# Platform Strategies in the Electronic Design Automation Industry

by

Arthur Low

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Technology Innovation Management

Carleton University
Ottawa Ontario

The undersigned hereby recommend to

The Faculty of Graduate and Postdoctoral Affairs

Acceptance of the thesis

# Platform strategies in the electronic design automation industry

# by

# Arthur Low

in partial fulfillment of the requirements for the degree of

**Master of Applied Science in Technology Innovation Management**

---

**Antonio J. Bailetti, Director**

**Institute of Technology Entrepreneurship and Commercialization**

---

**Steven Muegge, Thesis Supervisor**

Carleton University

September 2013

# Abstract

Platforms – architectures of related standards that allow modular substitution of complementary assets – feature prominently in technology-intensive industries. The motivations for firms to adopt a particular platform strategy and the ways in which platform strategies change over time are not fully understood. This thesis examines the platform strategies of three leading vendors in the Electronic Design Automation (EDA) industry from 1987 to 2002. It employs a two-part research design: (i) pattern-matching to operationalize and test a three-stage explanation previously developed by West (2003) to account for the evolution of platform strategies by firms in the computer industry, followed by (ii) explanation-building to account for differences between observations and the expected pattern. The pioneering EDA firm matches the expected pattern, but two other EDA firms bypass stage one to enter at stage two with open standards. All three firms later move to stage three simultaneously by adopting hybrid open source strategies.

## Acknowledgements

I would like to acknowledge my sons, Connor and Jake, for their tolerance, and my thesis advisor, Dr. Steven Muegge, for his patient guidance.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1 Introduction

Platforms are architectures of related standards that provide the basis for developing complementary assets (West, 2003). According to Evans et al. (2006), platforms are the "invisible engines" that drive innovation in technology-intensive industries, such as personal computers, video game consoles, mobile telephony, and Internet-based software. By building products and services on top of existing platform assets, an innovator can focus their R&D effort narrowly and efficiently on building differentiating capability (Muegge, 2013).

West (2003, p. 1259) describes the "essential tension" between *appropriability* and *adoption* faced by firms that employ a platform strategy:

> *To recoup the costs of developing a platform, its sponsor must be able to appropriate for itself some portion of the economic benefits of that platform. But to obtain any returns at all, the sponsor must get the platform adopted, which requires sharing the economic returns with buyers and other members of the value chain. The proprietary and open source strategies correspond to the two extremes of this trade-off. In making a platform strategy for the 21st century, leading computer vendors face a dilemma of how much is open enough to attract enough buyers while retaining adequate returns.*

From close study of Apple, IBM, and Sun Microsystems, West induced a three-stage explanation of platform strategies in the computer industry: platform strategies evolved in three distinct stages from proprietary to open standards to open source, motivated by a combination of technical and economic factors. To test West's explanation in another industry context, this thesis develops a methodology for answering important questions where strong opinions and personal experience tend to prevent objective analysis.

This thesis examines platform strategies in the Electronic Design Automation (EDA) industry – a highly specialized industry sector that produces the software tools used to design electronic systems such as printed circuit boards, integrated circuits, and programmable gate array devices (Low, 2012). EDA software can be usefully segmented into six architectural layers, and this thesis examines the interfaces *between* layers ("standards") and the implementation *within* layers ("source") of the three leading EDA platforms over the time period from 1987 to 2002. It conducts an empirical test of West's explanation, and develops refinements to account for differences between observations and the predictions implied by West's explanation.

## 1.1 Objective

This thesis examines the platform strategies of leading firms in the Electronic Design Automation (EDA) industry. More specifically, it examines the extent to which West's (2003) three-stage explanation of platform strategies in the computer industry can account for the strategic actions of the three leading firms in the EDA industry.

## 1.2 Deliverables

There are five deliverables:

1. A *pattern* of expected observations implied by West (2003); this pattern provides an operational test of West's explanation.
2. A *data set* and *narrative* that document the platform strategies of the three leading EDA firms from 1987 to 2002.

3. *Results* of the pattern-matching test comparing the data to the expected pattern.

4. An *explanation* of the observations and test results.

5. Refinements to West (2003).

## 1.3 Relevance

Understanding the evolution of platform strategies in the EDA industry is relevant to at least four groups of stakeholders. First, *top management teams* at technology companies need to formulate platform strategies that strike the right balance between appropriability and adoption; these decision-makers benefit from a new set of cases on platform strategies in an industry not previously examined. Managers in the EDA industry benefit directly; managers in other industries may discover insights that can applied in their own context. Second, *management researchers* benefit from refinements to West's explanation of platform evolution and insights into the explanation's validity. Third, *software managers* and *software architects* need to make technical decisions about interfaces and implementations; these decision-makers benefit from better connecting the technical decisions to strategic considerations and implications. Fourth, *business historians* will benefit from a rigorous and well-sourced multiple case study of an industry sector not previously examined from the perspective of platforms, interfaces, and implementations.

## 1.4 Overview of Research Method

This thesis employs a two-part research design that employs case methods for both theory-testing and theory-building. The first part employs a *pattern-matching approach* (Yin, 2003, pp. 116-120) to examine the extent to which West's (2003) explanation of evolving platform strategies in the computer industry can account for the platform strategies of the three leading firms in the EDA industry. The second part employs an *explanation-building approach* (Yin, 2003, pp. 120-122) to develop explanations for anomalies not explained by West (2003). Both stages employ case studies developed from publicly-available archival sources, including company shareholder reports and press releases, and articles published in trade journals and newspapers.

## 1.5 Summary of Key Findings

West (2003) explained the evolution of platform strategies as a three-stage linear progression from proprietary platforms to open standards to open source, with transitions motivated by a combination of four technical and economic factors. In West's study of the computer industry, this explanation accounted for the platform strategies of three leading firms: IBM, Apple, and Sun Microsystems (West, 2003). This thesis finds *greater diversity* of the evolution of platform strategies by the three leading firms of the EDA industry, and develops a refinement to West's explanation in order to account for these results.

One EDA platform vendor, Mentor Graphics, evolved its platform strategy as

 

predicted by West. This vendor established a successful proprietary platform consisting of an architecture of related standards that it controlled, which created high barriers to imitation and generated large profits. This EDA vendor was initially successful with a strategy based on the control of data and the bundling of tools.

The second EDA platform vendor, Cadence Design Systems, introduced new technology that permitted the unbundling of EDA tools, and entered the market with a platform strategy utilizing open standards. Customers of this vendor could select the best third-party tools. The success of this open standards-based platform strategy created strong motivations for the first EDA platform vendor to further open its platform to standards based data formats and allow third-party software to be run with direct access to data from the once proprietary platform.

The third EDA platform vendor, Synopsys, created an innovation that enabled a new kind of integrated circuit design methodology based on synthesis using hardware description languages (HDLs), which were open standards. It dominated this niche market, which enabled customers of the other EDA platforms to select this technology. Synopsys later acquired other companies (with interoperable products at different architectural layers) to establish its own viable EDA platform.

## 1.6 Contribution

This thesis makes four contributions to research methodology, and two contributions to management theory and one contribution to management practice.

The first contribution to research methodology is the research method; though

developed to answer a specific research question about the evolution of platform strategies over time, this method can be applied more generally to other research questions about industry evolution and change, especially where some industrial expertise is required to identify relevant data to collect. The second contribution to research methodology is the use of pre-defined rules for the post-processing of collected data; these rules can be applied by persons who are not subject matter experts, or by persons who are subject matter experts but may have strongly-held opinions which might introduce bias. The third contribution to research methodology is the specific pattern-matching test developed here. Yin (2003) documents the pattern-matching research approach, which has been employed successfully in other domains of inquiry, but it had not previously been employed in research on platforms; other platform researchers could replicate or adapt this test for explanations of platform strategies in other industry contexts. The fourth contribution to research methodology is the use of the Herfindahl-Hirschman Index as a quantitative measure of platform market power; this measure was not previously used in the platforms literature.

The first contribution to theory is a sharper understanding of the validity of West's explanation. The underlying logic of West's explanation is applicable outside the computer industry; it is valid also in the EDA industry. The second contribution to theory is a refinement of West's explanation. If a “stage 1” proprietary platform strategy is already infeasible for a combination of technical and economic factors, a firm can enter with a “stage 2” open standards strategy. Although this new path is consistent with the

 

logic underlying West's explanation, it was not found in West's case data, thus is absent from the explanation. This study makes that path explicit, with evidential support from new case data.

The contribution to management practice is the case results reported in chapter 4, including the chronology of the EDA industry and the case reports on the platform strategies of Mentor Graphics, Cadence Design Systems, and Synopsys. This thesis describes how West's explanation can be systematically applied to provides managers in the EDA platform industry with an analytic tool with which they can better understand how market forces shape platform evolution. Early identification of the market forces that affect platform feasibility can enable faster and more methodical decisions on how and when the platform can be modified to respond to emerging threats. architectures, competitive strategy, and anticipating threats from rivals and new entrants.

## 1.7 Organization of this document

The body of this document is organized as seven chapters. Chapter 2 reviews the scholarly and practitioner literature on platform strategies, minimum efficient scale and market power, case study methods of pattern-matching and explanation-building, and process theories. Chapter 3 specifies the research design and method used to produce the thesis deliverables. Chapter 4 presents the results on the platform strategies of the three leading firms in the EDA industry and the results of a pattern-matching test comparing the observed results with the expected results predicted by West's (2003) explanation of

evolving platform strategies. Chapter 5 develops explanations for the results reported in chapter 4, attending especially to the departures from the predicted pattern. Chapter 6 discusses the findings and presents some lessons learned in the development of this thesis. Chapter 7 concludes.

# *Chapter 2 Literature Review*

This chapter reviews the salient findings from the scholarly literature. It is organized in five sections. The first section reviews the literature on software platform strategies, including network externalities, standards, and the motivations for firms to employ open standards. The second section reviews the literature on market power and minimum efficient scale – concepts used later in data collection and analysis. The third section reviews the literature on case methods, including the analytic strategies of pattern-matching and explanation-building that are central to the research method, and the threats to validity and reliability of theory built from case data. The fourth section reviews the literature on process study methods, including techniques employed later to develop event time-lines and explanations of time-lines. The fifth section is a summary and synthesis of key insights from the literature.

## 2.1 Platform Strategies, Network Externalities, Motivations and Standards

This section presents in concise statements a summary of the salient literature on platforms, emphasizing the key concepts required to understand how platforms operate in a market environment.

Platforms, such as computer systems, are innovations that require complementary assets for the platform to be successfully commercialized (Evans et al. 2006; Gawer & Henderson, 2007; Gawer, 2009; Cusumano, 2010; Muegge, 2013). A proprietary platform is an architecture of related standards controlled by one or more sponsoring firms (West, 2003). Platforms must be hard to imitate, and the firm must have strong intellectual

property rights (IPR) to appropriate economic returns (Teece, 1986). Continued platform success requires that the firm must maintain control of the access to its complementary assets (Teece, 1986). Control of complementary assets is determined by the ability to create and evolve application programming interfaces (APIs) to the operating system (OS) (West & Dedrick, 2000). New platforms succeed by tapping unserved market niches to avoid competition until reaching critical mass (Bresnahan & Greenstein, 1999).

A network externality refers to the instability due to the re-cycling via positive feedback of platform success, coupled with the development of co-specialized (complementary) assets, that leads to the emergence of a de facto standard (Katz & Shapiro, 1985). Complements are more likely to be developed internally and by third-parties with a large software network (Katz & Shapiro, 1998). In terms of software compatibility, if two software systems can exchange data, they are said to be on the same network. Benefits to membership in the network increases as more users join the network. Wide adoption therefore increases the value of the software (Katz & Shapiro, 1985). Software markets feature strong network effects, which tend to increase market concentration (Katz & Shapiro, 1998).

The instability and positive feedback of success makes the contest tip to the technology leader (Besen & Farrell, 1994). Positive feedback that can lead to tipping is important to anti-trust analysis, standard-setting and compatibility. With software compatibility, there is one big network and tipping to a single variant is impossible. Markets may tip without compatibility (Katz & Shapiro, 1998). In a standards

 

competition, strong network effects in a market with natural monopoly elements will lead to the survival of just one industry standard (Liebowitz & Margolis, 1999).

Network effects drive expectation. With network effects, rational buyers will base their choices on expected network sizes. Therefore, drivers of expectations may have significant roles, such as pre-announcements, installed bases, current product attributes (Katz & Shapiro, 1998). Regarding markets for software platforms, marginal costs are very low, which imply that positive returns to scale coupled with strong network effects lead to very high profits (Katz & Shapiro, 1998).

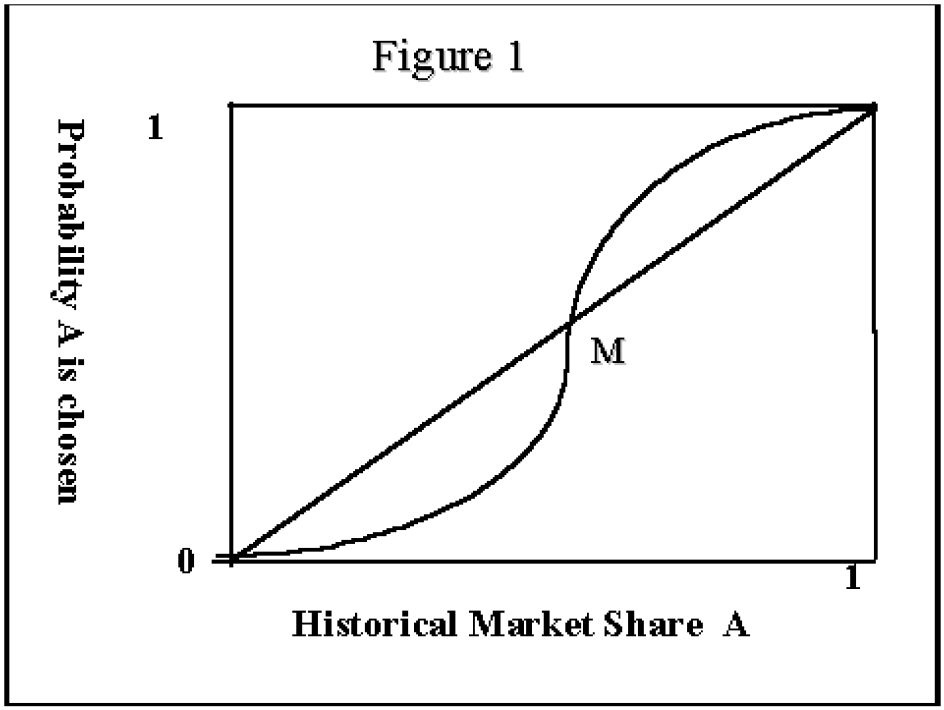


*Figure 1: Network Effects (Adapted from Liebowitz & Margolis, 1999)*

The S-curve in Figure 1 shows how the network effects concept can be applied to the determination of a de facto standard. When adoption of software system rises past a certain threshold (near M in the above illustration), the pace of adoption increases rapidly as the majority of undecided users selects the technology leader.

Increasing returns to scale are likely if the business can exploit positive network externalities to increase economic returns by adaptation as the dominant design emerges. The coupling of strong network effects with high switching costs between standards and high up front R&D costs lead to a dominant technology showing increasing returns to scale that magnify an early technological lead. A business with positive network externalities increases economic returns by adaptation as a dominant design emerges (Arthur, 1996).

Platform leadership is unstable when competing firms control different layers of the standards architecture. Control can shift without affecting the buyer's value proposition (Greenstein, 1997). High entry costs allow the pioneer and early challengers to form a stable oligopoly (Bresnahan & Greenstein, 1999). Late entrants have trouble gaining market share due to switching costs (Greenstein, 1997).

A firm with a larger market share and higher profits has little incentive to collaborate on a standard that will effectively reduce its market share, to the benefit of its rival and the consumer. Firms with large networks and good reputations find it is in their interest not to be compatible with other firm's products. On the other hand, firms with

small networks and less established reputations will be motivated strongly to make their products compatible with those of other firms. A firm with dominant market power has an interest that its platform remains the de facto standard (Katz & Shario, 1985).

## 2.2 Measuring Minimum Efficient Scale and Market Power in Software Markets

Software markets tend to be concentrated (Katz & Shapiro, 1998). With near-zero manufacturing costs, software has low marginal cost relative to average costs, which includes development costs plus other costs divided by the number of software copies released to the market. Therefore, to determine minimum efficient scale for software markets, it is better to consider average costs. Gross margin is very high in the software industry. As the adoption of a software platforms increases, the positive returns to scale coupled with strong network effects lead to very high profits.

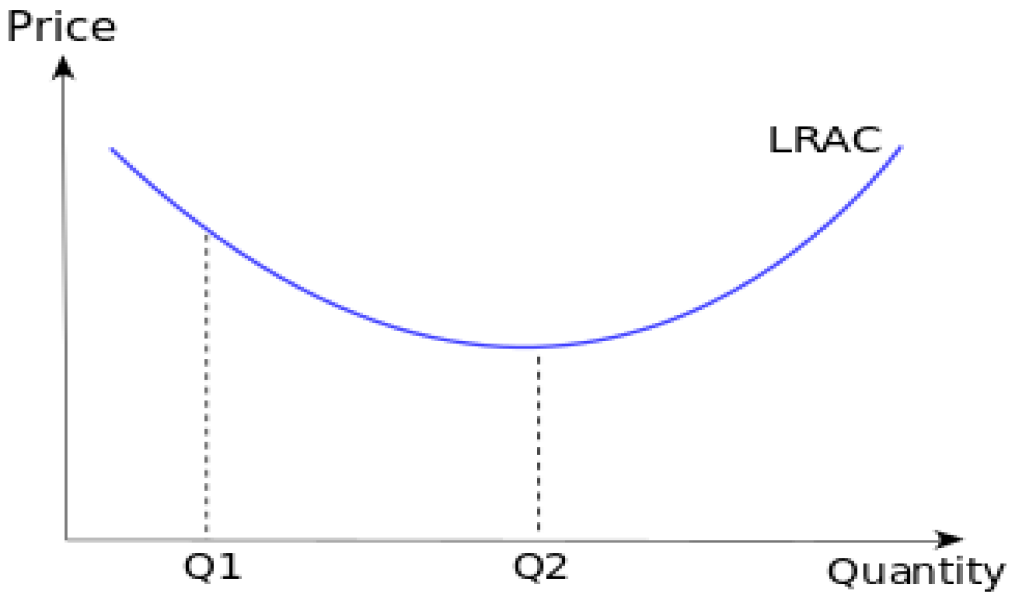


*Figure 2: Average Cost Curve (Adapted from Wikipedia)*

With perfect competition, long terms average profits are zero, and minimum efficient scale is defined when the slope of the the long run average cost curve (LRAC) is zero, which is shown as at Q2 in the above illustration: The minimum efficient scale is generally reached when the marginal cost equals the average cost of production. Increasing returns to scale is experienced by the movement along the negative slope of the above curve (in the direction from Q1 to Q2).

The United States Department of Justice (DoJ) presumes that a merger is "likely to enhance market power" if it produces an increase in the Herfindahl-Hirschman Index (HHI) of more than 200 points in a highly concentrated market. Three HHI ranges are defined: not concentrated (under 1500), moderately concentrated (between 1500 and 2500) and highly concentrated (more than 2500). The HHI is employed here as a measure of platform market power.

The HHI contribution of a firm is the square of its market share. If four firms split the market evenly, then the HHI is 2500 (highly concentrated). If three firms have each 20% market share, and 10 other firms evenly split the rest of the market, the HHI = $3 \times 20^2 + 10 \times 4^2 = 1360$. Now, if one of the larger firms acquires one of the smaller firms, the new HHI = $2 \times 20^2 + 1 \times 24^2 + 9 \times 4^2 = 1520$. This acquisition changes the market concentration to moderately concentrated, from not concentrated. If the same firm were to have acquired also another of the smaller firms, the new HHI = $2 \times 20^2 + 1 \times 28^2 + 8 \times 4^2 = 1712$, and the change in the HHI is almost +200 points. This shows that the acquiring

firm acts to concentrate the market, and by so doing, its market power is “likely enhanced”.

For such market share analysis to be informative of market power, market share must not shift markedly over time. What matters most are what assets the various firms bring to future competitions. Then, market share represents how much control the firm has over those assets. Revenues or units sold can be used, but revenues are important if software prices are high (Katz & Shapiro, 1998).

## 2.3 Case Study Research Methods

A case is a study of “a contemporary phenomenon within its real-life context” (Yin, 2003). A clear research framework should demonstrate logical causality, i.e., that x caused y. Patterns, described by earlier research, or predicted in advance of current research, should be compared with empirical data from observations in the field. Evidence can support or refute a pattern, and when this occurs, the investigator needs to seize upon this as something "interesting", which should lead to new questions and attempts to gather evidence along those new lines of inquiry. Collected data, analysis and conclusions must have construct validity, internal validity, external validity, and reliability; each is examined more closely below.

### Construct Validity

Construct validity requires that at least one operational theory must be specified before the data collection phase begins, but one should also consider less likely rival

theories. A theory and its rivals leads to questions that require evidence to answer. Intelligent inferences more easily follow from application of a guiding theory, which is a construct that makes sense of the evidence. Without a guiding theory, the investigation can easily be side-tracked when the actual data collected varies somewhat from what was expected.

Rival theories are as important to have as the more likely theory, because without rival theories the investigator is more inclined to conduct research with conscious or unconscious bias might lead to the ignoring of important evidence when it does not support the likely theory. Criticism of case studies is stronger when the case was developed without considering rival theories. With rival theories, the investigation is less likely to be biased by evidence selectively gathered or rejected based on its support or contradiction of the theory.

**Internal Validity**

Internal or logical validity applies in the data analysis phase. A researcher must build a case by collecting evidence, including statements made by corporate actors or other persons of interest close to or deeply familiar with the business activity, to develop a probable sequence and timing of events relevant to the specific business decision. A case study makes sense due to its internal validity. The data collection phase is not complete if the case does not hold water due to gaps in the chain of evidence.

The two types of corroborative evidence “triangulate” to strengthen the case. Each type of evidence (there are six types) ideally should be supported by other evidence. An

anonymous engineer's blog outlining a technology limitation is confirmed when a corporate executive makes a press release to announce a new strategy to plug a hole in its offering. The research needs to “read between the lines” of the evidence. In other words, one must infer that if A happened and C happened, then B must have happened. By inferring that B happened, is there evidence to support that inference?

**External Validity**

A single case is often encountered, like a rare disease. The patient's symptoms, personal background, family and medical history, environment and so on can represent the findings of the case. In other words, the physician might say “I don't really know why, but this is what I observed.” Considering multiple case studies, a cross-case analysis can offer powerful support for theoretical implications, such as to favor the environment over heredity as the cause of the disease. When a multiple case study is done combined with cross-case analysis between four to ten case studies, external validity can be demonstrated if a pattern is seen in one case, and is repeated in another and another. Such an analysis can lead to a reasonable generalization as a basis to build a theory that explains the observations (Eisenhardt, 1989).

**Reliability**

Reliability in a case study refers to the transparency and reproducibility of the original researcher's insights if another researcher follows the same steps. An evidence collection protocol designed to (a) reduce the chance that evidence will be compromised,

and (b) to ensure that the evidence collected answers all of the questions developed when planning the case study, and (c) that rival theories are considered.

### 2.3.1 Tables and Time-lines and the Data Collection Process

The credibility of a case study issues from the evidence gathered to support it. Yin refers to Miles & Hubermann (1994) and the use of tables developed as a guide to data collection. Yin also refers to time-lines for events, and in particular the case study method is ideal for analyzing when, how and why strategic decisions were made. The data collected needs to be kept in a case database. Field notes of ideas, possible lines of inquiry, and insights of the moment are important components of the research and data collection process. Such notes can later aid the investigation and analysis, as well as indicate the value of a follow-up research.

### 2.3.2 Literal Replication and Theoretical Replication

Yin (2003) provides a useful nomenclature for categorizing results of case research. In pattern-matching research designs, a case that conforms exactly to a predicted pattern is called a *literal replication*. A case that diverges from a predicted pattern, but for reasons that can be anticipated and explained within the logic of the underlying theory is called a *theoretical replication*. Theoretical replications may motivate refinements to the underlying theory.

Following Carlile & Christensen (2005), a case that diverges from a predicted pattern in ways that cannot be explained within the logic of the underlying theory is

called an *anomaly*. Explaining an anomaly may motivate extensions to the original theory, or rejection of the original theory and development of a new theory.

## 2.4 Process Study Design Methodology

A process study is a research methodology that goes beyond subjective "eyeballing" of raw data to identify patterns (Van de Ven, 2007). Basic concepts and ideas can be developed from raw data. Beginning with small units of data or what can be called incidents one can gradually construct a system of categories or key concepts that describe the phenomena being observed. When a particular process model is known beforehand, categories for data collection and analysis can be developed by stating what kinds of empirical indicators would have to be observed for the theoretical constructs to be operational. As data collection proceeds, the categories are refined. As an example of this, the search term entered into Google can become very precise when the attributes that define an event are better understood in terms of precisely specified categories than limit search results.

A process study may feature few cases and few events, which is typical of comparative case studies. A process of change might be as simple as two stages where one signal event separates stage one from stage two. There may be only a few salient events that indicate a well-defined point that identifies a the start of a decision-making process to change. When there are few cases, but many events, the researcher may need to consider summary measures to collapse the data. Statistical methods are required when

there are many cases and many events.

**Empirical Incidents Support Deduction of Event Constructs**

Events are theoretical constructs that are indicated by incidents. A stream of incidents occurring in places and at times is the first order evidence that a second order sequence of events has occurred. An overlapping series of events pertaining to the negotiation and closing of a business deal, for example, might be deduced from a number of qualified incident reports of business meetings between executives of two different firms.

**Decision Rules for Qualitative Datum Recording**

An incident is recorded as a qualitative datum as a bracketed string of words following a set of "decision rules". As an example, a business incident is said to exist if any one of these five core attributes of changes: innovation, people, transactions, context or outcomes. The decision rule is considered reliable if its classification would enable two different researchers to agree that the incident is properly coded. The decision rule should make sense to someone familiar with the field of study.

**Time-Line of Incidents and Expected Order and Sequence of Events**

Incidents should enable the identification of theoretically meaningful events. The time-lines of the incidents should support the identification of a theoretically meaningful order and sequence of events. In the abductive approach, the incident data is looked at to derive categories from the ground up. In the deductive approach, the theory will specify

the expected order and sequence of events prior to collecting data. Operational templates based on one or more process theories are used to observed how closely an observed event sequence matches theory. For example, using the Dialectic theory, company A would act to make the business environment unstable and company B would react to this instability. Business B might fail to react in time because of Life-cycle dependencies and lose significant market share.

**Code Events Based on Conceptually Relevant Dimensions**

Events can be coded based on several conceptually relevant dimensions that enable a richer description of the process. A process theory needs to penetrate the logic behind the observed temporal progressions of incidents. The surface observations of incidents must give way to the realization an underlying process theory in action.

**Move From Description of Events to a Sense-Making Narrative**

A process theory should include time sequences. There should be a clear beginning, middle and end to the story, where the chronology is the central organizing factor. The goal is to move from a description of a set of incidents in time and place to an sequence of theoretical events that make sense in terms of a story or narrative. The focus should be on one or more key actors such as the protagonist and maybe his antagonist. Character development is not important. What is important is that there is an agent of change that ties the events of the narrative together. The narrative should be made with an identifiable voice or point of view. There should be meaning and cultural value, and

standards should be set by which the character can be judged according to, for example, a sense of right and wrong. The process needs to have a context. The story does not take place without some anchor to reality that sets the events in context.

## 2.5 Summary and Key Insights of the Literature Reviewed

Review of the platforms literature suggests five rules for platform success:

1. A successful platform strategy is hard to imitate (Teece, 1986).
2. The platform vendor must control value-adding complementary assets (Teece, 1986).
3. The platform vendor must control, create and evolve APIs (West & Dedrick, 2000).
4. The platform must tap unserved niches before reaching critical mass (Bresnahan & Greenstein, 1999).
5. The successful platform increasingly attracts software developers as it grows (Katz & Shapiro, 1998).

The literature also suggests five features of software markets:

1. Software markets feature strong network effects, which tend to increase market concentration (Katz & Shapiro, 1998).
2. Strong network effects also cause the standards contest to tip to the technology leader (Besen & Farrell, 1994)..
3. A free market will support just one standard (Liebowitz & Margolis, 1999).
4. Software buyers base their choices on expected network sizes (Katz & Shapiro,

1998).

5. Positive returns to scale coupled with strong network effects lead to very high profits (Arthur, 1996; Katz & Shapiro, 1998).

Other key insights from the literature include the following:

- The Herfindahl-Hirschman Index (HHI) provides a possible measure of firm or platform market power.
- Pattern-matching is an analytic strategy for testing theory with case data (Yin, 2003). With respect to a predicted pattern, a case can be a literal replication, a theoretical replication, or an anomaly.
- Process study methods (Van de Ven, 2007) provide a methodical and structured approach for constructing a timeline of events.
- Good case research takes deliberate action at the research design stage, before data collection and analysis, to address known threats to validity and reliability of theory built from case research (Yin, 2003).

# *Chapter 3 Research Method*

This chapter describes and explains the method used to produce the deliverables. It begins with an explanation of the research design, followed by five sections each specifying one step of the five-step research method. A sixth section examines threats to validity and reliability and the tactics built into the research method to mitigate those threats.

This research employs a two-part design that includes both theory-testing and theory-building. The first part employs a *pattern-matching approach* (Yin, 2003, pp. 116-120) to examine the extent to which West's (2003) explanation of evolving platform strategies in the computer industry can account for the platform strategies of the three leading firms in the EDA industry. The second part employs an *explanation-building approach* (Yin, 2003, pp. 120-122) to develop explanations for anomalies not explained by West (2003). Both stages employ cases studies developed from publicly-available archival sources, including company shareholder reports and articles published in trade journals.

Table 1 summarizes the five steps of the research method and the actions undertaken at each step. Subsequent sections specify each step in detail.

*Table 1: Steps in the Research Method*

| Step | Action Undertaken to Produce Each Deliverable |
|---|---|
| 1. Conceptual Development | Develop a tabular summary of the West (2003) explanation of the emergence of hybrid platform strategies in the computer industry. Identify key constructs and the associations between constructs at a theoretical level. |
| 2. Develop measures, instruments and tests. | Apply West's explanation to the EDA industry context.<br>• Specify the constructs and associations in ways appropriate to the EDA industry.<br>• Identify a set of operational variables and measures (quantitative and qualitative) and the data sources required to assign attributes to those variables.<br>• Specify, prior to data collection, a predicted pattern of observations to be expected if West's explanation is valid in the EDA industry.<br>• Specify a table shell (Yin, 2003, p.75) of data to be collected. |
| 3. Data collection. | Collect the data needed to populate each cell in the table shell. |
| 4. Pattern-matching. | Compare the empirical observations to the predicted pattern. Assess the fit and note any anomalies. |
| 5. Explanation-building. | Develop an explanation of evolving platform strategies in the EDA industry:<br>• account for similarities and differences between actual and expected observations. |

## 3.1 Conceptual Development

### 3.1.1 West's Explanation

According to West (2003), platform strategies in the computer industry evolved in three distinct stages from proprietary, to open standards, to open source. In stage 1, vendors prefer proprietary platform strategies because control of the interfaces provides them with better barriers to imitation and better margins. In stage 2, some combination of technical and economic factors make proprietary platform strategies infeasible, and vendors respond by modifying their platform strategies to incorporate open standards shared with one more competitors. In stage 3, firms disclose some of their technology in an economic trade-off between the extremes of a totally proprietary strategy, which the market would reject, and a totally open strategy, which would eliminate all competitive advantage. The result is a hybrid strategy – either open parts (a mix of some commodity layers and some proprietary layers) or partly open (disclosure under restrictions intended to favour customers and place competitors at a disadvantage). The transition to open source is a continuation of the open systems strategy, driven by many of the same factors.

West identified four key motivations for firms to move from stage 1 to stage 2, and from stage 2 to stage 3: i) market share below minimum efficient scale to support proprietary R&D, ii) insufficient market power of the firm to resist buyer demand for openness, iii) the tipping of the contest to the open standard, and iv) the acceptance of commoditization of a particular architectural layer of the firm's platform. West also identified two implications for the firm transitioning between stage 1 and stage 2: i)

possible pitfalls of implementation, and ii) conflicts with corporate culture and core competencies. The transition from stage 2 to stage 3 has the additional implication that the basis of competition must change because implementations are visible to all.

Table 2 is a tabular summary of West's explanation of platform strategies in the computer industry.

*Table 2: West's Explanation*

| Stage | 1: Proprietary platforms | 2: Open standards | 3: Open sources |
|---|---|---|---|
| Description of stage | The firm delivers a complete proprietary platform solution. | The firm modifies its platform strategy to incorporate open standards that are shared with one or more competitors. | The firm disclose technology seeking to find the right compromise between totally proprietary and totally open. Hybrid strategies are *open parts* or *partly open*. |
| Motivation | Proprietary platforms are preferred because they provide better barriers to imitation and better margins, and there are more attractive profits for the firm. | Some combination of technical and economic factors make a proprietary platform infeasible:<br><br>A - Market share lower than minimum efficient scale to support R&D<br><br>B - Insufficient market power to resist buyer demand for open standards<br><br>C - Tipping of the contest in favour of the open standard<br><br>D - Acceptance of commodification of one architectural layer and shift of competitive advantage to another layer | The transition to open source is a continuation of that to open systems.<br><br>The transition is driven by many of the same factors.<br><br>Totally proprietary platforms would be rejected by the market and totally open platforms would eliminate all competitive advantage. |
| Implications | | May run contrary to corporate culture and previously valued core competencies.<br>Possible pitfalls. | Implementation details are visible to all. |

### 3.1.2 Theoretical Constructs Drawn from West's Explanation

West explanation has five key theoretical constructs that are defined in this subsection. Three constructs concern the architecture of the platform: *standard*, *platform* and *source*. A fourth construct relates the platform architecture to business and competition: *platform strategy* is the degree of *openness* of standard, platform, and source, and classification at a *stage* according the values of these attributes. A fifth construct is the *motivation* of managers at the firm to modify the platform strategy.

**Defining the Technology and Business Model Constructs**

A *standard* specifies the *interface* between modular layers of an architecture. It may take the form of a structural format for data representation, an applications programming interface (API), a programming language, or other general agreement which ensures interoperability between modules and or architectural layers in a software system or data network.

A *platform* is a layered architecture of related standards (West, 2003, Section 2.1). The EDA platforms examined in this thesis are layered architectures of related *software* standards.

*Source* refers to the *implementation* of a modular software layer within the platform. The source code of a module may implement standard interfaces. More than one implementation of the standard may be offered for users to select.

A *platform strategy* is the *degree of openness* of the standards and source at each

 

layer of a platform. It may be useful to classify platform strategies into categories or stages.

### Associating *Openness* to the Technical Constructs

A *closed standard* is controlled by a single firm; another firm must seek permission from the controlling form to employ it. Control of an *open standard* is shared with one or more of a firm's competitors (West, 2003, Section 9.12).

*Open source* is a software implementation that meets the criteria of the Open Source Definition (OSD), which sets ten criteria for the attribution of authorship and the integrity of any specific version with respect to modification or patches to the code, and requirements for distribution and licensing of the source code (Open Source Initiative, 2013). *Closed source* is any implementation that does meet the criteria of the OSD.

An *open platform* allows the substitution of a vendor's software module by that of another vendor, whereas a *closed platform* restricts such modular substitution of software modules – because the standards are not defined in a way to permit modular substitution.

### Associating *Stage* to the Platform Strategy Construct

A *stage* is a period of time in which the firm's platform strategy is distinct from the platform strategy at previous or subsequent stages. According to West, there are three distinct stages that transition in a one-way, simple progression from stages one to two to three. In the process theory nomenclature of Van de Ven (2007), West's explanation is driven by a *life cycle process*, like the progression through stages of life by an organism

(see section 2.4). In stage one, the vendor's platform is proprietary and implements closed standards. In stage two, at least one closed standard within the vendor's platform is replaced with an open standard. In stage three, at least one layer of the platform architecture is implemented in open source software with an implementation that is visible to all.

### *Motivations* to Modify the Platform Strategy

According to West (2003, section 9.12), the transition from one stage to the next is motivated by a combination of technical and economic factors that make the present stage infeasible. West identifies four factors*:*

1. *Market share* less than *minimum efficient scale* to support proprietary R & D.
2. *Market power* less than *buyer demand for openness*.
3. The *tipping* of the standards contest in favor of the open standard.
4. A *decision* to accept commoditization of an architectural layer of the platform and shift competitive advantage to another layer.

Technical concepts from the economics and strategy literature are explained below.

- *Market share* (MS) is the fraction of total market revenue captured by the firm. It is calculated by dividing the firm's revenues by the revenues of the total market. If the unit price of the software is very high, revenue market share is informative (Katz & Shapiro, 1998).

- *Minimum efficient scale* (MES) is normally defined to be reached when the long term average cost of production equals the margin cost of production. According to Katz & Shapiro (1998), MES for software vendors, MES is achieved when the firm or the industry exhibits long term average profitability.
- *Market power* (MP) is the power of the hypothetical monopolist to impose a "small but significant and non-transient increase in price" (SSNIP) above the pre-merger level (DOJ Horizontal Merger Guidelines).
- *Buyer Demand for Openness* (BDO) is the collective power of buyers in the market to reject a vendor's totally proprietary platform strategy (West, 2003, Section 9.1.3). This rejection could be motivated by the philosophical principles of the open source movement (West, 2003, Section 3) or other reasons. Software markets exhibit strong network affects which make this measure forward-looking (Katz & Shapiro, 1998).

The *tipping* of the standards contest in favor of the open standard (Bessen & Farrell, 1994) is the instability and self-reinforcing nature that is exhibited when strong network externalities combined with high switching costs between standards lead to increasing returns to scale that are magnified by an early lead (Arthur, 1996).

Table 3 shows the associations between the key constructs of West's explanation.

Table 3: Associations Between West's Key Constructs

| Construct Associations | | Description / Explanation |
|---|---|---|
| Platform Strategy | Stage | A distinct three stage evolution from proprietary to the use of open source. (West, 2003, Section 9.1) |
| | Stage 1<br><br>Proprietary and Closed Standards | Vendor establishes a successful proprietary and closed standard platform strategy. (West, 2003, Section 9.1.1) |
| | Stage 2<br><br>Proprietary implementations of Open Standards | The vendor's proprietary platform strategy is modified to incorporate an open standard that are shared with one or more competitors. (West, 2003, Section 9.1.2) |
| | Stage 3<br><br>Open Source | The vendor's proprietary and open standard platform strategy is again modified to incorporate an open source standards whose implementation is visible to all. (West, 2003, Section 9.1.3) |
| Motivations for transitions from stage to stage | Platform strategy becomes infeasible for some combination of technical and economic factors. | A: Market Share less then the Minimum Efficient scale (MES) to support proprietary R & D. |
| | | B: Market Power less than Buyer Demand for Openness. |
| | | C: Tipping of the standards contest in favor of the open standard. |
| | | D: A decision to accept commoditization and shift competitive advantage to another layer in the platform architecture. |

## 3.2 Develop Measures, Instruments and Test

West's explanation was induced from close study of three leading firms in the computer industry, but this thesis examines a different industry context and a different interval of time. This section develops an operationalized specification of West's explanation in a form appropriate to the context of this study. The outcome is an empirical pattern-matching test (Yin, 2003, pp. 116-120) to assess the explanatory power of West's explanation in the EDA industry context. The first subsection compares the computer industry and EDA industry, and introduces the EDA industry platforms, platform architectures, layers, and standards to be examined. The second subsection defines a set of operational variables and variable attributes, identifies the measures for each variable, and explains the data sources, collection procedures, and decision rules to assign attributes to each variable. The third subsection specifies, *a priori*, the pattern-matching test of West's explanation. The fourth subsection develops the *table shell* (Yin, 2003, p. 75) that structures data collection by focusing research activity on the specific variables and data sources required for the pattern-matching test.

### 3.2.1 Comparing the Computer and EDA Industry Contexts

This subsection compares the computer and EDA industry contexts, relating the definitions of standard, platform, and source. Table 4 compares the computer industry and EDA industry.

*Table 4: Comparison of the Computer Industry and EDA Industry*

<table>
<tr><th colspan="3">Computer Industry Standards</th><th colspan="3">EDA Industry Standards</th></tr>
<tr><td colspan="6">Programming Languages</td></tr>
<tr><td rowspan="2">Programming Language</td><td>Commercial</td><td>C</td><td rowspan="2">Hardware Description Language (HDL)</td><td>Commercial</td><td>Verilog</td></tr>
<tr><td>Military</td><td>Ada</td><td>Military</td><td>VHDL</td></tr>
<tr><td rowspan="2">Increased system complexity</td><td rowspan="2">Object-orientation</td><td rowspan="2">C++</td><td rowspan="2">Electronic System-level Language (ESL)</td><td rowspan="2">Systems on chips</td><td>SystemC</td></tr>
<tr><td>SystemVerilog</td></tr>
<tr><td colspan="6">Platform</td></tr>
<tr><td colspan="3" rowspan="4">Software</td><td rowspan="4">Software: CAD Tools</td><td>Design Entry</td></tr>
<tr><td>Simulation</td></tr>
<tr><td>Synthesis</td></tr>
<tr><td>Place & Route (Layout)</td></tr>
<tr><td colspan="3">Operating System APIs</td><td colspan="3">CAD Framework APIs</td></tr>
<tr><td colspan="3">Middle-ware (Java / Databases)</td><td colspan="3">CAD Design Database</td></tr>
<tr><td colspan="6">Source</td></tr>
<tr><td colspan="3">Software Module</td><td colspan="3">Software Module</td></tr>
</table>

**Standards**

The computer industry uses standard software programming languages. C was developed as a proprietary language, and is widely used commercially. Ada is a language developed for military software applications where reliability is a deep requirement. As software systems increased in complexity, object-orientation was introduced which allowed software libraries for specific system functionality to be shared and re-used and code to be developed hierarchically that exploited standard interfaces.

The EDA industry uses standard hardware description languages (HDLs). Verilog

has a C-like syntax and was widely introduced commercially as a proprietary language. The Very High Speed Integrated Circuit (VHSIC) military project spawned the VHDSIC hardware description language (VHDL), which is similar syntactically to Ada. Verilog was conceived around the same time as VHDL. VHDL was later mandated for military applications. Both Verilog and VHDL were standardized by the Institute of Electrical and Electronics Engineers (IEEE). As integrated circuit (IC) designs increased in complexity, and microprocessors could be embedded in an IC, Electronic System Level (ESL) languages were introduced to enable co-simulation of hardware and software.

**Platform**

Operating systems (OS) and software applications were integrated into platforms through standard applications programmers interfaces (APIs). Some operating systems were hardware-specific, and the platform and / or OS vendor's might control the software APIs released under license to software developers. Some OS's , like Unix and Linux, were developed to enable hardware abstraction, allowing the migration of the OS to other hardware architectures. Middle-ware, such as databases and the Java language, which used a virtual machine concept, were developed.

The EDA industry started with proprietary computer-aided design (CAD) platforms consisting of proprietary hardware, OS and software, and then developed into a mainly software industry with different software layers defining the platform. Proprietary databases were used to store design and stimulation data for simulation and circuit layout that represented deeper physical characteristics as the logic circuit was developed in a

progression through a chain of tool processes. Frameworks were developed to allow independently developed design tools to inter-operate.

The EDA platform is essentially a suite of software tools that can share data via direct exchange or via a central design database. The general tools work in a directed procedural “methodology”, also called a “design flow”, consisting of design entry, simulation, synthesis, and place-and-route (layout).

**Source**

Source code in the computer and EDA industry is compiled to a machine code. After compilation, programs are released as binary machine codes, which is not humanly readable. Source in the EDA industry can also refer to the HDL code written to describe a logic design, but HDL source code is usually called Intellectual Property (IP). Such IP could be licensed as behavioral, synthesizable, or structural logic. Behavioral HDL contains non-synthesizable constructs. Synthesizable HDL is written using a sub-set of the full HDL language specification. Structural HDL is an inter-connected list of wires and primitve logic elements called gates. Generally speaking, the EDA platform offered by vendors consists of design tools, and any HDL-based IP – or hardened IP (physically processed  modules like a memory structure) might be offered to customers separately from the platform.

### 3.2.2 Operational Variables and Measures

Table 5 shows the association between constructs, variables, variable attributes, how attributes are assigned, and the source of the data collected to assign attributes.

*Table 5: Constructs, Variables, Measures and Sources*

| Construct | | Variables | Measure Type / Value | | Source |
|---|---|---|---|---|---|
| Platform Attributes | Openness of platform to standards shared with competitors. | Standard | Design Entry | Closed (0) Open (1) | Electronics EDN |
| | | | Simulator | | Electronics EDN |
| | | | Synthesis | | Electronics EDN |
| | | | Layout | | Electronics EDN |
| | | | Framework | | Electronics EDN |
| | | | Database | | Electronics EDN |
| | Openness of platform layer(s) implementation (source code). | Source | Source code license | Open Source (2) Community Source (1) Commercial (0) | Electronics EDN |
| Motivations | A | MS | Compare: (MS < $MES_{R\&D}$) | FALSE (0) TRUE (1) | Annual Reports |
| | | $MES_{R\&D}$ | | | |
| | B | Market Power (MP) | Firm contribution to HHI | Low (0) Moderate (1) High (2) | Annual Reports |
| | | Buyer demand for openness (BDO) | Market Concentration | Low(2) Moderate (1) High (2) | Annual Reports |
| | Tipping of the contest in favour of the open standard | Number of Vendors & Market share | # of top 3 or 4 EDA vendors adopting open standard or open source | N < 2 (0) N < 2 (1) | Electronics EDN |

| Construct | | Variables | Measure Type / Value | | Source |
|---|---|---|---|---|---|
| | A decision to accept the commoditization of one layer of the architecture and shift competitive advantage to another layer. | Release of proprietary technology | Release of source code under open source license.<br><br>Joining an open source initiative. | Default (0)<br>Release (1)<br>Join (1) | Electronics EDN |

In the “constructs” column of Table 5, the two platform attributes are openness to standards shared with competitors, and openness of source code. The architectural layers (design entry, simulation, synthesis, layout, framework and database) are shown to be associated with the standard variable, but this association also applies to the source code variable. To simplify the presentation of Table 5, the second set of associations of the layers to the source code variable are not shown.

The platform has these software layers as measures: design entry, simulation, synthesis, layout, plus the database and design framework as middle-ware layers. The openness of the vendor's platform layers to standards and source will be measured as follows:

- **Design Entry**. The design entry layer presents a closed (open) standard if a design file exported from vendor A's platform cannot (can) be directly imported into vendor B's platform without intermediate translation.

- **Simulation**. The simulation layer is closed (open) if the design's logic and the external stimuli (called a test) to exercise the logic (implemented using an HDL or otherwise) exported from Vendor A's platform cannot (can) be directly imported into vendor B's platform without intermediate translation.
- **Synthesis**. The synthesis layer is closed (open) if Vendor A's export (pre-synthesized HDL or post-synthesized gate-level netlist) cannot (can) be directly imported into vendor B's platform without intermediate translation.
- **Layout**. The layout layer is closed (open) if the physical placement and wiring of the design cannot (can) be directly imported into vendor B's platform without intermediate translation.
- **Database**. The database layer is closed (open) if the database records cannot (can) be directly imported into vendor B's platform without intermediate translation.
- **Framework**. The framework layer is closed (open) if users cannot (can) select third-party modular tools that plug into the framework.
- **Source Code**. A platform layer is an open source implementation if the vendor releases or adopts source code, and/or joins an open source initiative, where the source code license for that layer is released under an open source license. Otherwise, the source code for the platform layer is closed, even if it is released under a community source license, because de facto source code control remains with the vendor.

The “sources” column of Table 5 identifies three sources for the data required to assign attributes to variables. *Electronics* and *Electronics Design News* (EDN) are industry trade journals archived by the website www.business.highbeam.com. “Annual reports” refers to the shareholder reports and other publicly-available financial documents that are legally-mandated for publicly-held companies. Note that Table 5 focuses exclusively on the variables required for the pattern-matching test. This research also draws on other sources, described later, to examine the broader context of these variables (see, for example, subsection 3.3.2).

West's four firm motivations to modify a platform strategy are the following:

1. Market Share (MS) that is less than the Minimum Efficient Scale to support R & D ($MES_{R\&D}$)
2. Market Power (MP) is insufficient to resist Buyer Demand for Openness (BDO)
3. Tipping of the (standards) contest in favour of the open standard.
4. A decision to accept commoditization of an architectural layer and shift competitive advantage to another layer.

The first two motivations are specified as composite variables computed from primitive elements of market share, market power, minimum efficient scale, and buyer demand (for openness.

The EDA software industry operates with high margins and very high unit price.

*Market share* can be therefore be reasonably inferred from a comparison of the firm's revenues to the whole industry's. As a result, revenue market share will be used in general. Where specific data is available on unit sales, this data is used to supplement the revenue data collected from Electronics and EDN archives, but also from the companies' annual reports.

*Market power* will be estimated by considering the yearly change in the HHI (see section 2.2). If an important merger or acquisition is observed, and the revenue of the acquired firm is known, this will be used to estimate whether or not the change in HHI due to the acquisition had a likely increase in market power using the DOJ's standard. The market share will be considered in the context of what assets the firm brings to future competition.

*Minimum efficient scale*, following Katz & Shapiro (1998), will assume that the long term profitability of the firm indicates the firm's platform has achieved minimum efficient scale. This is based on the use of average cost instead of marginal cost, as the software industry always has marginal cost below average cost.

*Buyer demand for openness* will be based on the relationship between the market shares for competing platforms and their degrees of openness considering all the platform architectural layers. This is consistent with West's approach, where the dramatic fall of Apple's market share is cited as a reason for it to turn to open peripheral standards (West, 2003, Section 9.1.2). Archival data may be used to show that the debate over platform or

architectural layer openness has occurred in the media by statements made by industry analysts or vendor representatives.

To measure *market share less than minimum efficient scale* to support proprietary R & D, let A be the comparison function that evaluates to TRUE when market share is less than the minimum efficient scale to support proprietary R & D. Katz and Shapiro (1998) argue that market share is informative of market power, provided it doesn't switch markedly over time. So, a large firm that is a sponsor of a proprietary platform will need to make a clear transition from showing profit to experiencing a sustained loss. The data on the profits, revenues and market shares of public companies are trailing indicators, meaning internally the firm may have had information not available to the market when decisions were made. The HHI can measure **buyer demand for openness** by assigning the HHI contribution of competing firms to open or closed categories to produce two HHIs: the HHI for closed and for open platforms. If the HHI for closed (open) platforms is significantly higher than the HHI for open (closed) platforms, then buyer demand for openness is low (high). If the market is moderate to highly concentrated, then a few firms have significant market share and a large number of smaller firms have very little market share. In this case, buyer demand for openness is very high if even one leading firm's platform has a high degree of openness.

*Tipping* is related to network effects, which is a function of relative adoption of standards. If tipping occurs, the winner is rewarded positive returns to scale. The revenue

market shares of the competing firms combined with qualifying information on platform openness will be used to make this assessment. Specifically, since EDA software commands high margins, tipping should be indicated by a significant increase in adoption that leads to large profit gains for the open platform standard.

Commoditization of an architectural layer will be considered to have occurred when that particular software, or a market substitute, has been released as open source. The *decision to accept commoditization* will be either that event, or the response to that event by a particular vendor to follow suit.

### 3.2.3 Pattern-matching test

Robust profits and growth of market share are characteristics of a successful proprietary platform strategy. Market share and profit margins must be impacted before the firm is motivated to modify its platform strategy to incorporate open standards that are shared with its competitors. The open source stage is a continuation of the open standards movement, and is driven by many of the same technical and economic factors. The pattern implied by West's explanation, if it is operational in the EDA industry, requires each firm to evolve its platform strategy in three stages from proprietary and closed standards to open source, as indicated by Table 6.

*Table 6: Pattern To Test*

| Construct | Composite Variable | Stage 1 | Stage 2 | Stage 3 |
|---|---|---|---|---|
| Platform Attributes | "Openness to platform standards that are shared with competitors.<br><br>0 = Proprietary (not shared)<br>1 = Open (shared with competitors) | 0 | 1 | 1 |
| | "Openness" of platform implementation (source code).<br><br>0 = Not open source license<br>1 = Open source license | 0 | 0 | 1 |
| Motivations | 0 = Market Share >= minimum efficient scale to support R & D<br>1 = Market Share < minimum efficient scale to support R & D | 0 | 1 | 1 |
| | 0 = Market power >= buyer demand for open standards.<br>0 = Market power < buyer demand for open standards. | 0 | 1 | 1 |
| | 0 = No open standard dominates<br>1 = Contests tips to the open standard | 0 | 1 | 1 |
| | 0 = No decision to accept the commoditization of one layer of the architecture and shift competitive advantage to another layer.<br><br>1 = A decision to accept the commoditization of one layer of the architecture and shift competitive advantage to another layer. | 0 | 0 | 1 |

In the above table, the four motivations are given expected values.

### 3.2.4 Table Shell for Data Collection

Table 7 is the "table shell" (Yin, 2003) guiding data collection.

*Table 7: Table Shell for Data Collection*

| Construct | Variable (attributes/units) | | Year | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Company Name | | | 1980 | | | 1990 | | | | | | | | | | 2000 | | |
| | | | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| Platform | Standard<br><br>6 platform layers<br>(0: Close;<br>1: Open) | Design Entry | | | | | | | | | | | | | | | | |
| | | Simulation | | | | | | | | | | | | | | | | |
| | | Synthesis | | | | | | | | | | | | | | | | |
| | | Layout | | | | | | | | | | | | | | | | |
| | | Database | | | | | | | | | | | | | | | | |
| | | Framework | | | | | | | | | | | | | | | | |
| | Source<br><br>6 platform layers<br>(0: Closed license;<br>1: Community source license;<br>2: Open source license) | Design Entry | | | | | | | | | | | | | | | | |
| | | Simulation | | | | | | | | | | | | | | | | |
| | | Synthesis | | | | | | | | | | | | | | | | |
| | | Layout | | | | | | | | | | | | | | | | |
| | | Database | | | | | | | | | | | | | | | | |
| | | Framework | | | | | | | | | | | | | | | | |
| Motivations | MS < MES | Market Share (%) | | | | | | | | | | | | | | | | |
| | | Long Term Ave. Profits (US$) | | | | | | | | | | | | | | | | |
| | MP < BDO | Market Share2 | | | | | | | | | | | | | | | | |
| | | HHI Open (Std \| Src) | | | | | | | | | | | | | | | | |
| | Tipping | # Top 3/4 Open Vendors | | | | | | | | | | | | | | | | |
| | | # Top 3/4 Closed Vendors | | | | | | | | | | | | | | | | |
| | Decision to Accept Commoditization | Released Open Source | | | | | | | | | | | | | | | | |
| | | Joined Open Source Initiative | | | | | | | | | | | | | | | | |

## 3.3 Data Collection

### 3.3.1 Obtaining and Coding Raw Data

The data required to populate the table shell (Table 7) were obtained from articles in EDA industry trade journals and company reports. Trade journal articles report on and analyze the features and performance of standards-based EDA design tools of the day, and whether these tools were developed “in-house”, or were brought into the company's control via a merger or acquisition. The archives are available online (at http://business.highbeam.com and http://cbronline). Each archive site provides search engine capability to specify key words and dates to delimit the search. The following five steps were taken to gather raw data for storage in a database to be post-processed.

1. Enter key words that identify the company or companies, platform layer, such as simulation,”place & route”, logic synthesis, etc.
2. Limit the search by specifying a date range, or order the articles chronologically with the oldest articles shown first.
3. If the returned article(s) appear relevant by title, open the article and use the web browser's “find” feature (usually CTRL-F) to specify a text string to find in the article one or more key words that the search engine hit in its archival search.
4. If the information is deemed relevant (by identifying the company, its featured product, its capability and use of proprietary or open standards, and/or its competitive position against other EDA vendors), copy the relevant text from the

article and add it to the database of archived articles found – indexed by URL, article title and platform layer feature.

5. Continue with the article search process until there is sufficient raw data (after the post-processing steps identified in the next sections) to populate the table shell for data collection for each EDA vendor.

### 3.3.2 Adding Time and Industry Context to the Raw Data

This subsection describes how the raw data was used to develop a time-line of EDA industry and vendor milestones. The timeline (reported in section 4.1) has four columns: the first EDA industry in general, and each of the EDA vendors studied.

An EDA industry event is presented as a concise description and dating of when new technology was introduced by an EDA vendor, the EDA industry accepted a new open standard, or a third-party organization was formed to promote an open standard or open source initiative. Presented in the context of other such events in a time-line, an EDA industry or vendor event is called a **milestone**. Multiple sources of evidence enable the association of each milestone to its historical context with respect to the EDA industry and the activity of each vendor's closest rivals. The historical context enables the identification of potential cause and effect between the activities of one vendor and the industry or one vendor and its rivals. The rules for creating this table are as follows:

1. Develop a concise summary of an event, which may be described by multiple entries in the raw article database, with respect to the specific architectural layer

of the EDA platform the introduction of a new standard (industry) or technology (vendor's product)

2. Enter the event in the milestone table, assigning it to either the EDA industry in general (because it is vendor non-specific, or vendor specific, but not a vendor studied by this thesis), or to a specific vendor. Earlier event dates are presented earlier in the table, resolving down to the month, if necessary.

### 3.3.3 Use the Milestones to Develop a Sense-making Narrative

The table of EDA industry milestones is then used as the source material to develop a narrative providing context for milestones. The narrative is intended to serve three functions:

1. Make sense of the data by making associations between milestones. This places the EDA industry, as well as each vendor, in the context of the technological and industry imperatives that drove the EDA industry forward.

2. Identify whether or not there are gaps in the data that make it difficult to make sense, because an event is not linkable to other salient events, or because the event is not relevant and can be excluded from the table without loss of meaning or continuity.

3. Identify whether or not the event signals a strategic modification to the vendor's platform, or is simply tactical. A tactical platform modification would be reversible, if, for example, participation in a failed effort to create an open

standard. On the other hand, a strategic modification is irreversible, such as when key platform source code is released under an open source license.

### 3.3.4 Procedure to Populate the Table Shell Platform Attributes

The cells of the table shell for each case company (Table 7 in section 3.2.4) are populated in multiple steps, beginning with the platform attributes. The completed tables are reported in Chapter 4 (Table 10, Table 11, and Table 12).

Attributes for the platform variables were assigned according to the following four-step procedure:

1. If the specific platform standard architectural layer does not implement an open standard as its core feature or in a key interface between it and other modules of the platform, assign the value 0 to the cell for that year. If it does, assign a 1 to the particular cell.
2. Once a 1 has been assigned to a cell, assign a 1 to the cell to the immediate right unless evidence shows that the layer no longer implements an open standard, in which case assign a 0 to that cell.
3. If the specific platform source architectural layer does not feature an open source implementation at its core or in a key interface, assign the value 0 to the cell for that year. If it does, assign a 1 to the particular cell.
4. Once a 1 has been applied to a cell, assign a 1 to the cell to the immediate right unless evidence shows that the layer no longer has an open source

implementation, in which case assign a 0 to that cell.

### 3.3.5 Procedure to Populate the Table Cell Motivations

All three case companies are publicly-held corporations, which must file quarterly and annual reports of their financial results, especially revenues, profits (losses). In 1992, major firms in the EDA industry established the Electronic Design Automation Consortium (EDAC) to help the EDA industry better understand its market. EDAC began publishing industry revenues in 1995. Dataquest, an industry statistics reporting firm, published EDA industry revenues for the years 1987 to 1994.

Using these sources of information, the financial components of the table shell are assigned as follows:

1. The market share of each firm is computed by dividing the firm's revenues by the EDA industry's revenues.
2. The profits (or losses) of the firm are reported annually.
3. The firm's contribution to the HHI is the square of its market share (in percent).
4. The HHI of all EDA vendors is computed assuming that the typically 30% to 40% of the EDA industry revenue that is not shared by the top three or four companies is shared evenly by 45 companies. This is valid because the HHI of one such company would then be less than 1, and so the total HHI of the smaller players is less than 45, whereas the HHI of a company with just 10% market share is 100.

5. The HHI of all EDA vendors offering an open standard solution is assumed to be the HHI of the industry once the top three EDA vendors incorporate open standards into their platforms.

Tipping is assessed by comparing the number of the top three EDA firms offering an open standard platform compared to those offering a proprietary platform. If a firm releases source code under an Open Source license, it is assigned a 1, otherwise a 0 in that row. If the firm joins an Open Source initiative, it is assigned a 1, otherwise a 0 in that row.

### 3.3.6 Assignment of Stages

An additional table provides a summary of platform attributes and motivations for all three case firms, and the assignment of a "stage" to each year (if applicable), according to the decision rules previously specified in Table 6 (see subsection 3.2.3). Only the composite variables are reported, permitting a side-by-side comparison of the platform strategies of the three case firms over time. For each of the "standards" and "source" composite variables, a value of 0 is assigned to the platform only if a value of 0 has been assigned at each architectural layer, and a value of 1 is assigned to the platform if a value of 1 has been assigned to at least one layer of the platform; the assignment of 0 or 1 is complete and mutually exclusive. Those results are reported in Chapter 4 (Table 13).

Following the decision rules in Table 6 and subsection 3.2.3, a company is said to

operate at Stage 1 if the platform strategy is proprietary (standards = 0; source = 0), at stage 1 if the platform strategy is “open standards” (standards = 1; source = 0), and at stage 3 if the platform strategy is “open source” (standards =1; source = 1). A value of “n/a” is assigned if none of the conditions are met.

### 3.4 Pattern-Matching

The pattern-matching test compares the observations from each case to the expected pattern of Table 6 predicted by West's explanation. This has two aspects:

(1) transition of platform strategy from stage 1, to stage 2, to stage 3, and

(2) some mix of technical and economic factors that motivate transitions because the current platform strategy is no longer feasible.

A vendor must be successful with a proprietary and closed EDA platform to be categorized as operating in stage 1.

To transition from stage 1 to stage 2, a vendor must later be observed to modify its platform architecture to support an open standard that applies to at least one architectural layer. At least one of the first three of West's four motivations must be observed in advance of the observed platform modification to infer causation.

To transition from stage 2 to stage 3, a vendor must later be observed to modify its platform strategy to implement open source code in at least one architectural layer, either by publicly releasing open source code for an architectural layer of its platform, or by joining an open source project, such that this action is consistent with West's fourth

 

motivation, a decision to accept commoditization of that architectural layer and shift competitive advantage to another layer. According to Table 2, this transition is a continuation of the previous transition and driven by the same factors, thus at least one other motivation must be observed in advance of the platform modification to infer causation.

If observations support the progression of the firm's platform strategy from stage 1 to stage 2 to stage 3, with the motivating factors observed prior to each transition, then observations are said to match the expected pattern. Otherwise, observations are said to depart from the expected pattern.

### 3.5 Explanation-Building

The fifth step of the research method employs the analytic strategy of explanation-building (Yin, 2003, pp. 122) to account for differences between the case study observations and the expected pattern. The objective is to modify West's explanation – by adding, removing, or modifying aspects of Table 2 – to produce an explanation that better explains the results. This can take the form of a *refinement* – a relatively minor change that adds some new subtlety or clarification – or an *extension* – a new category or theoretical consideration required for the EDA industry that was needed in the computer industry.

## 3.6 Validity and Reliability

Yin (2003, p.34) identifies a number of case study tactics to address possible threats the validity and reliability of case research. Section 2.3 previously reviewed the literature on construct validity, internal validity, external validity, and reliability. Table 8 summarizes the tactics employed within this research design to address each area. In the explanations provided in Table 8, the "steps" refer to the five steps of the research method (Table 1) specified previously in this chapter: (1) conceptual development (section 3.1), (2) develop measures instruments and tests (section 3.2), (3) data collection (section 3.3), (4) pattern-matching (section 3.4), and (5) explanation-building (section 3.5).

*Table 8: Tactics for Validity and Reliability (Adapted from Yin, 2003a, p. 34)*

| Test | Tactic | Action taken in this research design |
|---|---|---|
| **Construct Validity:**<br><br>correct operational measures for concepts being studied | Establish chain of evidence | The case study database (section 3.3; also see below) explicitly links each observation to the source for that observation. |
| | Use multiple sources of evidence | Data collected from multiple sources (including trade journals, company reports, industry reports, and online sources) and multiple types (quantitative and qualitative) provides *triangulation* (Jick, 1979). |
| **Internal Validity:**<br><br>correct causes and relationships | Do pattern-matching | An *a priori* theory, in the form of an expected pattern, is specified in advance of data collection. |
| | Do explanation-building | A narrative explanation (subsection 3.3.3 and section 4.2) is developed iteratively throughout data collection (step 3) and pattern-matching analysis (step 4). Explanation-building (Yin, 2003) is the main analytic strategy to account for differences between observations and expectations (step 5). |
| | Address rival explanations | Rival explanations are actively pursued during explanation-building (step 5), and documented in the explanation of results (Chapter 5) and discussion of results (Chapter 6). |
| **External Validity:**<br><br>findings can be generalized in a specific domain | Use replication logic in multiple case studies | Multiple cases (platform strategies of three leading EDA vendors) developed using the same case study protocol (step 2) permits *replication logic* (Yin, 2003; Eisenhardt, 1989) – both between the cases developed here and with the cases of West (2003) in the computer industry. Both literal replication and theoretical replication are possible (section 2.3.2). |
| **Reliability:**<br><br>results can be repeated | Develop case study database | All data, both quantitative and qualitative (e.g., salient excerpts of articles) are compiled together and retained in a case study database (section 3.3), with explicit links to sources. |

# *Chapter 4 Results*

This chapter presents the results on the platform strategies of three leading firms in the EDA industry: Mentor Graphics, Cadence Design Systems, and Synopsys. It is the outcome of step 4 of the five-step research method previously specified in Chapter 3 (Table 1 on p. 25 and section 3.4). It is organized as five sections, closely following the explanation of the research method in Chapter 3. Section 4.1 develops a time line of EDA industry milestones. Section 4.2 is a narrative that provides background and context on events in the EDA industry and at the three case firms, drawing extensively on archival sources. Section 4.3 presents tables of time series data on the variables of interest for each of the three case firms; this includes the characteristics of the platform and measures of the economic and technical factors of the West (2003) explanation. It includes worked examples to illustrate how values were assigned to variables in the data collection framework. Section 4.4 explains how values were assigned to the composite variables of the pattern-matching test. Section 4.5 presents the results of the pattern-matching test.

## 4.1 EDA Industry Milestones

Table 9 presents concisely the important milestones in the EDA industry that are relevant to the determination of what stage each of the three EDA vendors was operating in at any particular time from 1987 to 2002.

*Table 9: EDA Industry Platform Milestones*

| Date | Industry | Mentor | Cadence | Synopsys |
|---|---|---|---|---|
| 1981 | | Mentor founded | | |
| 4/1983 | ECAD introduces Dracula. | | | |
| 1984 | Phil Moorby invents Verilog.at Gateway DA. | Goes public.<br>Integrated suite of<br>Schematic Capture<br>Simulation<br>Cadisys IC layout | | |
| 1/1985 | DoD proposes VHDL at IEEE conference.<br><br>SDA develops modular CAD Design Framework. | | | |
| 11/1985 | EDIF: intra-vendor data transfer is impossible | | | |
| 1986 | DoD donates VHDL language to IEEE | | | Starts-up logic synthesis market. |
| 1987 | VHDL becomes IEEE Std.<br><br>ECAD Inc goes public. | Introduces QuickSim digital simulator. | | |
| 5/1987 | SDA Design Framework to Toshiba and SGS. | | . | Introduce Design Compiler (synthesis) |
| 12/1987 | Tangent Systems IC layout software. | | | |
| 5/1988 | | | ECAD + SDA = Cadence | |
| 6/1988 | Trimeter Synthesis<br><br>CAD Framework Initiative (CFI) formed at DAC | | | |
| 9/1988 | MILSTD-454 -> ICs documented in VHDL | | | |

| Date | Industry | Mentor | Cadence | Synopsys |
|---|---|---|---|---|
| 1/1989 | Gateway Verilog has 14% of simulator market. | Verilog grabs significant simulator revenues. | | |
| 2/1989 | | VHDL: central to account control strategy. | | |
| 3/1989 | | | Acquires Tangent: std LEF & DEF (layout) Leading TanCell and TanGate layout tools | |
| 4/1989 | | System-1076 VHDL | | |
| 6/1989 | Verilog has 500+ commercial licenses<br><br>Verilog easily fits into Design Framework.<br><br>CFI recommends standardization on a third-party framework. | | | |
| 8/1989 | | | Merges with Gateway | |
| 9/1989 | | Acquires Trimeter Technologies (Synthesis) | | |
| 1989 | Motorola uses EDIF as framework back-plane. | | | |
| 1989 | Silicon Compiler: Foundation framework | | | |
| 1/1990 | | Acquires Silicon Compiler | | |
| 2/1990 | | Start port tools to Sun workstations | | |
| 4/1990 | Valid Logic: RapidSIM VHDL simulator | Quicksim II (VHDL) | | |
| 5/1990 | | | Opens Verilog language | |

| Date | Industry | Mentor | Cadence | Synopsys |
|---|---|---|---|---|
| 6/1990 | | | Design Framework II integrates open standards | |
| 9/1990 | | | | Synopsys acquires VHDL simulation from Zycad |
| 10/1990 | | | Acquires Valid Logic for $198M | |
| 1990 | | | 3000+ seats for Verilog | |
| 1991 | | | 3000 more Verilog seats | |
| 1991 | | | Logic synthesis tool for Verilog, later for VHDL. | |
| 1991 | | “We got... lazy having... a closed system" | | |
| 1992 | | | | Goes public. |
| 1992 | Chronologic: Verilog Compiled Simulator | | | |
| 1992 | VHDL slow adoption for lacking. features | | | |
| 1993 | | | Cadence acquires Seed Solution's Leapfrog VHDL simulator. | Synopsys sues Cadence and Seed Solutions' founders over VHDL trade secrets related to Cadence's acquisition of the LeapFrog simulator. |
| 1994 | | Acquires MTI for VHDL V-System | | |
| 1994 | Viewlogic acquires Chronologic (VCS) | | | |
| 1994 | Verilog gets IEEE working group | | | |
| 1994 | Simulator sales: Verilog: $66.8M VHDL: $59.2M ASP Verilog: $9000 | | | |

| Date | Industry | Mentor | Cadence | Synopsys |
|---|---|---|---|---|
| 1995 | Verilog becomes IEEE Standard | | | |
| 1995 | | | Files suit against Avant! for theft of IP. | |
| | Simulator sales:<br>Verilog: $73.3M<br>VHDL: $52.5M<br>ASP Verilog: $17,500 | | | |
| 6/1996 | Synthesis standard without Synopsys? | | Cadence has only 5% share in logic synthesis | |
| 1/1997 | | | | Acquires EPIC for deep sub-micron IC verification |
| 1997 | | QuickHDL + V-System = ModelSim | | |
| 1997 | | | | Acquires Viewlogic (VCS) |
| 1998 | | | Acquires Ambit (synthesis) | |
| 1999 | | | | Forms Open SystemC Initiative (OSCI) |
| 4/1999 | | | Stock drops 40% CEO replaced. | |
| 5/1999 | | | Genesis Database | |
| 8/1999 | CynApps releases source vs SystemC | | | |
| 1999 | | Welcomes CynLib | Welcomes CynLib | |
| 3/2000 | | | SystemC license “so onerous no one ... would sign it” | |
| 4/2000 | | | | SystemC released under “community source” license. |
| 6/2000 | Silicon Integration Initiative (Si2) starts open source methodology (EDA.org) | | Opens LEF and DEF physical design formats to Si2 at EDA.org. | |

| Date | Industry | Mentor | Cadence | Synopsys |
|---|---|---|---|---|
| | | | Releases TestBuilder C++ library as open source | |
| 4/2001 | | | OpenVera is not true open source. | Open Vera LRM released |
| 6/2001 | OSCI chair resigns: Synopsys “charade” | “..concerns about openness of SystemC.” | OpenAcess API to Genesis database released with “Community Source” license | |
| 6/2001 | | "All that stood between us have gone away." | | Releases control of OSCI |
| 2/2002 | | | | Acquires Avant. |
| 12/2002 | | | Cadence announced release of open source Genesis (OpenAccess) code to Si2. | |
| 1/2003 | | | Cadence releases OpenAccess source code. | |

## 4.2 Background and context

This section develops a brief narrative to provide context for the time line of the previous section and the tabular data displays of following sections. It draws extensively on archival sources.

### 4.2.1 The Dominant Proprietary EDA Platform of Mentor Graphics

The EDA industry, which is now primarily a software industry, was pioneered in the early 1980s by three companies, Daisy Systems, Mentor Graphics and Valid Logic. Mentor Graphics, started in 1981, defined the proprietary EDA platform strategy as a purely software offer called the IDEA series starting in the early 1980's (Brooks & Giesen, 1985). Mentor's tools only ran on Apollo workstations. Its chief competitors, Daisy Systems and Valid Logic, developed their own workstations to work with their software (Runyon, 1984). By 1985, Mentor lead the EDA platform market with a closed EDA platform.

Typically, EDA vendors offered a platform whereby design tools were “bundled” (Gold, 1984) and linked together by a proprietary database and data exchange format. Mentor and its competitors, Daisy Systems and Valid Logic, maintained proprietary data formats that made it difficult for logic designs to be transferred to other vendor's platforms. Translators needed to be developed for this purpose to enable data from another vendor to be imported (Clarke, 1989). Users wanted an open Electronic Data Interchange Format (EDIF), but the leading EDA vendors purposely implemented low

quality EDIF writers to make switching to another vendor's platform difficult. (Schindler, 1987).

Mentor's (and its main competitors) customers designed logic using a graphical method called schematic capture. Once captured, the logic circuit was stored in the EDA vendor's database in a proprietary format. Logic simulation is a key step in the verification of the correct implementation of an IC design. Mentor introduced QuickSim, a proprietary logic simulator (Electronic Design, 1987). If a customer needed data from a Mentor database, because possibly Daisy had a better tool of some kind, that data needed to be translated from Mentor's to Daisy's format. Customers didn't like this, and demanded an open standard data format that was called the electronic Design Interchange Format (EDIF), but all three purposely implemented EDIF writers very poorly (Orr, 1985) to restrict their customers from exporting their data.

Mentor Graphics and its chief rivals (Daisy Systems and Valid Logic), the leading EDA vendors in the United States, were focused on military markets (Hooper, 1986). The Very High Speed Integrated Circuits (VHSIC) project (VHSIC Program Office, 1990) recognized that military IC technology lagged by about 10 years that of the commercial IC market. A consortium of American semiconductor companies was formed in 1986 called Sematech (Sematech, 2013) in order to regain technical superiority in semiconductors, which it lost to the Japanese. In the early 1980's, the U.S. Military, as an offshoot of its VHSIC program, contracted IBM, Intermetrics and Texas Instruments to

develop a hardware description language (HDL) in order to document the logic of an IC design, called the VHSIC hardware description language (VHSIC HDL), known more simply as VHDL (Rawles, 1987).

While the large EDA companies had developed platform strategies based on the bundling of proprietary tools, and were focused on mainly US military projects, a number of smaller innovators introduced technologies that appealed to the leading Japanese firms serving commercial markets.

In 1987, SDA Systems' Design Framework successfully penetrated the Japanese market. The most enthusiastic customers were not Americans, but Japanese. “They buy it like crazy,” SDA's founder said (Coy, 1987). The new Design Framework allowed SDA's customers to define a custom EDA software platform by the modular addition of their choice of point tools that worked with a unified design database (Computer Business Review, 1987c). Companies like Toshiba (Cole, 1989) used the SDA framework to assemble a complete system for designing a digital IC based on de facto industry standards. The tools the customers liked the most were those offered by four other companies: Gateway, ECAD, Synopsys, and Tangent Systems.

Verilog was a proprietary Hardware Description Language (HDL) that was invented in 1983 by Phil Moorby of Gateway Design Automation. (Goering, 2005). Digital logic could be described in a text format (.v) using Gateway's Verilog language for simulation with its proprietary simulator called Verilog-XL. Verilog was developed as

a language and simulator to augment a logic synthesis product (Goering, 2005). Commercial requirements, especially fast logic simulation, drove the developers of the emerging de facto Verilog language standard to describe (VHDL's original purpose), simulate and verify the circuit timing of completed integrated circuit (IC) designs.

ECAD offered a leading physical layout design rules checker (DRC) called Dracula (Chen, Gau, & Liao 1986). It had also acquired a competitive product called MASKAP from a bankrupt firm (Albany Times Union, 1986). ECAD Inc went public that year (Richards, 1990).

Also in 1986, spun-off from General Electric, a new company called Optimal Systems formed to commercialize a new technology called logic synthesis, which could produce a logic design automatically from a text-based hardware description (Synopsys, 2013). A year later, Optical Systems (renamed to Synopsys) introduced its Design Compiler synthesis tool, which would become the de facto standard for logic synthesis (Synopsys, 2013).

Tangent Systems has developed IC layout software that could be used in gate arrays (TanGate) and standard cell-based designs (TanCell). The software offered powerful place and route algorithms that efficiently wired circuits on several layers of metal (Computer Business Review, 1987g).

Figure 3 illustrates the Mentor platform of bundled EDA tools shown in a side-by-side comparison with the open systems approach being prototyped by companies like

 

Toshiba which were rapidly adopting SDA System's Design Framework and the leading "point tools" of the time.

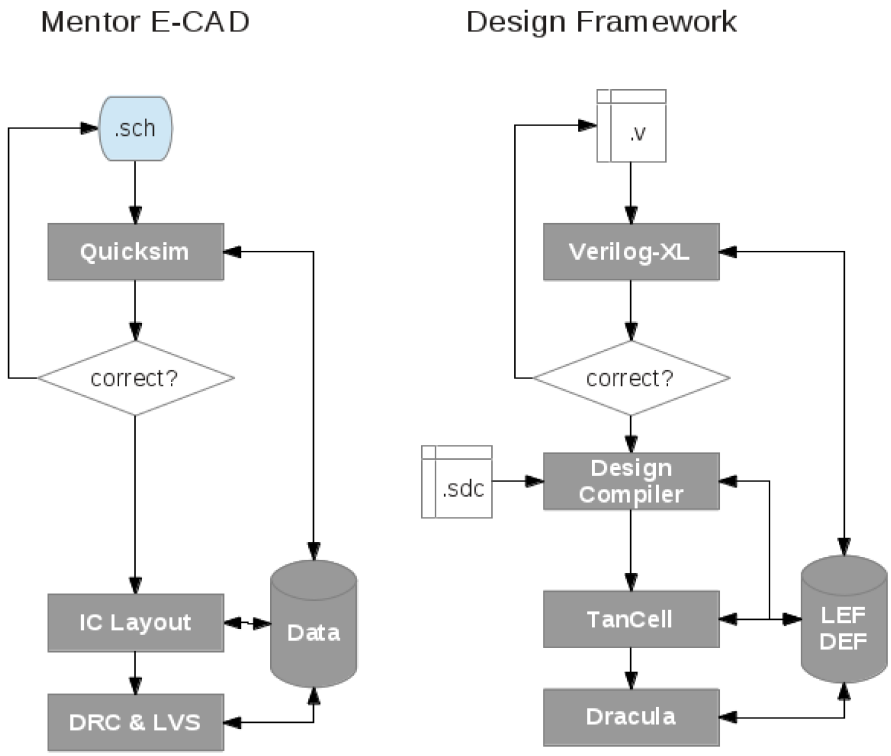


*Figure 3: Mentor's Closed Platform Versus the "Open" Design Framework*

The open systems IC design flow, enabled by the Design Framework, operated as follows:

- A collection of Verilog HDL text files (ending with the suffix: .v) defined a logic design from the "top down". A collection of hierarchical Verilog text files could

be simulated with Gateway's Verilog-XL simulator.

- After a correct simulation, the same set of Verilog files could be read in to Synopsys' Design Compiler and synthesized into a gate-level “netlist” as output from the tool. The netlist was generated based on a specific ASIC vendor's gate-array or standard-cell library (specified in the LEF format) of logic gates using Synopsys' Design Compiler, controlled by a script read in from another text file (.sdc), or sets of files.
- The Tangent Systems TanCell (for standard cell-based IC design) or TanGate (for gate-array based IC design) IC layout tools was used to perform “timing-driven” place and route of the input netlist, written out as a DEF file.
- The completed DEF layout was then verified using the ECAD's Design Rule Checking (DRC) technology called Dracula.

The Mentor “design flow” started with a graphical method of schematic capture of the IC logic gates by a design engineer. The output of the schematic tool, a gate-level netlist in Mentor's proprietary format, could then be simulated and verified for correctness by Mentor's QuickSim. The logic gates then passed to Mentor's IC layout tools. The completed layout was then verified by Mentor's DRC tools then checked using a Layout Versus Schematic (LVS) tool.

In 1988, SDA and ECAD merged and renamed itself Cadence Design Systems (McLeod, 1989c). Following this merger, Cadence then acquired Tangent Systems in

April, 1989 (Computer Business Review, 1989b). Cadence now was in control of the industry's de facto standard for the “back-end” of IC design, the physical layout and verification of IC designs. For the “front-end”, Cadence then merged with Gateway Design Automation in October, 1989 (Computer Business Review, 1989f). Cadence was now in control of four out of the five key technologies comprising an open systems EDA platform, which posed a real threat to Mentor Graphics.

Prior to the merger with Cadence, Gateway Design Automation had found a way to infiltrate Mentor's platform to enable Mentor's customers to use Gateway's Verilog-XL simulation technology, which had captured 14% of the simulation market (McLeod, 1989d). As of February, 1989, one third of Mentor's revenues were due to simulation technologies. That month, Mentor responded to this infiltration by the emerging de facto standard Verilog HDL by modifying its “account control” (platform) strategy to be based on the open standard VHDL. In April 1989, Mentor announced its System-1076 technology for VHDL-based simulation, which the company had internally referred to as the “Verilog killer” (McLeod, 1989b).

Verilog had been developed as a simulation language that worked in concert with Synopsys' Design Compiler, the other key tools in the “front-end”. Logic synthesis was becoming more important to the EDA industry, but neither Mentor nor Cadence had such technology “in-house”. Trimeter Technologies introduced its Design Consultant in 1988 to compete with Design Compiler (Milne, 1988). Shortly after its adoption of VHDL,

Mentor acquired Trimeter Technologies in September, 1989, a month after Cadence merged with Gateway (Computer Business Review,1989e). Cadence remained dependent on Synopsys for logic synthesis until Cadence introduced its synthesis tool called Optivizer in 1991 (Maliniak, 1991a).

In 1988, the CAD Framework initiative was established to define an open standard for how third-party EDA tools could inter-operate (Maliniak, 1992b). The Design Framework became recognized as a key innovation that would compete effectively against Mentor's close platform strategy. By 1990, Cadence introduced its second generation Design Framework II. Its new framework was claimed to be fully compliant to the CFI standard set of tool and database APIs (Gunn, 1990a). Silicon Compiler had competitive layout technology, but also introduced its framework called Foundation in late 1989 (Magnus, 1989). In January, 1990, Mentor acquired Silicon Compiler, and announced that it would support in addition to Apollo workstations, Sun workstations, which was gaining market share in the EDA industry (Computer Business Review, 1990a).

Silicon Compiler's “back-end” or physical design tools and its framework technology became the new Falcon Framework (left side of Figure 4 below). Mentor had been working on a re-write of its software at the time of its acquisition of Silicon Compiler (Clarke, 1990a). Development of this new software (version 8.0) was delayed (Boerger & Willett, 1992), which enabled Cadence to continue to gain market share on

Mentor. In 1991, Cadence acquired Valid Logic, and the combined companies revenues now matched Mentor's (Computer Business Review, 1991b). In 1992, a project led by Hewlett-Packard was started to demonstrate the integration of Cadence's Verilog-XL with Mentor's Design Architect, design entry system (Maliniak, 1992a).

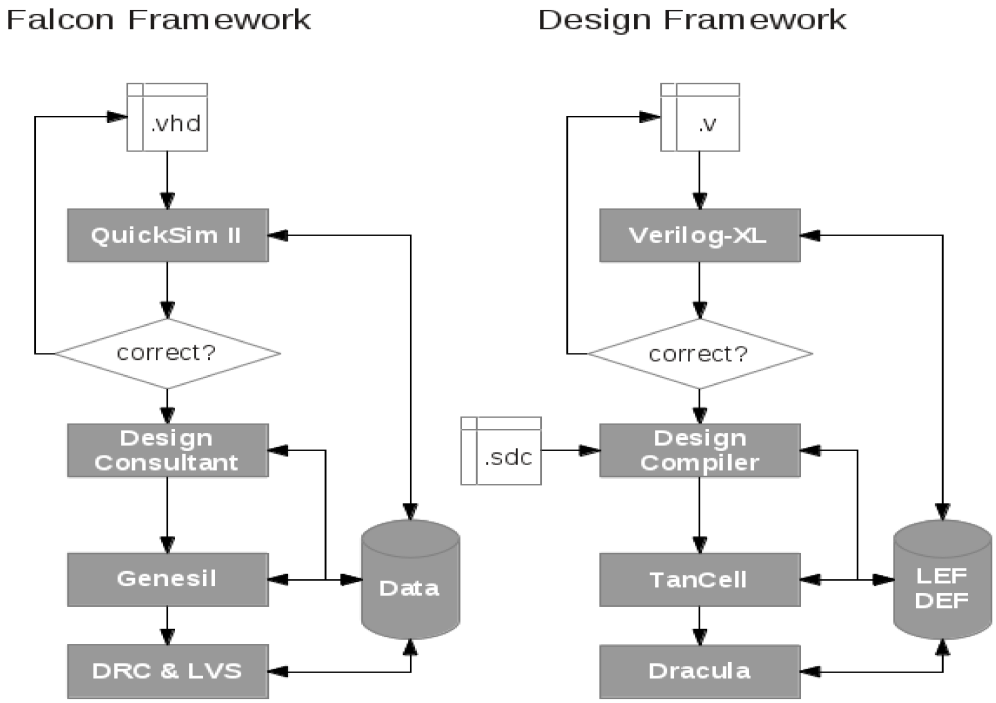


*Figure 4: Mentor's Falcon Versus Cadence's Design Framework*

While CAD frameworks were being standardized, many EDA vendors had followed Mentor Graphics to support the VHDL open standard HDL language (Clarke 1989), including Synopsys, which acquired the VHDL simulation business from Zycad in 1990 (Bloom, 1990). In response to the reasonable concern that Verilog, as a proprietary HDL, could not survive against the broader EDA industry support for the open VHDL, Cadence donated, in May 1990, the Verilog language to the Open Verilog International

 

organization for the purposes of promoting Verilog as an open standard (Vaughn, 1990).

A new company, Chronologic Simulation, was started by John Sanquinetti to commercialize the development of a Compiled Verilog Simulator (VCS) that would compete with Verilog-XL. This new simulator was the first demonstration of Verilog technologies developed from the recently opened language. Chronologic was acquired by ViewLogic in 1994 (Software Industry Report, 1994), although Synopsys had also made an offer. ViewLogic experienced some difficulties later, which alienated the Chronologic team (Goering, 1995).

From 1991 to 1995, Verilog and VHDL competed as Cadence and Mentor competed. Cadence released a VHDL simulator called VHDL-XL, or LeapFrog. VHDL had the benefit of being an IEEE standard, but Verilog supported a set of important features that limited VHDL as a significant technical threat to Verilog (Yu, 1993). The VHDL standard was updated in 1993 to support multi-valued logic based on technology donated by Synopsys. The Standard Logic support is referred to as IEEE Standard 1164. This was followed by support for standard ASIC libraries in the 1995 update to VHDL, referred to as the VHDL Initiative toward ASIC Libraries (VITAL) (Goering, 1995c). In 1994, the IEEE formed the Verilog working group, which lead to Verilog becoming the IEEE Standard 1364 in 1995 (Verilog-1995). Japanese support for Verilog was instrumental in the language becoming an IEEE standard. Sony's general manager for EDA tools was the vice-chairman of the IEEE 1364 Verilog working group, which started

in Japan in 1994 (Computer Business Review, 1994b).

Mentor renamed its VHDL simulator from System-1076 to QuickSim II in 1990 (Gunn, 1990b). In 1994, it acquired Model Technologies Inc (MTI), which had developed the leading VHDL simulator for Windows NT-based VHDL, called the V-System (Computer Business Review, 1994a). Mentor confused its customers by then offering QuickSim II and QuickHDL, based on MTI's technology, which operated as a separate division. After Verilog became an IEEE standard, the MTI division began to support Verilog. In 1997, Mentor integrated both simulation technologies into one product called ModelSim (Santorini, 1997).

Synopsys developed its platform strategy after years of domination of the logic synthesis niche market. Although it acquired the Zycad VHDL simulation technology in 1990, its lack of “back-end” technologies for IC layout prevents its classification as an EDA platform vendor until it made two key acquisitions in 1997, ViewLogic, which had acquired the other leading Verilog simulator (VCS), and Epic Technologies (Goering, 1997a). These two acquisitions brought Synopsys important Verilog simulation, static timing and IC layout technology, enabling Synopsys to offer its customers an alternative to Cadence and Mentor tools.

Another company, Avant! started in controversy, since Cadence claimed Avant! had stolen important IC layout technology when a key employee left Cadence to form Avant! Lawsuits and criminal charges were laid against Avant! (Goering,1995f), which

 

ultimately was found guilty as charged. Even so, while this dispute ran its course in the courts, Avant! grew impressively by making key acquisitions to emerge as the fourth largest EDA vendor, next to Cadence, Synopsys, and Mentor. Eventually, in 2002, Avant! was acquired by Synopsys (Electronic Packaging and Production, 2002), making Synopsys the leading EDA platform vendor.

Cadence made a key acquisition when it bought Coopers and Chyan Technologies (CCT) in 1996 (Langberg, 1996), which lead the US Department of Justice (DoJ) to investigate the business deal. The acquisition was allowed to proceed, but Cadence was required to not prevent other EDA tool vendors from access to its Virtuoso platform by closing its APIs (Katz & Shapiro, 1998).

Although the successful challenge to Mentor's dominance was enabled by the development of the CAD framework technology pioneered by SDA Systems, by 1995 CAD frameworks were in some ways rejected by the EDA industry. Standardization of framework technologies, like standardization of EDIF, was generally resisted as each vendor's implementation of the standards was designed to make its own tools perform better than other vendor's tools in terms of database access and inter-tool communications. Another reason was that users became disillusioned with frameworks for their “bugs”, which caused lost productivity . So, users simply stopped using frameworks (Cooley, 1994).

With the industry acceptance of two language standards, VHDL and Verilog, it

was also obviously burdened by the complex efforts to support both languages in what had been called a language war (Goering, 1996b, 1996e). As designs became more complex, and larger due to advances in semiconductor technology, the EDA industry was concerned about the evolution of languages towards a new concept called the Electronic System-level Language (ESL) (Fuller, 1995b). The EDA industry in general is very competitive, and many companies started to propose ESLs based on their own proprietary algorithms. Synopsys proposed its proprietary technology based on a C++ library it called SystemC (Goering, 1999c), which it donated to the Open SystemC Initiative (OSCI), but set itself up as the main sponsor of the OSCI.

Another company, CynApps, founded by the inventor of the VCS simulator, John Sanguinetti, proposed an alternative C++ library called CynLib. Synopsys was willing to release SystemC under a community source license (Clarke, 1999), but Sanguinetti understood that appropriability of the CynLib-based ESL methodology would not be reduced by the open source release of CynLib (Aycinena, 2004). Therefore, in 1999, CynApps released CynLib as an open source library which users could freely download from its website, and modify it anyway they wanted (Goering, 1999b).

Synopsys resisted for two years users and other vendor's (Mentor and Cadence, for example) calls for SystemC to be released as open source, like CynLib. Eventually, in 2001, a co-chair of the OSCI resigned and wrote an open letter to criticize the OSCI as a Synopys “marketing charade”. Mentor Graphics refused to join the OSCI unless

Synopsys would release control of the evolution of SystemC to the EDA community at large (Goering, 2001). To resolve the controversy, Synopsys relented to user demands to support SystemC as an open source initiative, which paved the way for Mentor, Cadence and most other EDA firms to join the OSCI (Clarke, 2001).

As a final point concerning open source strategies, Cadence released its DEF and LEF as open standards (Goering 1999d), and its TestBuilder, also a C++ library (Santorini, 2000b). But TestBuilder was not an ESL technology by comparison to CynLib or SystemC. In support of promoting open EDA tools and technologies, Cadence donated in 2002 a binary form of its Genesis CAD database, renaming it OpenAccess, and agreeing to release its database APIs to the Silicon Integration Initiative (Si2) (Goering, 2002b). Cadence also promised to release its source code, but under a community source license. Synopsys had acquired Avant! that year, and was unwilling to share its new Milkyway database.

## 4.3 Data displays

Table 10, Table 11, and Table 12 present the data collected about the platform strategies and motivations of Mentor Graphics, Cadence Design Systems, and Synopsys, respectively, over the time period from 1987 to 2002. This is followed by worked examples of how values were assigned to specific cells of the data collection tables. These worked examples are intended to illustrate how the data displays were produced through the rigorous and systematic application of the research method of Chapter 3.

*Table 10: Platform Strategy of Mentor Graphics (1987-2002)*

| Construct | Variable (attributes/units) | | Year | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mentor Graphics | | | 1980 | | | 1990 | | | | | | | | | | 2000 | | |
| | | | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| Platform | Standard<br><br>6 platform layers<br>(0: Closed;<br>1: Open) | Design Entry | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Simulation | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Synthesis | None | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Layout | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Database | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Framework | No Framework | | | SW Late | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Source<br><br>6 platform layers<br>(0: Not open source license<br>1: Open source license) | Design Entry | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Simulation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Synthesis | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Layout | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Database | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Framework | No Framework | | | SW Late | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Motivations | MS < MES | Market Share (%) | 27 | 32 | 36 | 38 | 33 | 28 | 27 | 24 | 24 | 19 | 17 | 15 | 15 | 16 | 15 | 15 |
| | | Long Term Ave. Profits (US$) | 20 | 33 | 45 | -14 | -62 | -51 | -32 | 27 | 50 | -5 | -31 | -0.5 | 2 | 55 | 31 | -14 |
| | MP < BDO | HHI due to firm only | 719 | 995 | 1305 | 1451 | 1124 | 820 | 702 | 592 | 576 | 379 | 284 | 235 | 226 | 247 | 236 | 217 |
| | | HHI Open (Std \| Src) | 23 | 50 | 184 | 2057 | 2215 | 2009 | 1661 | 1724 | 2080 | 1717 | 1776 | 2364 | 1822 | 1189 | 1793 | 1780 |
| | Tipping | # Top 3 Open Vendors | 0 | 0 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | # Top 3 Closed Vendors | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Decision to Accept Commoditization = 1 | Released Open Source | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Joined Open Source Initiative | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

*Table 11: Platform Strategy of Cadence Design Systems (1987-2002)*

| Construct | Variable (attributes/units) | | Year | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cadence Design Systems | | | 1980 | | | 1990 | | | | | | | | | 2000 | | | |
| | | | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| Platform | Standard<br><br>6 platform layers<br>(0: Closed;<br>1: Open) | Design Entry | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Simulation | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 10 | 1 | 1 | 1 | 1 |
| | | Synthesis | No synthesis tool | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Layout | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | | Database | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | | Framework | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Source<br><br>6 platform layers<br>(0: Not open source license;<br>1: Open source license) | Design Entry | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Simulation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Synthesis | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Layout | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Database | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Framework | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Motivations | MS < MES | Market Share (%) | 5 | 7 | 14 | 20 | 33 | 34 | 29 | 30 | 34 | 32 | 34 | 41 | 32 | 34 | 37 | 32 |
| | | Long Term Ave. Profits (US$) | N/A | 16 | 28 | 16 | -22 | 55 | -13 | 37 | 97 | 29 | 234 | 275 | 78 | 123 | 216 | 72 |
| | MP < BDO | HHI due to firm only | 23 | 50 | 184 | 410 | 1079 | 1163 | 825 | 911 | 1174 | 1039 | 1150 | 1702 | 1034 | 1146 | 1279 | 1045 |
| | | HHI Open (Std \| Src) | 23 | 50 | 184 | 2057 | 2215 | 2009 | 1661 | 1724 | 2080 | 1717 | 1776 | 2364 | 1822 | **1189** | **1793** | **1780** |
| | Tipping | # Top 3 Open Vendors | 0 | 0 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | # Top 3 Closed Vendors | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Decision to Accept Commoditization = 1 | Released Open Source | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | Joined Open Source Initiative | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

*Table 12: Platform Strategy of Synopsys (1987-2002)*

| Construct | Variable (attributes/units) | | Year | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Synopsys | | | 1980 | | | 1990 | | | | | | | | | | 2000 | | |
| | | | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| Platform | Standard<br><br>6 platform layers<br>(0: Closed;<br>1: Open) | Design Entry | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Simulation | No Simulation | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Synthesis | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Layout | No Layout Technology | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Database | No Database Technology | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Framework | No Framework Technology | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 |
| | Source<br><br>6 platform layers<br>(0: Not open source license;<br>1: Open source license) | Design Entry | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Simulation | No Simulation | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | Synthesis | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Layout | No Layout Technology | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Database | No Database Technology | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Framework | No Framework Technology | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |
| Motivations | MS < MES | Market Share (%) | 0 | 0 | 1 | 2 | 3 | 5 | 12 | 15 | 18 | 17 | 18 | 21 | 24 | 21 | 17 | 23 |
| | | Long Term Ave. Profits (US$) | Privately held | | | | | IPO | 16 | 15 | 31 | 14 | 72 | 123 | 191 | 112 | 56 | -200 |
| | MP < BDO | HHI due to firm only | 0 | 0 | 0 | 4 | 11 | 26 | 133 | 219 | 329 | 298 | 341 | 427 | 562 | 430 | 289 | 513 |
| | | HHI Open (Std \| Src) | 23 | 50 | 184 | 2057 | 2215 | 2009 | 1661 | 1724 | 2080 | 1717 | 1776 | 2364 | 1822 | 1189 | 1793 | 1780 |
| | Tipping | # Top 3 Open Vendors | 0 | 0 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | # Top 3 Closed Vendors | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Decision to Accept Commoditization = 1 | Released Open Source | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 |
| | | Joined Open Source Initiative | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 |

### 4.3.1 Mentor Graphics

Table 10 shows the data collected for Mentor Graphic's platform. This table show that its attributes for standards and source are initialized to 0, meaning Mentor begins with a closed platform with proprietary source code implementations. The milestones table shows that:

- Mentor adopted VHDL as being central to its account control strategy in early 1989. Since VHDL is the IEEE open standard for a hardware description language (HDL), a 1 is assigned to the Design Entry cell. Thereafter, Mentor Graphics operates with an open standard Design Entry platform attribute.
- Mentor offers the market its first VHDL simulation technology, named QuickSim, in 1990, so the Simulation cells from 1990 onward are assigned the value of 1.
- Mentor acquired Silicon Compiler in 1990, which gave the company important layout technology, but also CAD framework technology called Design Foundation. Due to delays in releasing the new version 8.0 of its EDA software, which included a port to Sun workstations, and the re-branding of the framework technology as “Falcon”, the Framework attribute is not assigned a 1 until 1992, when Mentor and Cadence's frameworks are shown to be inter-operable by HP researchers.
- Mentor's starts a number of years of sustained losses after 1989. Since software

has very low marginal cost, but very high profit margins in the EDA industry, this establishes that Mentor no longer operated above minimum efficient scale to support its proprietary R & D.

- Before 1990, the increase in the HHI for the EDA vendor operating an open platform is due entirely to Cadence Design Systems. From 1990 on, the entire EDA industry adopted VHDL (or Verilog which Cadence published), so the HHI from 1990 on represents the HHI of the open EDA platform vendors.

The above method shows that Mentor operated a Stage 1 platform strategy until 1990, when its loss of market share, profits and market power made it infeasible to continue operation in Stage 1, and thus Mentor began operating in Stage 2 from 1990 and continued to do so for a decade. Furthermore, in 2001, Mentor accepted the fact that Synopsys had released its control of the Open SystemC Initiative (OSCI), and modified the source code license for SystemC to an Open Source license from a Community Source license. Mentor's joining of the OSCI is therefore the event which signals its transition to operating in Stage 3 from 2001 on.

### 4.3.2 Cadence Design Systems

Cadence Design Systems was formed in 1988 by the merger of SDA Systems with ECAD Inc. It merged the next year with Gateway Design Automation and acquired Tangent Systems. These events are coded into Table 11:

- SDA Systems (later Cadence) controlled the key innovation – its Design

Framework – from 1987 onward, which enables an open standards-based EDA platform to be assembled from third-party tools. Therefore, a 1 is always assigned in the Frameworks row associated with platform standards for Cadence.

- Cadence merged with Gateway in 1989, which gave Cadence control of the proprietary Verilog standard for HDL-based Design Entry and Simulation. In 1990, because Cadence opened the Verilog language, the cells in the table for Design Entry and Simulation transition from 0 to 1 in 1990.
- Cadence did not have a logic synthesis capability until it introduced its Optivisor tool in 1991. There was no open synthesis standard. Synopsys dominated this market niche. Cadence tried in 1996 to drive an open standard, but is tiny (5%) market share in the synthesis market did not influence Synopsys. Nevertheless, because of this effort, a 1 is assigned in Platform Standards (Synthesis) in 1996. Since this effort failed, the value reverts to a 0 for the years following 1996.
- Cadence opened APIs to its Genesis Database in 2000, which requires a 1 to be assigned to the Platform Standards (Database) cell from 2000 on.
- Cadence released as Community Source code, its Genesis Database software in 2001 with intentions to later release it under an Open Source license, but this occurred after the period studied in this thesis. Since a Community Source license is not true Open Source, a 0 is assigned to the Platform Source (Database) from 2001 on.

- Cadence released to the public domain its physical design standards (data formats for exchanging of IC layouts between tools) LEF and DEF standards for library and data exchange formats in 2000, which requires a 1 to be assigned to Platform Standards (Layout) from 2000 on.
- For the same reasons as given for Mentor in the preceding section, Cadence is assigned a 1 for Design Entry and Simulation in the Platform Source category, starting in 2001.

It can therefore be established clearly that Cadence operated from the outset a platform strategy based on open standards, and therefore is deemed to be operating in Stage 2 from 1987 to 2000, and in Stage 3 from 2001 on, just like Mentor.

### 4.3.3 Synopsys

Table 12 presents the Synopsys results. For a number of reasons, Synopsys must be declared to have not operated an EDA platform at all until 1996. Two reasons are that Synopsys did not control any layout technology, and did not have any particularly important logic simulation technology either, until it made two strategic acquisitions in 1996.

- Synopsys established and dominates the logic synthesis market with a proprietary technology. Synopsys was able to resist efforts to drive on open synthesis standard, led by Cadence in 1996. Therefore, a 0 is maintained for the whole period of the study for Platform Standards (Synthesis).

- On the other hand, Synopsys acquired the VHDL assets of Zycad in 1990, and made VHDL an important aspect of its overall strategy – calling itself “a VHDL methodology company”. Therefore, for Platform Standards (Design Entry and Simulation), a 1 is assigned from 1990 on.
- Synopsys founded the Open SystemC Initiative (OSCI) in 1999. In 2000, the first release of SystemC source code was under a Community Source license, therefore a 1 is assigned in the Platform Source (Design Entry and Simulation) in 2000. The next year, bowing to EDA industry pressure from users and other vendors, Synopsys modifies its SystemC license agreement to an Open Source license, which removed the barriers for Mentor and Cadence to join the OSCI.

With no layout, database or framework technology, until Synopsys acquired ViewLogic and Epic Technologies, Synopsys simply cannot be considered to have operated an EDA platform until 1996, when it acquired the missing components from other vendors who had failed to establish successful EDA platforms. Therefore, similar to Cadence's pattern, in 1996, Synopsys starts its EDA platform strategy in Stage 2, but fully a decade after it began to pioneer the logic synthesis market niche. The key to understanding the transition to Stage 3 platform for all three leading EDA vendors' strategies is Synopsys' effort with the OSCI. Synopsys's transition to Stage 3 removed the barriers for Mentor and Cadence to make the same transition, almost simultaneously.

## 4.4 Composite Variables of the Pattern-matching Test

This section presents the completed table of platform strategy and motivation results required to assess the fit between observations and the expected pattern. For each of the three case companies, Table 13 reports the composite variables for platform strategy and motivation over the time period from 1987 to 2002, and assigns a "stage" to each year of operation according to the decision rules previously specified in section 3.3.6. The following paragraphs provide explanations and worked examples of how specific values in Table 13 were assigned.

*Table 13: Platform Strategies and Motivations of Leading EDA Vendors (1987-2002)*

| Construct | Variable (attributes/units) | Year | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1980 | | | 1990 | | | | | | | | | | 2000 | | |
| | | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| Mentor Platform | Standard (0: Closed; 1: Open) | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Source (0: Not open source license; 1: Open source license) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Mentor Motivations | MS < MES | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| | MP < BDO | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Tipping to the open standard | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Decision to accept commoditization | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Stage | 1 = Proprietary, 2 = Open Standard, 3 = Open Source | 1 | | | 2 | | | | | | | | | | | 3 | |
| Cadence Platform | Standard 0: Closed; 1: Open) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Source (0: Not open source license; 1: Open source license) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Cadence Motivations | MS < MES | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | MP < BDO | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **1** | **1** | **1** |
| | Tipping | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Decision to accept commoditization | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Stage | 1 = Proprietary, 2 = Open Standard, 3 = Open Source | 2 | | | | | | | | | | | | | 3 | | |
| Synopsys Platform | Standard (0: Closed; 1: Open) | No platform | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 |
| | Source (0: Not open source license; 1: Open source license) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Synopsys Motivations | MS < MES | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | MP < BDO | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Tipping | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Decision to Accept Commoditization | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Stage | 1 = Proprietary, 2 = Open Standard, 3 = Open Source | Not applicable | | | | | | | | | | 2 | | | | 3 | |

The two platform variables, standard and source, were determined using the decision rule specified in section 3.3.6. For example, in 1989, the Mentor Graphics platform employed closed standards at four layers of the EDA architecture (simulation, synthesis, layout, and database) and an open standard at one layer (design entry). At that time, the Mentor platform did not implement a framework layer. This information is obtained from Table 10 (p. 76) by inspecting the column for 1989. Following the decision rule of section 3.3.6, the Mentor platform in 1989 employs "open standards" because at least one of the six architectural layer employs an open standard. Thus, in Table 13, in the rows for the Mentor platform, the cell for standards in 1989 has the value 1. In the same year, the Mentor platform had no open source implementations at any layers of the architecture – all "source" cells in Table 10 for the year 1989 have values of either 0 or n/a. Thus, in Table 13, in the rows for the Mentor platform, the cell for source in 1989 has the value 0.

More worked examples are provided below:

- **Condensing Platform Standards and Source Layers to One Measure**. Any value of 1 in any layer of the platform standards or source for each firm applies to the whole platform.
- **Evaluating Market Share**. Losses (negative profits) cause the evaluation of MS < MES to be set to 1. For example, Mentor Graphics had four years of losses starting in 1990 (evaluates to 1), and two years of profits after (evaluates to 0).

- **Evaluating Market Power**. If several firms offer open platforms, the sum of their HHI contributions represents the buyer demand for open platforms. Mentor Graphics HHI contribution is seen to be many times that of the open platform vendors. The HHI of Mentor Graphics versus the HHI of vendors offering open platforms, this function switches from 0 to 1 in 1990 when both firms offer open standards in their platforms.
- **Condensing Tipping**. Tipping to open standards occurred in 1990 when the EDA industry widely switched to VHDL and Cadence released Verilog to the public via the OVI.
- **Decision to Accept Commoditization**. Any release of open source code or joining an open source initiative causes the evaluation to switch to 1 from 0 for the Decision to Accept Commoditization measure. Cadence's release of its TestBuilder C++ library in 2000 as open source is the reason why it moves to Stage 3 ahead of the other two firms.

## 4.5 Outcome of the Pattern-Matching Test

The outcomes of the pattern-matching test are described below.

### 4.5.1 Mentor Graphics Matches the Expected Pattern

Mentor Graphics follows the expected pattern. Its establishes a successful proprietary platform (Stage 1) consisting of bundled software that customers had to accept entirely, or not at all. A competitive open standards-based (Stage 2) platform entered the market, long-term financial losses began, buyer demand for openness increased and the contest tipped to open standards.

Thus, in the nomenclature of case research (section 2.3.2), the Mentor Graphics case is a *literal replication* of the West (2003) explanation.

### 4.5.2 Cadence Design Systems and Synopsys Start in Stage 2

Cadence Design Systems successfully attacked Mentor's Stage 1 platform strategy by starting its platform in Stage 2. Synopsys dominated the synthesis niche market, but then in 1996 acquired technologies and market share that enabled it to offer a full EDA platform to compete successfully against both Cadence's and Mentor's Stage 2 platform strategies.

The Cadence and Synopsys cases do not follow exactly the expected pattern. Further analysis is required to determine whether these two cases are *theoretical replications* of West's (2003) explanation, or *anomalies* that require refinements or extensions to West's (2003) explanation. That analysis is provided in the next chapter.

### 4.5.3 Evolution of System-level Languages Requires Stage 3 Operation

Synopsys proposed inter-vendor collaboration on the development of source code libraries for the ESL language SystemC, but neither Mentor nor Cadence would participate in the Open SystemC Initiative unless it was based on Open Source code licensing practices. Synopsys accepted that SystemC could not be commercialized without broad EDA industry support. The Community license was changed, and all three EDA platforms moved together into Stage 3.

The chapter has presented the results of the theory-testing stage this thesis, employed the analytic strategy of *pattern-matching* (Yin, 2003, pp. 116-120. The next chapter presents the results of the theory-building stage, employing the analytic strategy of *explanation-building* (Yin, 2003, pp. 120-122).

# *Chapter 5 Explanation of Results*

This chapter develops an explanation for the results of the pattern-matching test presented in Chapter 4. It is the outcome of step 5 of the five-step research method previously specified in Chapter 3 (Table 1 on p. 25 and section 3.5). It is organized as five sections. Section 5.1 is a concise summary of the results of the pattern-matching test of the previous chapter. The next three sections develop explanations for the platform strategies of the three case firms: Mentor Graphics in section 5.2, Cadence Design systems in section 5.3, and Synopsys in section 5.4. Section 5.5 is a summary and synthesis; drawing on the emergent explanations of Cadence Design Systems and Synopsys, it describes two platform strategies used by new entrants to compete successfully against a powerful incumbent with dominant market share.

## 5.1 Summary of Results of the Pattern-Matching Test

- Mentor Graphics platform strategy evolved over the period of the study to match the expected pattern. This pattern evolved in response to introduction of new platform strategies by Cadence Design Systems and Synopsys.
- Cadence then is observed to start as a Stage 2 EDA platform vendor by its mergers and acquisitions of several companies in 1988 and 1989.
- Synopsys pioneered and dominated by a huge margin the logic synthesis market, but for ten years did not establish itself as an EDA platform vendor. It did acquire VHDL simulation technologies from Zycad in 1990 to establish a solid front-end,

but lacked the back-end layout and verification tools, until it acquired ViewLogic and Epic Technologies in 1986.

- All three companies modified their EDA platforms to present a Stage 3 platform in 2001, when when Mentor, Cadence and many other EDA vendors, as well as a broad section of users pressured Synopsys into releasing control of the Open SystemC Initiative and releasing SystemC as an open source C++ library. Arguably, Cadence's release of the TestBuilder C++ library source code in 2000 indicates it made the transition to Stage 3 earlier – but this decision is actually overshadowed by the more significant decision to join the OSCI a year later.

## 5.2 Implications of a Successful Proprietary Stage 1 Platform Strategy

Mentor Graphics became the leading proprietary platform by offering bundled software that ran on a third-party workstation. Mentor's software solution represented a high barrier to enter the market for other vendors because of the high up front investment (sunk costs) in software development of a complete solution for electronics system design, especially integrated circuits. The platform design made it difficult for customers to export their data to other platforms or to use third-parties tools (that Mentor did not control).

### 5.2.1 Understanding Mentor's Proprietary 3-Plank Platform Strategy

To appreciate the breadth and scope of Cadence's successful scaling of the high barriers to entry into the EDA platform market, it is interesting to analyze Mentor's

proprietary (Stage 1) EDA platform as an integrated architecture based on these three planks:

1. **Mentor developed highly integrated EDA proprietary software**. This strategy worked very well for Mentor Graphics when its chief EDA competitors were Valid Logic and Daisy Systems, because they developed both software and hardware. Mentor's source code base was written specifically for the Apollo workstation, which Mentor also sold directly to its customers. Mentor's adoption of Apollo, a third-party hardware vendor, enabled the software-only strategy to benefit from the scale economies of Apollo workstations that were used in other markets besides hosting EDA tool suites.
2. **Mentor controlled all the software that could run on its platform**. Mentor's platform strategy was based on its control over which proprietary software its customers could use, a practice called software bundling. Bundling software presents compelling customer value when it solves a complex system design problem by offering an integrated suite of tools that work together and share a common, yet proprietary, data set.
3. **Mentor developed proprietary data formats and software interfaces that it did not share with its competitors.** Mentor's integrated suite of software tools for electronic system design was purposely engineered to make it difficult for a customer to export its design data to other platforms.

### 5.2.2 SDA Systems' Design Framework Bypasses Mentor's Planks

Mentor's successful Stage 1 platform strategy presented a high entry barrier for any head-on challenger that would offer another proprietary EDA platform. The following three points show that the Design Framework innovation by SDA Systems presented customers with the means to step over each of the three planks that were essential for Mentor's “account control” (platform) strategy:

1. **SDA Systems Design Framework software was modular and workstation independent.** SDA Systems Design Framework software architecture featured a design database with well-defined APIs. Software modules that implemented those APIs could therefore communicate with other tools in the framework, as well as the database. The software could run on the Apollo, but also the Sun workstation, which was growing in popularity.

2. **Customer's could select third-party tools as software modules.** SDA Systems's customers, for example Toshiba in Japan, valued the new and open platform concept because they could easily integrate what they considered to be the best modular third-party tools, such as Verilog-XL (simulator), Design Compiler (synthesis), Tangate or Tancell (IC layout), and Dracula (design rules check).

3. **Customers and third-party EDA tool developers had easy access to their data.** SDA Systems shared the Design Framework's APIs with other third-party EDA vendors, and their customers, so they could independently develop new

tools to rapidly advance the state of the art in EDA.

### 5.2.3 Cadence Consolidates the First Open EDA Platform

By 1988, SDA Systems framework technology was distributing to a handful of smaller EDA companies the economic returns of an open EDA platform. The new platform lowered the barriers to directly compete with Mentor Graphics' proprietary EDA platform. The privately-held SDA Systems then merged with ECAD, which had gone public, to form Cadence. In 1989, Cadence then bought Tangent and soon after merged with Gateway. It was expected to then acquire Synopys (McLeod, 1989e), but Synopsys never sold out. But in any case, the EDA industry understood by that expectation that Cadence's plan was to take control of the entire open EDA platform, and consolidate all its economic returns.

Cadence continued it acquisition strategy, and this did not escape the notice of the US Department of Justice, which investigated Cadence's acquisition of Coopers and Chyan Technologies in 1997 (Katz & Shapiro, 1998). The investigation concluded that the Cadence Virtuoso software was in itself a platform, and that for a fair market to operate, Cadence should be required to maintain open software interfaces to allow third-party tools to plug into the platform. This is ironic, considering this was the founding idea of the original Design Framework.

## 5.3 Explanation for Synopsys' Platform Entry at Stage 2 (Bypassing Stage 1)

### 5.3.1 Logic Synthesis Stood Alone

Unlike Cadence, initially Synopsys did not seek to compete directly with the large incumbent by marketing an EDA platform. Synopsys' initial business strategy was to pioneer and dominate the logic synthesis market. Logic synthesis is a bridge between the front and back ends of a modern EDA platform. Synopsys's acquisition of Zycad for its VHDL simulation technology led Synopsys to pursue its strategy of being a “VHDL methodology” company. Synopsys therefore was able to sell its technology into Cadence and Mentor accounts, because it had internal expertise to support both a Verilog and a VHDL flow. Synopsys therefore was able to appropriate the most economic returns of its synthesis innovation by remaining independent.

### 5.3.2 Synopsys Assimilates Lesser EDA Platform Vendors

Prior to Synopsys' own series of acquisitions beginning in 1996, the aggressive M&A strategy that Cadence demonstrated was replicated by Avant!, even though the theft of Cadence' layout technology soiled their reputation and mired them in legal trouble for years. Synopsys also acquired and enhanced its status as an EDA platform vendor, when it acquired both ViewLogic and Epic Technologies in 1996, and finally Avant! in 2002. The acquisition of Avant! did not escape anti-trust scrutiny (Office of Fair Trading, 2002), when the UK government investigated, but accepted the acquisition because the new combination of assets was largely complementary, and the majority of business dealings were in the US, where the DOJ would react if Synopsys were to “use its position to

foreclose competition in segments of the EDA software industry.

## 5.4 Explanation for Movement to Open Source

The EDA industry experienced both commercial and military drives to standardization. Two IEEE standards for hardware description languages VHDL and Verilog emerged. Each EDA vendor had to support both languages. Users religiously defended their adoption of one or the other HDL languages. The CEO of Cadence, Joe Costello, called the industry's support of both Verilog and VHDL a “$400 million dollar mistake” (Cooley, 2000). Later, the EDA industry recognized the need for the development of an electronic system-level language (ESL), which led to a competition was between proprietary and open source software development practices for the control of the evolution of SystemC (Low, 2012). Synopsys was at the center of this process. Its attempt to maintain control of SystemC was soundly rejected by the EDA vendor and user community. Therefore, Synopsys released control of the Open SystemC Initiative (OSCI) and released the SystemC C++ library source code under an open source license. This was the event that enabled both Mentor and Cadence to join the OSCI, and as a group, all three major EDA vendors firmly entered the open source era, and Stage 3 EDA platform dynamics.

## 5.5 Summary

This chapter explained that Mentor Graphics had developed three planks in its platform strategy – proprietary source code, software bundling of a suite of applications,

and control over the export of user data from the Mentor platform. SDA Systems developed the CAD framework innovation called Design Framework, which enabled its customers to substitute third-party software as modules that performed specific parts of the IC design process, in both the front-end and back-end. The SDA innovation led to an open systems EDA platform whose components were controlled by at least five small EDA companies. The framework  innovation was highly effective at stepping over each of the planks that Mentor had erected as barriers to enter into the EDA platform market. Cadence Design Systems developed and executed a strategy of mergers and acquisitions that brought control of the framework and three out of four of the top EDA tools that worked together to create the open EDA platform. The one leading technology that Cadence did not acquire was Synopsys' logic synthesis technology. As a result, Synopsys technology became the industry standard for logic synthesis, and efforts to form an open standard were not successful because Synopsys enjoyed a very high market share and therefore did not need to collaborate with the Cadence open synthesis initiative. In 1996 Synopsys acquired two companies, ViewLogic and Epic Technologies, that enabled it finally to offer a complete EDA platform with both front and back ends. Cadence continued its strategy of acquisitions with the Coopers and Chyan acquisition of the Virtuoso layout technology, which the DOJ investigated to ensure Cadence was not able to operate a monopoly. Avant! stole technical secrets from Cadence, and used a similar M & A strategy to quickly grow, but in 2002 Synopsys acquired Avant!, and this acquisition too was investigated and presented no anti-trust concerns. The open source movement

gathered in strength in the late 1990's and EDA firms developed different strategies associated with the promotion of C++ libraries where source code was released under various licenses. When the issue concerned the primary ESL that enabled more complex systems to be developed through the industry, the user demand for openness was too great to allow anything but a true open source license, and all three EDA vendors studied accepted this fact, and modified their platforms to mix both proprietary and open source implementations at various architectural levels.

The chapter has presented the results of the theory-building stage this thesis, employed the analytic strategy of *explanation-building* (Yin, 2003, pp. 120-122). The next chapter discusses the results.

# *Chapter 6 Discussion*

This chapter is a discussion comprised of four parts. First, it answers the research questions that motivated this thesis. Second, it compares the observed results to the literature that was reviewed in chapter 2. Third, it elaborates on the lessons learned and personal reflection of the author. Fourth, it explains how this thesis contributes to theory and practice.

## 6.1 Answer to the Research Question

This thesis was motivated by the following research question, introduced in section 1.1: *To what extent does West's (2003) three-stage explanation of platform strategies in the computer industry account for the strategic actions of the three leading firms in the EDA industry?* This section presents the answer to that question.

The platform strategies of the three leading EDA platform vendors followed three different paths. The first path closely follows the explanation of West (2003). The other two paths differ from West's explanation in ways that can be explained.

Mentor Graphics matched the expected three-stage evolution from proprietary to open standards to open source. Cadence bypassed the first stage (proprietary) and formed a Stage 2 EDA platform when it used a strategy of mergers and acquisitions to take control of an open standard EDA platform. Synopsys also bypassed Stage 1, but in a different way than Cadence.

Customers of Cadence (initially SDA Systems) could select their preferred EDA

tools, instead of being forced by Mentor Graphics to accept a bundle of Mentor's proprietary tools. Other companies tried to duplicate the merger and acquisition strategy that Cadence used to acquire control of the essential technologies that defines a modern EDA platform. When several of these firms faltered, Synopsys seized its opportunity to acquire the back end technologies so it could now present a complete platform solution.

Mentor Graphics also acquired companies to enable it to transition from a Stage 1 EDA platform vendor to a Stage 2 platform vendor. Cadence's rise to dominance was not just what companies it merged with or acquired, it also managed its affairs very well, whereas Mentor Graphics was less effective, and its software development got out of control. The result was the loss of several critical years in the early 1990s when its software was late, allowing Cadence to cement its position as the new market leader.

The late 1990's was an era when open source code practices forced EDA vendors to acknowledge the clearly strong buyer demand for openness. Synopsys fought for several years against this new trend, in order to maintain its proprietary control of the evolution of the next generation ESL language, SystemC. Eventually, Synopsys let go of the idea of controlling SystemC, and this signaled the start of Stage three for all three EDA vendors studied in this thesis.

The answer to the research question therefore has two parts:

(1) The strategic actions of leading firms in the EDA industry is *consistent with the underlying logic* of West's (2003) explanation – that firms employing a platform strategy

 

face an essential tension between appropriability and adoption, and attempt to open “just enough” to attract enough buyers while retaining adequate returns. Thus firms prefer more closed platform strategies, but transition to more open platform strategies when more closed strategies become infeasible.

(2) To fully explain all observations, a refinement to West's explanation is required. A new entrant need not enter at stage 1 with a proprietary platform strategy. If a proprietary platform strategy is *already infeasible*, a new entrant will enter at stage 2 with a platform strategy employing open standards.

## 6.2 Comparison with the Literature Reviewed

This section compares the results found in this thesis about the evolution of platform strategies the EDA industry to the literature on platforms and software markets previously reviewed in chapter 2. The first subsection examines the agreement with five rules for platform success. The second subsection examines the agreement with five characteristics of software markets.

### 6.2.1 Five Rules For EDA Platform Success

These five rules for platform success identified from the literature reviewed in section 2.5 were observed to be operating in the EDA industry:

1. **Overcoming barriers to imitation**. Cadence did not imitate Mentor's Stage 1 proprietary platform strategy. Cadence's Stage 2 open EDA platform strategy lowered the barriers to compete effectively against Mentor's successful

 

proprietary EDA platform by sharing technical standards with competitors.

2. **Control of complementary assets**. The key to Mentor's Stage 1 proprietary strategy was its practice of bundling its software and limiting its customers' ability to run third-party tools. Cadence's platform offered customers more control over their design data once it was entered into the EDA platform. Cadence established its Stage 2 platform by a strategy of mergers and acquisitions that targeted EDA tool vendors who offered the most attractive complementary assets to its SDA Systems Design Framework. Other EDA vendors tried to imitate Cadence's mergers and acquisitions strategy, such as as ViewLogic and Epic, and most notably Avant!, but were later acquired by Synopsys in 1996 and 2002, respectively.

3. **Control, create and evolve platform APIs**. Mentor controlled its platform by developing software based on closed API standards, which made it difficult for third-parties to offer complementary software. Cadence used its early lead in the open EDA platform to ensure that its increasingly popular platform complied to the open standards of the CAD Framework Initiative. Synopsys refused to cooperate on the Cadence-led initiative to develop open standards for logic synthesis, because of its dominant market share in that market.

4. **Tap unserved market niches before reaching critical mass**. In the late 1980's and early 1990's, the Japanese commercial semiconductor industry was an

important niche market with more advanced requirements than demanded by the flagging US semiconductor industry. Before Cadence executed its M & A strategy, at least five smaller EDA vendors addressed the unserved needs of large Japanese customers for openness in an EDA platform. The Cadence Stage 2 platform was first seen as separate EDA tool offerings that could be integrated by the customer using the SDA Systems Design Framework. The innovative Design Framework allowed customers to select their preferred EDA tools, which addressed growing customer demand for openness. This was disruptive to Mentor's practice of bundling software that it controlled. The Synopsys Stage 2 platform evolved out of the market power that Synopsys enjoyed when it pioneered and dominated the newly introduced logic synthesis market, a niche market which was central to all EDA platform vendor's future strategies.

5. **Strong network effects attract developers to the increasingly successful open EDA platform**. The emerging success of the SDA Systems Design Framework in the Japanese market made it an attractive platform for the development of third party tools to complement it.

### 6.2.2 Five Features of Software Markets

The five features of software markets identified from the literature reviewed in section 2.5 were observed to be operating in the EDA industry:

1. **Software markets feature strong network effects, which tend to increase**

**market concentration**. The EDA platform industry is dominated by three large platform vendors whose collective share of the market increased over the period studied.

2. **Strong network effects cause the standards contest to tip to the technology leader**. The proprietary Cadence standard for Verilog HDL was technologically more advanced than IEEE standard VHDL-1987. The release of the Verilog language, combined with its technical superiority, acknowledged by the Japanese, led with their help to its standardization by the IEEE in 1995.
3. **A free market will support just one standard**. Two HDL standards were introduced because the US military mandated the use of VHDL. The proprietary Verilog standard was meeting the needs of the commercial IC industry, led by the Japanese, whose semiconductor industry was technologically more advanced than the military-dominated US semiconductor industry. Free market behavior was observed when Synopsys and other companies competed to offer differing ESL languages, particularly Cynlib versus SystemC. The result was that only one standard did survive with the best efforts of all contributors to the open source initiative for SystemC standardization.
4. **Software buyers base their choices on expected network sizes**. Verilog survived because users expected that its 1990 public release and superior features (compared to VHDL until 1995) would lead to other vendors introducing products based on the open standard. VHDL survived because of the support that Mentor

Graphics, and vendors other than Cadence, such as Synopsys, gave the language in 1989. Cadence rapidly grew once it had firmly taken control of most of the entire platform from the original companies (except Synopsys) who developed the set of third-party tools that made the platform technologically superior to Mentor's.

5. **Positive returns to scale coupled with strong network effects lead to very high profits**. Once Cadence consolidated and digested its acquisitions in 1989 to 1990, it experienced a decade of huge profits and market dominance of the EDA industry. Synopsys showed this principle when it used its dominance of the logic synthesis market to maintain high profits and avoid being acquired by larger EDA companies.

This subsection and the previous subsection have examined the ways in these results are in agreement with results from the extant literature. The next subsection reports on contribution of the thesis, emphasizing the ways in which these results differ from the extant literature.

## 6.3 Lessons Learned In the Writing of this Thesis

In this section I elaborate on some lessons I have learned in the writing of this thesis. There are two subsections. In the first subsection, I introduce my industry experience as an IC designer and architect and discuss the opportunities and liabilities for this research. I explain how I acquired my experience in the EDA industry and developed strong personal opinions about how the industry operates. Then I describe how the

 

research approach developed for this thesis offers a powerful technique to distinguish between opinion and evidence, and to benefit from both. In the second subsection, I offer some practical insights I have learned about how to approach the writing of a thesis.

### 6.3.1 Opportunities and Liabilities of Field Experience

As an IC industry expert, I am not a detached, unbiased external observer, but instead have first-hand knowledge of the time period, the companies, and the technologies examined in this thesis. My studies in Electrical Engineering began in 1987, the first year in the period studied in this thesis. As a junior engineer in Ottawa in 1993, I designed my first "integrated circuit" using schematic entry on the Mentor Graphics / Apollo platform. My formal IC design training began the next year at Nortel's research arm, Bell-Northern Research (BNR). I was introduced to Verilog HDL in a course offered by Cadence Design Systems, with simulations running on Sun workstations using Verilog-XL. My industry training continued with a course taught by the noted Verilog expert, Cliff Cummings, who presented the new methodology called “top-down” design using the Verilog language that featured a small microprocessor. The logic was synthesized using Synopsys' Design Compiler, and verified by re-running the Verilog simulation on the resulting gate-level netlist.

By 1996, I was working in a start-up networking company to design high-performance digital logic for gigabit Ethernet switch chip sets in VHDL. Simulation of the design was done with ViewLogic simulators (soon to be acquired by Synopsys), which included the famous Chronologic Verilog Compiled Simulator (VCS), invented by

 

John Sanguinetti (Low, 2012). By 1998, I was the technical lead at IBM's applications-specific IC (ASIC) design center in Ottawa, where I helped Nortel and Cisco Systems to develop some of the world's most challenging communications IC designs done to that date. I was a Linux and open source enthusiast, and IBM was supportive. At that time, between 1998 and 2000, IBM's EDA tools were leading the evolution of high performance ASIC designs with advanced static timing and physical synthesis technology that Synopsys later acquired.

I am deeply invested in this field setting: I contributed, invented, and used my knowledge and expertise to earn a good living and feed my family, and later to start my own company. With the advanced training in world-class IC design at BNR (Nortel) and IBM, I was recruited to another start-up company as the Senior IC Architect to lead a large design team to develop the world's fastest Secure Sockets Layer (SSL) session setup chip. I filed for at least three patents in encryption and network security processing. I began a second career as an IC design consultant, inventor and entrepreneur. My company (Crack Semiconductor) has successfully introduced to the market a line of advanced Silicon Intellectual Property (Silicon IP) and network security processors (Low, 2011; Low & Muegge, 2013)

I began research for this thesis with many stories of things that happened, and many opinions about why things happened. But other people have different stories and different opinions. There is no easy way to determine whose opinions are right, or even if anyone's opinions are right. Like John Sanguinetti (Low, 2012), I am a Verilog bigot,

 

which means that I have never held much appreciation for the VHDL language, yet I have designed ICs in both HDLs. From my experience, I think Verilog is easier than VHDL to use and get required results. The Verilog versus VHDL debate continues today, and many people in the industry disagree with me. I have a client in France who has never used Verilog, and does not question the superiority of VHDL. Readers of this thesis need to understand that the opinions held by many in the EDA industry were and remain deeply polarized in regard to which HDL standard is better. The debate over language superiority continues today with similar opinionated arguments about SystemC versus SystemVerilog, which emerged as an IEEE standard in 2005.

I have learned in this thesis that what separates opinion from evidence-based knowledge is the method used to answer the question. Deeply held opinions are treated as facts by their owners. Each opinion is the "correct answer" to a given question. This explains why the research method developed in this thesis can be presented as an important contribution. There is a difference between the statements of strong opinion holder and the answer produced by a properly formulated research question and its methodical investigation to gather evidence to support reasonable conclusions.

Being an IC designer with direct experience of the operation of the EDA industry naturally has led me to hold some opinions. Before my research method was firmly understood, my opinions were a constant source of distraction. My methodology focused my investigation, but also severely limited my ability to bias both the data collection and analysis phases of my case studies.  I then simply had to post-process my data using a

 

purely mechanical process that led me very quickly to answer my research question. My opinion that was derived from my personal experience yielded to a more reliable history of the EDA industry as a narrative based on facts discovered by organized research. My experience informed me about how to specify and test my assertions, what data to collect, and how to obtain the data.

My intent was to write a thesis, but in the course of doing this work, I learned about knowledge production, expertise, opinion, evidence, overcoming bias, learning new things about a domain that I already "understand", and convincing others. It may well be that the thesis question is interesting to some, yet the research method I have developed in this thesis is of more universal interest. I started out asking a specific question, and in order to answer it, I have developed a more general solution to the method of answering questions in domains that are defended by expert bias and where strongly held opinions earned by anecdotal experience can be effectively challenged.

I have elevated the discussion of platform strategies in the EDA industry from rival opinions (with no clear way to judge which is best) to a critique of logic and method. My arguments, method, and results are explicit and transparent in way that enables useful critique, and potentially could be improved as a result of critique (rather than rejected or displaced by a rival opinion). This insight, which emerged during my research, is consistent with arguments by Levin (1954), Van de Ven (1989), Christensen & Raynor (2003), Pfeffer & Sutton (2006) and others about the central pragmatic role that theory and evidence can play in management practice.

### 6.3.2 Practical Insights for the Thesis Writer

In this second subsection of lessons learned, I offer to other thesis writers three of my personal insights on how to save time and effort in the literature review, setup and writing of a thesis.

First, attend thesis progress reviews of students who are further along in their own research and talk to the students to get their insights into the obstacles they have encountered that have slowed their thesis developments.

Second, set up a robust thesis document template before the writing of the thesis becomes the critical path. In the final analysis, a thesis is a document that presents to very critical readers a structured statement of the research question and its detailed and methodically-developed answer. When the writing of the thesis moves into the final phase, and the deadline for submission looms, the master's student cannot afford the distraction of technical formatting problems, such as page numbering, headers, footers, automatic table of contents, lists of tables, figures and so forth. High on the list of priorities should be a clear understanding of the formats for citations and the bibliography. These technical issues should be solved at the beginning of a thesis course, not in the middle, or the end.

Third, read broadly; review more than the literature necessary to understand how to answer the research question based on the selected research methodology. A master's program requires a lot of reading. A good master's program present a broad theoretical

 

background as required reading in the general curriculum. Use these readings as opportunities to write quality, succinct summaries of all papers read in the course of the master's program. In general, I found that readings of papers in the entire master's program should be done with the objective of writing a good summary of the paper, and the identification of the salient points. Some of these summaries and salient theoretical statements then can be directly applied to the literature review. Additionally, the research question will require the development of specialized theoretical knowledge that is specific to the thesis, and these readings should be guided by the requirements of satisfying the research method. Good notes are valuable later when developing and revising the thesis literature review.

## 6.4 Contributions

This thesis makes four contributions to research methodology, and two contributions to management theory and one contribution to management practice.

### 6.4.1 Four Contributions to Research Methodology

**Contribution 1:** The research approach – combining pattern-matching and explanation-building with the industry expertise of a principal researcher – can be applied more generally to other research questions about industry evolution and change.

**Explanation:** The author of this thesis is an IC engineer and architect with decades of experience with the EDA tools from all three of the vendors studied, and experience with both HDL languages and the ESL language, SystemC. Industrial expertise helps in a

number of ways: to operationalize the explanation for a specific industry context, design the pattern-matching test to compare observations with predictions, decide which data are needed to perform the test, and to decide where and how to get the data. Once the architectural layers are identified – which may require several rounds of data collection by the expert to verify that the architectural model will enable good data to be collected – the data collection method requires less expertise and can be assigned to non-experts. A researcher without deep industrial expertise would have significant difficulty with those parts of the research design. Different industry expertise would be required for different industry contexts.

**Contribution 2:** The research method, which uses pre-defined rules for the post-processing of collected raw data, can be applied by persons who are not subject matter experts, or by persons who are subject matter experts, but may have an opinion which might introduce bias.

**Explanation:** The use of pre-defined rules for data post-processing is an important step in the research method for two reasons. First, the existence or non-existence of an event can be coded without expert experience. Second, because the rules are explicit and strictly defined, the biases of the coder are less influential. A second investigator or a research assistant could independently complete some or all steps of data collection and analysis as a tactic for addressing threats to validity and reliability (Yin, 2003, p. 34). That second investigator need not be an industry expert. The use of pre-defined rules for post-processing, demonstrated here, is available for use by others.

**Contribution 3:** Pattern-matching is an effective research design for testing explanations of platform strategies.

**Explanation:** Pattern-matching is well-documented by Yin (2003), and has been successfully employed in other domains of inquiry, but has not previously been employed in research on platforms. Pattern-matching was used effectively in this research to test the explanatory power of West's (2003) explanation of evolving platform strategies. Other researchers could replicate this research design to test explanations of platform strategies in other industry contexts, or adapt the specific tests performed here to test West's explanation of platform strategies in other industry context.

**Contribution 4:** The Herfindahl-Hirschman Index (HHI) provides an operational measure of platform market power.

**Explanation:** A pattern-matching approach requires precise *a priori* specification of constructs, variables, methods, and expectations (Yin, 2003). This thesis adapted the HHI, previously employed by the United States Department of Justice as a measure of firm market power when assessing the impact of a proposed merger or acquisition, as a measure of platform market power. Other researchers may also find this to be a useful measure for empirical studies of platform strategy.

### 6.4.2 Contributions to Management Theory

**Contribution 5:** West's (2003) explanation has validity beyond the computer industry context in which it was developed; it is valid also within the context of the EDA industry.

**Explanation:** The evolving platform strategies of the three leading EDA vendors provide

one literal replication (Mentor Graphics) and two theoretical replications (Cadence Design Systems and Synopsys) of West's (2003) explanation. This is significant because the EDA industry differs in some important ways from the computer industry. For example, it is much smaller (by one to two order of magnitude), more concentrated, and based on a different platform architecture. Furthermore, the three leading EDA vendors are direct competitors, whereas West's firms competed indirectly with different value propositions.

**Contribution 6:** West's explanation can be refined so that firms enter at stage 2 if a proprietary platform is already infeasible.

**Explanation:** This path not found in West's data, but is entirely consistent with the logic of his explanation. A theoretical replication of West (2003) is observed in the cases of Cadence Design Systems and Synopsys with this explanation: If a proprietary platform is infeasible due to some combination of technical and economic factors, a new entrant will enter at stage 2 rather than stage 1. In this case, the appealing benefits of a proprietary platform – better barriers to imitation and better margins – are not available.

### 6.4.3 Contributions to Management Practice

**Contribution 7:** This thesis describes how West's explanation can be systematically applied to provides managers in the EDA platform industry with an analytic tool with which they can better understand how market forces shape platform evolution. Early identification of the market forces that affect platform feasibility can enable faster and more methodical decisions on how and when the platform can be modified to respond to

emerging threats.

**Explanation:** West's explanation requires analysis of platform evolution by separating the architectural layers and then observation of how, in reaction to competitive pressure, one or more of those layers are modified to comply to open standards that are shared with competitors, or to adopt open source implementations that are visible to all.

# *Chapter 7 Conclusions*

This chapter concludes the thesis, addresses the limitations of the research design, proposes recommendations for future research.

## 7.1 Summary

The thesis has conducted a pattern-matching test to compare the evolving platform strategies of the three leading EDA platform vendors, Mentor Graphics, Cadence Design System and Synopsys, to the predictions of an explanation developed by West (2003) in the context of the computer industry.

Following the introduction of the thesis, Chapter 2 presented a review of the literature focusing on software platform strategies, network effects and their role both in the establishment of standards and in motivating firms to collaborate on open standards. A definition was presented for the minimum efficient scale in software industries. The basis for measuring market power using the HHI comes from the DOJ's horizontal merger guidelines. Yin's case study research methods, including pattern matching concepts were reviewed and summarized. A summary of process theory from Van de Ven was presented.

Chapter 3 presents the five steps in this thesis's research method. The method begins with the identification of the key theoretical constructs and associations between constructs. These constructs were then mapped to the EDA industry context in order to

identify a set of variable, measures and sources for data collection. A table shell was developed which features a row for a measure of each variable and a column for each year of the study. Two sets of variables, one concerning standards, and the other concerning source code, were associated with each architectural layer of an EDA platform. A third set of variables were defined to measure the motivations of the firm's managers to modify their EDA platforms to open standards or open source. Variables for market share and vendor profits combine to show when a firm gains or loses minimum efficient scale. Variables for the vendor's market power and the market for open (standard or source) platform layers combine to show when a firm can no longer resist buyer demand for openness. Tipping to the open standard was measured by counting the number of the top three EDA platform vendors which operated with a Stage 2 or Stage 3 platform. A decision to accept commoditization was measured by observing when a firm released open source code or joined an open source initiative. A pattern was specified as a table with expected values for platform standards, source and West's four motivations.

Data were then collected to fill in the three blank table shells for each EDA platform vendor in the study. Raw data in the form of extracts from archived articles from the EDA industry were stored in a database. The raw data were processed to develop a table of EDA industry milestones for the industry in general and for each vendor. The milestone table presents data that enables each platform layer to be quantized as presenting either a proprietary standard (0) or an open standard solution (1), and either a closed source (0) or an open source implementation (1). The annual reports of each of

 

the three EDA vendors, plus supplemental data from archival sources provide values for revenues (from which market share was calculated), profits (losses), and for market power (the square of a firm's market share in percent).

The completed table shell for each company enables the objective assessment of what stage the vendor was operating in for any particular year. This enabled the completed table shell to serve as the information on which the pattern match was based.

This thesis has shown that there are at least two new ways that a platform vendor can compete against a successful incumbent. One way is to identify the emerging market leaders for each layer of the platform, and then attempt to merge with or acquire these companies. This has to be done early enough so that the cost of acquisition is ultimately profitable. The other way to compete against powerful incumbents is to develop market power within a critical market niche that the incumbents depend upon. For this to succeed, the technology and market application needs to be identified early, control of it needs to be maintained, and strong revenue growth needs to be re-invested to both grow and maintain profits at a high enough level that investors are happen to pursue a long term strategy.

## 7.2 Limitations

This thesis studied three companies that became fierce competitors in the same market for EDA tools. On the other hand, West's study of the computer industry featured three companies in the computer industry with fairly non-overlapping markets. This

thesis is therefore limited to the study of competitors in a primarily software market which features high up front development costs, very low marginal costs, and requires constant innovation by each vendor to counter the strategic developments of its key competitors.

## 7.3 Recommendations for Future Research

This thesis did not go into great detail concerning in-depth innovation in each particular architectural layer of the EDA platform. Two promising areas for future research in the EDA industry are the development of two HDL standards, which was influenced by the power and influence of the US military requirements for vendors to use VHDL, and the development of open source software strategies, which had just begun during the last two years of the period studied by this thesis. Of interest to many would be research to explain how Synopsys was able to dominate the logic synthesis market, and why it was able to resist so well being taken over by the larger EDA companies when such an acquisition would have been highly profitable for the acquiring firm.

Two promising areas for future research outside the EDA industry are the replication of the pattern-matching test to examine platform strategies in other industry contexts and the application of the pattern-matching methodology to examine other questions of industry evolution and change where strongly-held opinions may impair objective analysis.

# *Chapter 8 References*

The references are presented in two sections, with articles in scholarly and practitioner journals presented first in section 8,1, and case study sources from trade journals, company reports, newspapers, and websites presented second in section 8.2.

## 8.1 References in Scholarly and Practitioner Journals

Aycinena, P. 2004. John Sanguinetti - A profile. ***EDA Nation***, September. Available online: http://chipdesignmag.com/edanation/september2004/#2

Arthur, W. B. 1996. Increasing returns and the new world of business. ***Harvard Business Review***, 74(4): 100-109.

Besen, S. M., & Farrel, J. 1994. Choosing how to compete: Strategies and tactics in standardization. ***Journal of Economic Perspectives***, 8(2): 117-131.

Bresnahan, T. F., & Greenstein, S, 1999. Technology competition and the structure of the computer industry. ***Journal of Industrial Economics***, 47(1): 1-40.

Carlile, P. R., & Christensen, C. M.. 2005, The cycles of theory building in management research, Version 6.0. Available online: http://www.innosight.com/documents/Theory%20Building.pdf

Christensen, C. M., & Raynor, M. 2003. Why hard-nosed executives should care About management theory. ***Harvard Business Review***, 81(9): 66-74.

Cusumano, M. 2010. The evolution of platform thinking. ***Communications of the ACM***, 45(1): 32-34.

Davis, M. S., 1971. That's interesting! Towards a phenomenology of sociology and a sociology of phenomenology. ***Philosophy of Social Sciences***, 1(4): 309-344.

Eisenhardt, K. M., 1989. Building theories from case study research. ***Academy of Management Review***, 14(4): 532-550.

Evans, D. S., Hagiu, A. & Schmalensee, R. 2006. ***Invisible engines: How software platforms drive innovation and transform industries***, Cambridge, MA: MIT Press.

Gans, J. S., & Stern, S., 2003. The product market and the market for “ideas”: Commercialization strategies for technology entrepreneurs. ***Research Policy***, 32(2): 333–350.

Gawer, A. 2009. ***Platforms, Markets and Innovation***. Northampton, MA: Edward Elgar.

Gawer, A., & Henderson, R. 2007. Platform owner entry and innovation in complementary markets: evidence from Intel. ***Journal of Economics & Management Strategy***, 16(1): 1-34.

Greenstein, S. M., 1997, Lock-in and the cost of switching mainframe computer vendors, ***Industrial and Corporate Change***, 6(2): 247-274.

Jick, T.D., 1979. Mixing qualitative and quantitative methods: triangulation in action. ***Administrative Science Quarterly***, 24: 602-11.

Lewin, K., 1945. The research center for group dynamics at Massachusetts Institute of Technology. ***Sociometry***, 8: 126-135.

Liebowitz, S. J. & Margolis, S. Network effects. Available online: http://www.utdallas.edu/~liebowit/palgrave/network.html

Liebowitz, S. J. & Margolis, S. Path dependence. Available online: http://www.utdallas.edu/~liebowit/palgrave/palpd.html

Low, A., 2011. Developing silicon IP with open source tools. ***Open Source Business Resource***, May: 19-25. Available online: http://timreview.ca/article/442

Low, A., 2012. Making money from exploiting Schumpeterian opportunities: John Sanguinetti and the electronic design automation industry. ***Technology Innovation Management Review***, May: 18-22. Available online: http://timreview.ca/article/555

Low, A., & Muegge, S., 2013. Keystone business models for network security processors.. ***Technology Innovation Management Review***, July: 25-33. Available online: http://timreview.ca/article/703

Katz, M. L. & Shapiro, C., 1985. Network externalities, competition and compatibility. ***American Economic Review***, 75(3): 424-440.

Katz, M.L. & Shapiro, C., 1986. Technology adoption in the presence of network externalities. ***Journal of Political Economy***, 94(4): 822–841.

Katz, M.L., & Shapiro, C., 1998. Anti-trust in software markets, ***Competition, innovation and the Microsoft monopoly: Anti-trust in the digital marketplace, February 5, 1998, Washington, DC, Proceedings,*** 29: 82.

Miles, M. B. & Huberman, M. A., 1994. ***Qualitative Data Analysis, An Expanded***

 

***Source Book, Second Edition.*** Thousand Oaks, CA: Sage Publications.

Muegge, S. M., 2013. Platforms, communities, and business ecosystems: Lessons learned about technology entrepreneurship in an interconnected world. ***Technology Innovation Management Review***, February: 5-15. Available online: http://timreview.ca/article/655

Teece, D., 1986, Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy, ***Research Policy*** 15(6): 285-305.

Van de Ven, A. H. 1989. Nothing is quite so practical as a good theory. ***Academy of Management Review***, 14(4): 486-489.

Van de Ven, A.H. 2007. ***Engaged scholarship: A guide for organizational and social research***. New York, NY: Oxford University Press.

West, J. & Dedrick, J., 2000, Innovation and control in standards, ***Information Systems Research***, 11(2): 197-216.

West, J., 2003, How open is open enough? Melding proprietary and open source platform strategies, ***Research Policy***, 32(7): 1259-1285.

Yin, R., 2003. ***Case Study Research Design and Methods, Third Edition.*** Thousand Oaks, CA: Sage Publications.

## 8.2 References in Trade Journals, Company Reports, and Websites

Ajluni, C., 2000. System-Level Languages Fight To Take Over As The Next Design Solution. ***Electronic Design***, January 24.

Albany Times Union, 1986. Once dead high tech company begins breathing again. ***Albany Times Union***, November 2.

Arensman, R., 1998. Squeeze play: Electronic design automation vendors face new era. ***Electronic Business***, June 1.

Aycinena, P., 2000. Panel points finger at EDA interoperability mess -- Open-source idea gets kudos. ***Electronic Engineering Times***, June 19.

Aycinena, P., 2005. Phil Moorby – Kaughman Award. Available online at: http://www.aycinena.com/index2/index3/archive/phil%20moorby.html

Berman, V., 1992. Hardware description language meets top down design criteria. ***Canadian Electronics,*** February 1.

Bloom, M., 1990. Synthesis/simulation suites a must for EDA: Synopsys acquires Zycad Corp's full-VHSIC Hardware Description Language simulation business. ***EDN***: October 4.

Boerger, E., & Willet, K., 1992. What went wrong with 8.0? ***EDN***: June 8.

Boston Globe, The, 1989. Gateway to announce its sale. ***The Boston Globe***: October 3.

Brooks, R., & Giesen, L., 1985. Boeing Military slates $17M for CIM in '85. ***American Metal Market***, February 11.

Chen, N., Gau, T., & Liao, Y., 1986. Symbolic layout software accelerates IC design. ***Electronic Design***, June 12.

Clarke, M., 1989. VHDL and frameworks will dominate at DAC '89. ***EDN***, June 15.

Clarke, M., 1990a. Sparc is Mentor's second platform: prompted by SCS acquisition; hope to port before 1991. ***EDN***, February 8

Clarke, M., 1990b. DAC '90: Clear view of next-generation concepts; tool integration, test compilation, and multichip-module design stand out. ***EDN***, June 14.

Clarke, P., 1999. SystemC license terms come under question. ***Electronics Engineering Times***, November 22.

 

Clarke, P., 1999. Upstart language wins EDA backers. ***Electronic Engineering Times***, November 22.

Clarke, P., 2001. Open and shut cases at DAC: Synopsys reversal on OSCI opens door to Mentor membership. ***Electronic Engineering Times***, June 25.

Cole, B., 1989. Making the right moves in ASICs. ***Electronics***, November 1.

Collett, R., 1995a. Cadence - fight or flight? ***Electronics Engineering Times***, September 4.

Collett, R., 1995b. Illusory monopoly power. ***Electronic Engineering Times***, February 6.

Collett, R., 1996. Let's not fool ourselves. ***Electronics Engineering Times***, June 3.

Collett, R. 1998. Wedding bells, death knells: Cadence acquisition of Ambit Design Systems. ***Electronics Engineering Times***, September 14.

Computer Business Review, 1987a. Company Results: Daisy Systems. ***Computer Business Review***, October 28.

Computer Business Review, 1987b. Company results: Valid Logic. ***Computer Business Review***, February 10.

Computer Business Review, 1987c. SDA Systems has three-year agreement with Toshiba, Innovative Silicon Technology. ***Computer Business Review***, May 5.

Computer Business Review, 1987d. Company Results: Valid Logic. ***Computer Business Review***, May 17.

Computer Business Review, 1987e. Company Results: Valid Logic. ***Computer Business Review***, August 6.

Computer Business Review, 1987f. Company Results: Valid Logic. ***Computer Business Review***, October 29.

Computer Business Review, 1987g. Tangent tools for ASICs adopted by Motorola Inc, Toshiba Corp. ***Computer Business Review***, December 10.

Computer Business Review, 1988a. Company Results: Valid Logic. ***Computer Business Review***, February 3.

Computer Business Review, 1988b. Company Results: Valid Logic. ***Computer Business Review***, April 28.

 

Computer Business Review, 1988c. Company Results: Valid Logic. ***Computer Business Review***, July 31.

Computer Business Review, 1988d. Company Results: Valid Logic. ***Computer Business Review***, October 23.

Computer Business Review, 1988e. Company Results: Daisy Systems. ***Computer Business Review***, October 25.

Computer Business Review, 1989a. Company Results: Valid Logic. ***Computer Business Review***, February 13.

Computer Business Review, 1989b. Cadence acquires Tangent Systems. ***Computer Business Review***, March 12.

Computer Business Review, 1989c. Mentor Graphics introduces a wealth of new design tools for its design workstations. ***Computer Business Review***, April 4.

Computer Business Review,1989d. Unix is the way ahead as OS2 loses more ground every day, says ViewLogic Systems. ***Computer Business Review***, May 18.

Computer Business Review, 1989e. Mentor Graphics takes over the assets of Trimeter Technologies. ***Computer Business Review***, June 26.

Computer Business Review, 1989f. Cadence Design Systems to acquire Gateway Design Automation. ***Computer Business Review***, October 11.

Computer Business Review, 1989e. Company Results: Daisy Systems. ***Computer Business Review***, November 27.

Computer Business Review, 1990a. Mentor Graphics to acquire Silicon Compiler Systems, Adopt SUNs. ***Computer Business Review***, January 14.

Computer Business Review, 1990b. Daisy Systems in chapter 11 bankruptcy protection. ***Computer Business Review***, August 2.

Computer Business Review, 1990c. Company Results: Valid Logic. ***Computer Business Review***, February 19.

Computer Business Review, 1991a. Company Results: Valid Logic. ***Computer Business Review***, February 6.

Computer Business Review, 1991b. Cadence Design to pay $198M in shared for IBM partner Valid Logic. ***Computer Business Review***, October 3.

 

Computer Business Review, 1992. Company Results: ViewLogic Systems. ***Computer Business Review***, July 9.

Computer Business Review, 1993. Company Results: ViewLogic Systems. ***Computer Business Review***, January 20.

Computer Business Review, 1994a. Mentor to acquire Model. ***Computer Business Review***, December 9.

Computer Business Review, 1994b. Momentum building for Verilog description language to be agreed as standard as Japanese working group is formed. ***Computer Business Review***, December 14.

Computer Business Review, 1996. Company results: Avant!. ***Computer Business Review***, January 22.

Computer Business Review, 1998. Company result: Avant! Corp. ***Computer Business Review***, January 20.

Computer Business Review, 2001. Avant! reports 7th consecutive year of record growth. ***Computer Business Review***, February 12.

Cooley, J., 1994. CFI? The shape of things to come. ***EDN***, June 23.

Cooley, J., 2000. Another religious war revisited. Available online: http://www.deepchip.com/items/snug00-07.html

Coy, P. 1987. US technology losing its edge. ***Chicago Sun Times***, January 4.

Curran, L., 1991. Challenging the leader: two new Verilog tools from Cadence invade Synopsys' turf. ***Electronics***, March 1.

Dahlberg, R., 1989. Where logic synthesis fits. ***EDN***, June 15.

Davis, E., 1989. Mining for gold in CAE. ***EDN***, May 4.

EE Times, 1997. Ambit, Synopsys to lock horns in logic synthesis. ***Electronic Engineering Times***, February 17.

EE Times, 2003. EDA -- Legal fireworks. ***Electronic Engineering Times***, January 13.

Electronic Design, 1987. Best of 1987: Software & CAE. ***Electronic Design***, December 23.

 

Electronic Packaging & Production, 2002. Synopsys buys Avant! ***Electronic Packaging & Production***, February 1.

Electronics Weekly, 2000. Avant! reports revenues for fourth quarter, year. ***Electronics Weekly***, January 26.

Fitch, M., 1997. A market-beating pro picks five low-priced stocks that could climb up to 72%. ***Money***, September 1.

Foundylier, C., 1992, What's hot and what's not in EDA. ***Computer-Aided Engineering***, September 1.

Fuller, B., 1995a. EDAC offering new market-survey data. ***Electronic Engineering Times***, April 17.

Fuller, B., 1995b. Outlook for EDA is good. ***Electronic Engineering Times***, October 23.

Harbert, T., 1988. VHDL challenges industry. ***EDN***, September 1.

Goering, R., 1995a. Bilingual simulation solutions emerging. ***Electronic Engineering Times***, March 27.

Goering, R., 1995b. EDA friends turn foes in merger rift. ***Electronic Engineering Times***, May 29.

Goering, R., 1995c. Vital spec goes to ballot; standard will guide VHDL ASIC libraries. ***Electronics Engineering Times***, June 5.

Goering, R., 1995d. Mentor ups mixed-signal. ***Electronics Engineering Times***, October 16.

Goering, R., 1995e. Startup targets high-level synthesis. ***Electronics Engineering Times***, November 13.

Goering, R., 1995f. Police raid stuns Avant!. ***Electronics Engineering Times***, December 11.

Goering, R., 1996a. Design-automation industry soared in '95. ***Electronics Engineering Times***, February 5.

Goering, R., 1996b. HDL wars subside, but trouble persists. ***Electronics Engineering Times***, March 4.

Goering, R., 1996c. Synthesis in spotlight: VHDL International develops synthesis

 

standard without Synopsys. ***Electronics Engineering Times***, June 24.

Goering, R., 1996d. EDAC comes out with industry 'report card.' ***Electronic Engineering Times***, July 8.

Goering, R., 1996e. EDA group eyes next-gen language. ***Electronic Engineering Times***, October 14.

Goering, R., 1996e. Synopsys won't back common synthesis effort: Synopsys Inc announces second technology patent for register-transfer-level subset. ***Electronics Engineering Times***, December 16.

Goering, R., 1997a. A Synopsys deal of Epic proportions. ***Electronics Engineering Times***, January 20.

Goering, R., 1997b. Avant! makes sweeping bid for broadline. ***Electronic Engineering Times***, February 10.

Goering, R., 1997c. Cadence moves Verilog to NT in major EDA shift. ***Electronics Engineering Times***, May 5.

Goering, R., 1997d. ASICs to Windows: Mergers rock EDA: Synopsys buys Viewlogic Systems, and MicroSim may be acquired by OrCAD. ***Electronics Engineering Times***, October 20.

Goering, R., 1999a. DePalma off to a QuickStart. ***Electronics Engineering Times***, April 19.

Goering, R., 1999b. Startup steps up open-source EDA with publication of class library. ***Electronics Engineering Times***, August 30.

Goering, R., 1999c. Open software dominates ESC -- Synopsys-led group pushes C++ initiative in attempt to transform hardware design. ***Electronics Engineering Times***, September 27.

Goering, R., 1999d. Cadence cedes LEF but its future's uncertain. ***Electronics Engineering Times***, November 15.

Goering, R., & Santorini, M., 1999. Cadence CEO departs after stock plunges. ***Electronics Engineering Times***, May 3.

Goering, R., 2000a. Let's open SystemC. ***Electronic Engineering Times***, March 13.

Goering, R., 2000b. SystemC 1.0 ready for download. ***Electronic Engineering Times***,

 

April 10

Goering, R., 2000c. Web site takes the lead in delivering standards for tool interoperability -- EDA users rally for open source. ***Electronic Engineering Times***, June 12.

Goering, R., 2001. Antitrust complaints roil SystemC backers. ***Electronic Engineering Times***, June 11.

Goering, R., 2001. Synopsys opens Vera in language standard effort. ***Electronic Engineering Times***, April 2.

Goering, R., 2001. Cadence approach wins consortium backing; Avanti preps 11th-hour counterproposal -- EDA users rally around open-standard API. ***Electronic Engineering Times***, June 18.

Goering, R., 2002a. But Synopsys slams Cadence-centric data model -- Genesis backers: Let there be OpenAccess. ***Electronic Engineering Times***, February 4.

Goering, R., 2002b. Cadence to deliver 'production' OpenAccess code. ***Electronic Engineering Times***, December 16.

Goering, R., 2005. Verilog's inventor nabs EDA's Kaufman award. ***Electronic Engineering Times***, November 7.

Gold, M., 1984. Integrated workstations will answer the challenge of VLSI development. ***Electronic Design***, May 31.

Gunn, L.,1990a. Tools, framework let engineers mix system design methods. ***Electronics Design***, June 14.

Gunn, L., 1990b. Standards and integration steal the show at DAC. ***Electronics Design***, June 28

Hooper, L., 1986. Military projects spur demand for automated systems. ***Defense Electronics***, April 1.

Langberg, M., 1996. California's Cadence Design to acquire Cooper & Chyan Technologies. ***Knight Ridder/Tribune Business News***, October 29.

Magnus, S., 1989. Design frameworks meet with hope and skepticism. ***EDN***, May 4.

Maliniak, L.,1991a. Synthesis tools complete front-to-back EDA system. ***Electronic Design***, February 28.

 

Maliniak, L., 1991b. New version of synthesis technology raises design abstraction. ***Electronic Design***, April 11.

Maliniak, L., 1992a. CFI framework standards get put to the test. ***Electronics Design***, June 11.

Maliniak, L., 1992b. CAD frameworks ride a rough road to success. ***Electronics Design***, August 6.

Matsumoto, C., 1997. Cadence makes software-sharing Connections. ***Electronic Engineering Times***, January 6.

McLeod, J., 1989a. A giant leap for simulation. ***Electronics***, February 14.

McLeod, J., 1989b. CAE giant Mentor Graphics bites the VHDL bullet. ***Electronics***, February 1.

McLeod, J., 1989c. Why Cadence is top dog in the IC design pack: the answer for this maker of CAD chips is mergers, timing and product. ***Electronics***, June 1.

McLeod, J., 1989d Quickening the automation pace in CAD-CAE. ***Electronics***, June 1.

McLeod, J., 1989e. Synopsys: the right tools at the right time. ***Electronics***, November 1.

McLeod, J., 1989f. Going the distance with mergers and acquisitions: President and CEO Joe Costello guides the growth of Cadence Design Systems Inc. ***Electronics***, December 1.

McLeod, J., 1990. A new kind of engineering fuels CAD; 'concurrent engineering' sparks multilevel simulators and a push for frameworks. ***Electronics***, April 1.

McLeod, J., 1991. All talk, no action. ***Electronics***, June 1.

Milne, B., 1989. Synthesize your ASICs from RTL behavioral blocks. ***Electronic Design***, June 9.

Nield, G., 1991. Silicon Valley. ***Canadian Electronics***, May 1.

Office of Fair Trading, 2002. Completed acquisition by Synopsys Incorporated of Avant! Corporation, ***Office of Fair Trading***, Available online: http://www.oft.gov.uk/OFTwork/mergers/mergers_fta/mergers_fta_advice/synopsys#.UewQ7uM5tJA

Ohr, S.,1985. EDIF standard lacks the discipline to keep its offspring compatible.

 

**E*lectronic Design***, November 28.

Open Source Initiative, 2013, The Open Source Definition, ***Open Source Initiative***. Available online: http://www.opensource.org/osd

Oregon Business, 1989. Annual ranking of Oregon's public companies. ***Oregon Business***, June 1.

Rawles, J., 1987. VHSIC technology comes on line. ***Defense Electronics***, August 1.

Richards, E. 1990. San Jose standout revels in 'Corporate Culture'. ***Washington Post***, July 29.

Rosenberg, R., 1987. Telesis Systems to merge with Callifornia company. ***The Boston Globe***, February 7.

Runyon, S., 1984. The goal is to automate the engineer's tasks. ***Electronic Design***, February 23.

Santorini, M., 1997. Mentor, MTI merge tools. Mentor Graphics and subsidiary Model Technology Inc to consolidate Quick HDL and V-System simulation tools into ModelSim 5.0. ***Electronics Engineering Times***, August 18.

Santorini, M., 1999a. EDA users debate merits of opening Vera language. ***Electronic Engineering Times***, April 26.

Santorini, M., 1999c. New license and service revenues drive growth -- EDA industry steams ahead in the first quarter. ***Electronic Engineering Times***, August 16.

Santorini, M., 2000a. Chronology brings language wars into verification. ***Electronic Engineering Times***, June 19.

Santorini, M., 2000b. Cadence gives away C++ testbench class library. ***Electronic Engineering Times***, August 28.

Schindler, M., 1984. Transforming a simulated IC into silicon demands tools that unravel design decisions. ***Electronic Design***, December 13.

Schindler, M., 1987. CAE users want open systems, integration. ***Electronic Design***, September 3.

Shandle, J., 1990. Will vendors line up behind objectivity? ***Electronics***, April 1.

Software Industry Report, 1994. ViewLogic buying Chronologic. ***Software Industry***

 

***Report***, April 18.

Software Industry Report, 1998. EDA industry revenues clime close to $3 billion milesteone in 1997. ***Software Industry Report***, April 20.

Sematech, 2013. Corporate history. Available online: http://www.sematech.org/corporate/history.htm

Synopsys, 2013. Company Profile. Available online: http://www.synopsys.com/Company/AboutSynopsys/Pages/CompanyProfile.aspx

Takahashi, D., 1995. Cadence Design Systems reverses slide through tailored customer service. ***Knight Ridder/Tribune Business News***, March 5

US Department of Justice, 2011. Horizontal Merger Guidelines. Available online: http://www.justice.gov/atr/public/division-update/2011/merger-guidelines.html

Vaughn, J., 1990. How close is top-down design? ***EDN***, June 14.

VHSIC Program Office, 1990. Very High Speed Integrated Circuits. United States Department of Defense. Available online at: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA230012

Wiegner, K., 1991. We have to change. ***Forbes***, September 30.

Weiss, R., 1985. The hardware language devised for VHSIC is poised for takeoff. ***Electronic Design***, January 24.

Yu, T., 1993. Create optimal simulation libraries using VHDL. ***EDN***, May 13.