptyset$, i.e. $1 \in C_j$. Hence $C_j = D$, and it follows that $p_r^j \cap q_r^j = \emptyset$, i.e. $p_r q_r = 0 = (pq)_r$.

Case 4: $d(p,r) = d(q,r) = d(p,q) = 0$; $d(pq,r) \geq 1$. This is an impossible case. To see this, let A_j, B_j, C_j stand for the exponent of x_j in p, q, r, respectively. If $d(pq,r) \geq 1$, then there exists a j such that $A_j \cap B_j \cap C_j = \emptyset$, i.e. $(A_j \cap C_j) \cap (B_j \cap C_j) = \emptyset$; without loss of generality, let $A_j \cap C_j = \{1\}$, $B_j \cap C_j = \{0\}$; hence $C_j = D$, $A_j = \{1\}$, $B_j = \{0\}$, i.e. $A_j \cap B_j = \emptyset$, contradicting $d(p,q) = 0$.

Case 5: $d(p,r) = d(q,r) = d(p,q) = d(pq,r) = 0$. We show that for each j, $(pq)_r^j = p_r^j \cap q_r^j$. If $C_j \neq D$, then by Lemma C.1, $p_r^j = q_r^j = (pq)_r^j = D$. If $C_j = D$, then by Lemma C.1, $p_r^j = A_j$, $q_r^j = B_j$, $(pq)_r^j = A_j \cap B_j$.

Lemma C.3. Let F be a cover and p, r be cubes. Then $(pF)_r = p_r F_r$.

Proof. $(pF)_r = (\sum_{q\in F} pq)_r$. By the definition of the cofactor of a cover, $(\sum_{q\in F} pq)_r = \sum_{q\in F}(pq)_r$; and by Lemma C.2, $(pq)_r = p_r q_r$. Hence $p_r F_r = p_r \sum_{q\in F} q_r = \sum_{q\in F} p_r q_r = \sum_{q\in F}(pq)_r = (\sum_{q\in F} pq)_r = (pF)_r$.

Lemma C.4. Let F be a cover and p a cube. Then $pF = pF_p$.

Proof. Note that if $d(p,q) \geq 1$, $q \in F$, then $pq = 0$ and $q_p = 0$. Hence, if $R = \{q \in F : d(p,q) = 0\}$, and $G = \sum_{q\in R} q$, it suffices to show $pG = pG_p$. Let A_j, B_j^q be the exponents of x_j in p, q, respectively. By Lemma C.1, $q_p^j = D$ if $A_j \neq D$, and $q_p^j = B_j^q$ if $A_j = D$. Hence $(pq_p)^j = A_j \cap q_p^j = A_j$ if $A_j \neq D$; and $(pq_p)^j = B_j^q$ if $A_j = D$. In both cases, $(pq_p)^j = A_j \cap B_j^q$. Since $(pq)^j = A_j \cap B_j^q$ by definition, we obtain $pq = pq_p$ $\forall q \in R$. Hence $pG = pG_p$.

Lemma C.5. $p \leq F$ if, and only if, $pF \sim p$.

Proof. Clearly $pF \leq p$, so we need only show $p \leq F \Leftrightarrow p \leq pF$, or equivalently $pF' = 0 \Leftrightarrow p(pF)' = 0$. Since $(pF)' = p' + F'$ and $pp' = 0$, the equivalence is obvious.

Proof of Theorem 3.8. $p \leq F \overset{C.5}{\Longrightarrow} pF \sim p \Rightarrow (pF)_p \sim p_p \overset{C.3}{\Longrightarrow} p_p F_p \sim p_p \Rightarrow F_p \sim 1$. For the converse, $F_p \sim 1 \Rightarrow pF_p \sim p \overset{C.4}{\Longrightarrow} pF \sim p \overset{C.5}{\Longrightarrow} p \leq F$.

Theorem 4.2. Let F be a cover; $E = \{p \in F : p \not\leq (F\backslash p)\}$ the set of its relatively essential cubes; $R = \{p \in F\backslash E : p \not\leq E\}$ the set of its partially redundant cubes; and A its (reduced–size) covering matrix. Let x^* solve the $0-1$ linear programming problem $\min \sum_{p\in R} x_p$ subject to $Ax \geq 1$; let supp $x^* = \{p \in R : x_p^* = 1\}$. Then $E+$ supp x^* is a minimum–cost cover equivalent to F. □

Before proceeding with the proof, A is formally defined.

Definition C.1. For each cube r in R, let $\varphi(r)$ consist of all subsets S of R that satisfy

(a) $F \not\leq E + (R\backslash S)$;

(b) if $T \subseteq S$ and $F \not\leq E + (R\backslash T)$, then $T = S$.

Definition C.2. The (reduced–size) covering matrix A of F has rows in $1-1$ correspondence with the elements of graph (φ), and columns in $1-1$ correspondence with cubes in R. Its entries are defined as follows: For each $p \in R$ and $C \in \varphi(r)$, $r \in R$, $A_{rC,p} = 1$ if $p = r$ or $p \in C$; and $A_{rC,p} = 0$ otherwise.

Definition C.3. The feasible set of the linear programming problem of Theorem 4.2 is

$H = \{x \in D^R : Ax \geq 1\}$. The set B is defined by $B = \{x \in D^R : F \leq E + \text{supp } x\}$.

Lemma C.6. $H = B$.

Proof. Let $x \in H$; suppose, for contradiction, that $x \notin B$. By the contradiction hypothesis, $R + E \sim F \not\leq E + \text{supp } x$; hence there exists r in R such that $r \not\leq E + \text{supp } x$. Recall that supp $x = \{p \in R : x_p = 1\} \subseteq R$; it follows that supp $x = (\text{supp } x)'' = R \backslash (\text{supp } x)'$, and that $r \not\leq E + R\backslash(\text{supp } x)'$. Let $C \subseteq (\text{supp } x)'$ be a minimal set with this property; then $C \in \varphi(r)$ by definition C.1. Since $Ax \geq 1$, we must have $\sum_{p \in R} A_{rC,p} x_p \geq 1$, i.e. there must exist $p \in R$ with $x_p = 1$ and either $p = r$ or $p \in C$. If $p = r$, then we obtain a contradiction, because $x_p = 1$, $p = r$ imply $r \in \text{supp } x$, thus contradicting the fact that $r \not\leq E + \text{supp } x$. If $p \in C$, then again we obtain a contradiction, because $x_p = 1$ implies $p \in \text{supp } (x)$, while $p \in C \subseteq (\text{supp } x)'$ imply $p \notin \text{supp } (x)$. Hence $H \subseteq B$.

For the converse, let $x \in B$; suppose, for contradiction, that $x \notin H$. By the contradiction hypothesis, there is some $r \in R$, $C \in \varphi(r)$ such that $A_{rC,p} x_p = 0$ for all $p \in R$. By Definition C.2 this implies that $p = r$ or $p \in C$ imply $x_p = 0$, i.e. $C \cup \{r\} \subseteq (\text{supp } x)'$. The fact that $x \in B$ implies $r \leq F \leq E + \text{supp } x$, i.e. $r \leq E + R\backslash(\text{supp } x)' \leq E + (R \backslash C)$, a contradiction to $C \in \varphi(r)$. Hence $B \subseteq H$.

Proof of Theorem 4.2. If x^* solves $\min \sum_{p \in R} x_p$ subject to $Ax \leq 1$, then by Lemma C.6 and $Ax^* \geq 1$ we obtain $F \leq E + \text{supp } x^*$; since supp $x^* \subseteq R \subseteq F$, $F \sim E + \text{supp } x^*$. If $T \subseteq R$ satisfies $F \sim E + T$ and contains fewer cubes than supp x^*, then x^T, defined by $x_p^T = 1$ iff $p \in T$, satisfies $Ax^T \geq 1$ by Lemma C.6; and $\sum_{p \in R} x_p^T < \sum_{p \in R} x_p^*$, thus contradicting the optimality of x^*. Hence $E + \text{supp } x^*$ is a minimum–cost cover equivalent to F.

Acknowledgements

I would like to thank Benjamin Coriat, Giovanni Dosi, and Ed Green for useful conversations; and Barbara Bonke for valuable technical assistance. I am solely responsible for the contents of this paper.

References

Automotive Industries, (October) 1995, European Efficiency Lags Japan, U.S., 48.

Balas, E. and A. Ho, 1980, Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization, Mathematical Programming Study 12, 37–60.

Baldwin, C. and K. Clark, 1994, Capital Budgeting systems and Capabilities Investments in U.S. Companies After the Second World War, Business History Review 68, 73–109.

Brayton, R., G. Hachtel, C. McMullen and A. Sangiovanni–Vincentelli, 1984, Logic Minimization Algorithms for VLSI Synthesis (Kluwer, Boston MA).

Clark, K. and T. Fujimoto, 1991, Product Development Performance (Harvard Business School Press, Boston MA).

Cormen, T., C. Leiserson and R. Rivest, 1990, Introduction to Algorithms (MIT Press, Boston MA).

De Micheli, G., 1993, Synthesis and Optimization of Digital Circuits (McGraw Hill, New York).

Drucker, P., 1991, The New Productivity Challenge, Harvard Business Review (November–December), 69–79.

Dyer, J. 1994, Dedicated Assets: Japan's Manufacturing Edge, Harvard Business Review (November–December), 174–178.

The Economist, (March 4) 1995, The Kindergarten that will Change the World, 67–68.

The Economist, (October 18) 1997, A Fun Drive While it Lasted, 84.

Feitzinger, E. and H. Lee, 1997, Mass Customization at Hewlett–Packard, Harvard Business Review (January–February), 116–121.

Hammer, M. and J. Champy, 1993, Reengineering the Corporation (N. Brealey, London).

Hauser, J. and D. Clausing, 1988, The House of Quality, Harvard Business Review (May–June), 63–73.

Hayek, F., 1948, Socialist Calculation III: The Competitive "Solution", in: F. Hayek, Individualism and Economic Order (University of Chicago Press, Chicago).

Hayes, R. and W. Abernathy, 1980, Managing our Way to Economic Decline, Harvard Business Review (July–August), 67–77.

Henderson, R. and K. Clark, 1990, Architectural Innovation, Administrative Science Quarterly 35, 9–30.

Hill, G. and F. Peterson , 1993, Introduction to VLSI Design (Prentice Hall, New York).

Hurst, S., D. Miller and J. Muzio, 1985, Spectral Techniques in Digital Logic (Academic Press, London).

Ingrassia, P. and J. White, 1994, Comeback. The Fall and Rise of the American Automobile Industry (Simon and Schuster, New York).

Jensen, M., 1993, the Modern Industrial Revolution, Exit, and the Failure of Internal Control Systems, The Journal of Finance (July), 831–874.

Keller, M., 1990, Rude Awakening. The Rise, Fall, and Struggle for Recovery of General Motors (Harper Perennial, New York).

Lengauer, T., 1990, VLSI Theory, in: J. van Leeuwen, Handbook of Theoretical Computer Science (Elsevier).

Milgrom, P. and J. Roberts, 1992, Economics, Organization and Management (Prentice Hall, New Jersey).

Morton, O., 1994, Manufacturing Technology, The Economist (March 5), 1–20.

Naughton, K., 1995, Ford's Global Gladiator, Business Week (December 11).

Okino, S., 1995, Less is More, Automotive Industries (March), 81–82.

Peters, T., 1993, Liberation Management (MacMillan, London).

Ruddel, R. and A. Sangiovanni–Vincentelli, 1987, Multiple–Valued Minimization for PLA Optimization, IEEE Transactions on Computer–Aided Design, Vol. CAD–6, No. 5 (September).

Shingo, S., 1989, A Study of the Toyota Production System (Productivity Press, Portland, Oregon).

Simon, H., 1973, The Structure of Ill–structured Problems, Artificial Intelligence 4, 181–201.

Simon, H. 1981, The Sciences of the Artificial, 2ns Edition (MIT Press, Cambridge MA).

Solow, R., 1994, Perspectives on Growth Theory, Journal of Economic Perspectives 8, 45–54.

Taylor III, A., 1992, U.S. Cars Come Back, Fortune (November 16), 52–85.

Taylor III, A., 1995, GM: Some Gain Much Pain, Fortune (May 29), 46–50.

Taylor III, A., 1996, GM: Why They Might Break Up America's Biggest Company, Fortune (April 29), 38–43.

Taylor III, A., 1997 (a), GM: Time to Get in Gear, Fortune (April 8), 60-65.

Taylor III, A., 1997 (b), How Toyota Defies Gravity, Fortune (December 8).

Ulrich, K., 1995, The Role of Product Architecture in the Manufacturing Firm, Research Policy 24, 419–440.

Ward, A., J. Liker, J. Cristiano and D. Sobek, 1995, The Second Toyota Paradox, Sloan Management Review (Spring), 43–61.

Wegener, I., 1987, The Complexity of Boolean Functions (Teubner, Stuttgart).

Whitney, D., 1988, Manufacturing by Design, Harvard Business Review (July–August), 83–91.

Whitney, D., 1995, Nippondenso Co. Ltd: A Case Study of Strategic Product Design, in: J. Liker, J. Ettlie and J. Campbell, eds., Engineered in Japan (Oxford University Press, New York), Chapter 6.

Williams, K., D. Haslam, S. Johal and J. Williams, 1994, Cars. Analysis, History, Cases (Berghahn Books, Providence).

Womack, J., D. Jones and D. Roos, 1990, The Machine that Changed the World (Rawson Associates, New York).

Womack, J. and D. Jones, 1996, Lean Thinking (Simon and Schuster, New York).