Cost Feasibility Analysis for Embedded System Development and the Impact of Various Methodologies on Product Development Cycle

Ankur Agarwal, Ravi Shankar
ankur@cse.fau.edu, ravi@cse.fau.edu
Center for System Integration
Dept of Computer Science and Engineering
Florida Atlantic University, Boca Raton, FL-33431

Abstract

The cost and the time-to-market for a product will increase exponentially if innovations are not brought and deployed into the product development cycle (PDC). This is due to the fact the system complexity is increasing exponentially with time. This increasing complexity results from introduction of new protocols, higher radio frequencies, and increasing number of applications, among others, in the case of mobile devices. Such factors have led to the exploration of the multi-core architecture. The software (SW) and hardware (HW) concurrency, and need for optimization, in such architectures, will further complicate the product development process. Therefore, it is imperative that we introduce new and innovative ways to enhance productivity. Researcher have proposed new concepts such as executable specifications, component based design, system modeling language (SysML), model driven design (MDD), software decomposition, abstract performance evaluation, system level verification, field programmable gate array (FPGA) based network-on-chip (NOC), and abstract concurrency modeling. These proposed modeling, design and development techniques aim at increasing productivity by introducing leading edge design methodologies into PDC. In this paper we present a cost model that aims at understanding the current PDC and at evaluating the impact of new technologies on PDC. We focus on cost and product development time, and compare them for the existing PDC and the new PDC that we have conceptualized. We refer to this new PDC as "One Pass to Production" (OPP). This cost model may also help one identify potential bottlenecks and areas for further automations.

Key words: Design Productivity, Feasibility Analysis, System Design Flow, Software Design Flow, FPGA Design Flow

Tools and Languages: Excel Sheet, The GUI is being developed in Microsoft Visual Studio.NET

Introduction

System complexity, driven by both increasing transistor count and customer need for increasingly savvy applications, has increased so dramatically that system design and integration can no longer be an after-thought. As a consequence, system level design, long the domain of a few expert senior engineers, is coming into its own as a discipline. For the field to grow and enhance engineering design productivity, many key concepts have had to be adopted from other domains. Advances in semiconductor technologies have led to continual and rapid transistor scaling; this will soon facilitate the integration of a billion transistors on a small size integrated circuit. Product development time will increase from the current two years to five years or more, thanks to the increasing hardware complexity and the concomitant increasing sophistication of the applications demanded by the consumer. In order to maintain the product development time at two years or less, many concepts, such as architecture platforms, software design reuse, and model driven architecture, have been recently proposed and implemented. However, an integrated approach that synergistically combines the stages of specification, design, development, and prototyping has been missing. Our research center "*Center for System Integration*" has undertaken to address this.

Under such scenario, companies are looking for ways to get the most out of their design resources. In keeping up with the market pressure new technologies, languages, design methodologies and tools have been introduced. It is seen that a new technology is introduced in form of a language, which is represented in form of a design methodology after it gets matured. A successful design methodology is then later represented in form of an EDA tool. Design, development and verification engineers are then able to use these tools in their cutting edge produced design cycle (PDC). For example, SystemC – a hardware-software co-design language was introduced back in. New design methodologies were developed around

SystemC involving the design flows for hardware/software co-design, co-development and co-verification process. New tools such as, platform architecture, model designer, virtual platform designer, catapultC, NCSim simulator among others were introduced in order to support these design methodologies. It is expected that future design methodologies must support an increase in the amount of automation, design reuse and at the same time these design methodologies must be able to address the design concern at the highest level of abstract – System Level. Customers in today's competitive market environment are resistant to even "moderate" increase in cost; and the rate of doubling functions per chip (Moore's Law) is slowing [9]. Thus semiconductor manufacturers and systems companies must seek a new model to deliver the same cost-per function reduction that has fueled the industry growth.

Today, embedded system companies have to bet on consumer preferences two years hence and start developing a product to meet that demand which may not come to pass. A wrong bet may adversely the fortunes of a company. Therefore, the product design cycle needs to be further shortened, not just maintained at 24 to 30 months. Hence, more innovation is needed, especially at the earlier stages, since errors introduced there get magnified at later stages. As an example, NASA estimates that an error at the specification stage will require 138 times the effort to fix it at the prototyping stage and 536 times the effort to fix it in the field, as it would at the. It is expected that the future systems will have an increasing role of design automation, reusability, and componentization, thus increasing the market share for the EDA industry.

2. Impact of Cutting Edge Research Areas on Product Development Cycle

In this research we have analyzed the bottleneck and pitfalls in the current design methodology. This analysis has been carried out based on Cocomo cost and time analysis for software product development. We have further extended this study to represent the overall cost of PDC which includes hardware, software and RF cost. From this analysis we have focused on these resource consuming phases in a PDC and have researched innovative and multidisciplinary approaches for enhancing the productivity of system design. This research also takes advantage of many system engineering innovations that the industry has already evolved and applies them to requirement specification, design, development and prototyping phases.

2.1. Requirement Analysis

2.2. Specification Productivity

Eliminating errors in requirements phase can lead to substantial reduction in development cost and time. It is very important that as requirements pass through different phases of product development they are not misinterpreted and applied incorrectly or omitted. This can happen due to language grammar barrier, ambiguity, and incorrectness in the requirements. We show a process which implements the use of executable specification (digitization of requirements). We have focused on communication between user, marketing team, architects and developers with the use of UML, XMI, Java-based Animation.

2.3. Executable Specification

2.4. XML

2.5. Concurrency Modeling

Designers exploit design reuse to enhance system design productivity. Integration of pre-designed reusable blocks may fail if these blocks execute in parallel, share resources, and/or interact with each other. Such concurrency issues, if not addressed, may be detrimental to normal functioning of the system and may lead the system into a deadlock or a livelock state. Multiprocessor architectures, recently introduced to extend the applicability of the Moore's law, depend upon concurrency and synchronization in both software and hardware to achieve that goal. System design integration and verification approaches will not be cost-effective in exposing concurrency failures as they are intermittent; this can be costly (significantly increased time to market and field failures). One would have to develop abstract concurrency models and do exhaustive analysis on these models to test for concurrency problems. We have conceptualized a systematic approach that models concurrency by using an abstract symbolic modeling language to build a systematic high level concurrent system model; we also use an exhaustive analysis and verification tool to confirm that the system is devoid of concurrency failures. We propose that concurrency issues must be modeled after the system specification step and before defining system components. Concurrency modeling will help us to design reusable design block, thus is likely to enhance system design productivity.

2.6. Automated Code Generation

Unified Modeling Language (UML) provides is the ability of modeling at different levels of abstractions. Since C++ is object oriented, a modeling language such as the UML can be used to develop object modeling based systems. Accordingly, UML satisfies the basic requirements to link specifications with implementation: visualization, modularization and abstractions. The Unified Modeling Language provides model visualization through a variety of different types of diagrams. The UML diagrams are divided into three main categories: interaction diagrams, structural diagrams, and behavioral diagrams. Interaction diagrams show the existing relationships as well as the exchange of messages within the system. Structural diagrams show the building blocks of the system. Behavioral diagrams show the system's reactions to external and internal events. UML provides thirteen different diagrams. The usage of these diagrams depends on what is being visualized. For instance, use case diagrams show the services that actors can request from the

system, while sequence diagrams show the exchange of messages among the different modules in the system, and class diagrams show real-world entities and their relationships.

2.7. System Modeling Language
2.8. Component Based Design
2.9. Abstract Performance Modeling

With the continuing exponential growth of circuit complexities, early planning and understanding of the different trade-offs earlier in the design cycle becomes very critical, especially given the huge cost of correcting an early stage mistake later on. Traditionally, in this stage, a senior engineer ('Architect') with expertise and/or background in all the related engineering disciplines uses a spread sheet type of tool to conduct 'What-If' scenarios and decide on various component technologies (RF, Applications, OS, Architecture, Communication, and VLSI) that, based on technology projections and legacy experience, will lead to a new generation product two years hence. However, today's Architect is burdened with component technologies that are rapidly advancing and their integration into a synergistically working system is a virtual impossibility, unless of course certain system engineering principles are applied and supported a priori.

This software is for designing and analyzing systems on a multi-domain environment. The simulations on MLDesigner provide performance parameters, such as throughput, latency, resource usage, and network load. These parameters can be added with performance parameters extracted from hardware implementation such as, power consumption, maximum frequency of operation and cost (in terms of silicon area) to generate a specification set for different configuration of the NOC. This specification set can be used by the system architect to select the NOC configuration. Thus, system performance evaluations and what-if scenarios can be performed in the beginning of the design phase. This meets the requirements of the system to be designed without the need to invest time in designing the communication backbone from scratch.

2.10. Layered Architecture

Layered architecture is an effective way of dividing and conquering a problem. We have seen it working in an effective way in the of open system interconnection (OSI) model. It provides a way to separate the concerns of each different domain thus providing an effective solution to a domain specific problem. Such a platform can separate domain specific issues in separate layers, which will allow for more effective modeling of concurrency and synchronization issues, in an attempt to develop an optimized system. It can also be argued that this approach will provide us with a scalable solution in addressing inter-domain specific issues. We have defined a NOC based layered architecture for future embedded systems.

2.11. Software Decomposition

Multicore Architectures cannot be fully and effectively utilized with sequential model based software. Software cannot be re-written entirely in order to utilize next generation multicore architectures. We have developed a 10-Step methodology which effectively reverse engineers existing (Legacy) software to convert it into concurrent model based software. The aim is to solve embedded software and real-time concurrency issues while partitioning code over multiple concurrent architectures. We utilize top-down representation, bottom-up annotation, and middle out analysis to effectively implement this methodology. We perform performance annotation by analyzing computation, communication and concurrency cost.

2.12. SystemC : Hardware Software Co-design
2.13. System Level Verification

Typically, about 50% of the product design cycle time was spent on testing the product [12] [14]. Thus, more automation in the verification and testing phases of circuit design became inevitable. This reduced the product design time and effort. Several test methodologies have evolved during in past. This included assertions, RTL test benches, verification languages (such as Vera), fault coverage and automatic test generation. It also saw the introduction of formal verification methods with languages such as Sugar [15] [16]. However, these more abstract and early verification tools are yet to be adopted widely.

2.14. NOC Architecture

NOC communication sub-system is likely to enhance the productivity of system design phase as it provides an architectural platform for managing complexity and incorporate a reusable communication backbone. The NOC design comprises of two layers: network protocol layer and communication backbone layer. One of the aims of this NOC implementation is to develop IP libraries for packet-switched communication backbones for multi-core systems. This reduces development time spent on the communication backbone for multi-core system, the cost of designing the final product. The developed libraries have reusable, customizable and parameterizable components.

2.15. Model Driven Architecture
2.16. Resource Estimation
2.17. Annotation
2.18. Automating Overall OPP Process
2.19. Integrating Legacy Applications

3. Common Factors Affecting Cost Model for Embedded System

For developing an overall system cost model, we have partitioned the cost model into two different sub-models for software (SW) and hardware (HW). The final system cost model has been evolved from SW and HW cost model. For the conventional methodology (referred to here as Non-OPP methodology) has been developed based on the fact that it takes about two years time to develop an embedded system. We refer to our design methodology as OPP design methodology. For developing this cost model, we have assumed various parameters. Some of these parameters are common to both OPP and Non-OPP design methodologies while others are specific to a particular design methodology. The parameters which are common to both design methodologies – OPP and Non-OPP have been listed below:

3.1. Engineer's Cost

We assumed the following: the cost of an engineer is about $150K per year (including all the fringe benefits; there are about 230 working days in a year (365-52-52-31); and thus, the cost of an engineer per day is about $652.

3.2. System Complexity Factor

We know that the embedded system complexity is increasing every year as per the Moore's law. In today's technology the number of transistors in a chip and hence, the functionality of an embedded device doubles in approximately three years instead of eighteen months. Thus, approximately there is a 33% increase in the complexity of embedded devices. This increase in complexity can be realized in form of various advances that embedded systems are going through. Some of these advancements include (1) Introduction of new protocols. (2) More complex user interface. (3) Higher Quality of Services. (4) Higher radio frequencies (5) Multi-core architectures (6) Increasing Power Consumption (7) Advance video applications (8) High support for user services. This has increased the software component is the embedded devices and has resulted into the research of new domains viz. software decomposition for multi-core architectures, system verification, managing power consumption on embedded system devices among others. Thus, we represent this increase in complexity by a "Complexity Factor" in the cost model. We have assumed that the complexity increase every year. The rate of this increase in complexity for five years is shown in Table 1.

Table 1: Complexity Factor for next five years

Factor	1st Year	2nd Year	3rd Year	4th Year	5th Year
Complexity Factor	1.00	1.33	1.67	2.00	2.67

Thus, it is expected that the design time and the cost of embedded system would increase in accordance with this complexity factor if new innovations are not bought into the PDC.

3.3. Management and Co-ordination Factor

As the design complexity increases, a company will have to devote more resources in handing with this increase in the design complexity. Thus, it is likely that an increasing amount of management of the product development team and an equally increasing amount of co-ordinations among various engineers in a team might be needed. We represent this parameter in terms of "Management and Co-Ordination Factor". We represent this parameter in Table 2.

Table 2: Management and Co-ordination Factor for next five years

Factor	1st Year	2nd Year	3rd Year	4th Year	5th Year
Management and Co-ordination Factor	0%	5%	10%	20%	40%

As this factor increase with time the cost and time of product development is likely to increase declining the system design productivity.

4. Cost Analysis for Non-OPP Design Methodology

A high level system design flow include requirement analysis, specification, architectural selection, RF Design and development, HW/SW Co-design, HW/SW co-development, HW/SW co-verification and virtual prototyping. It is a myth that the product is designed in different phases of PDC: specification, design, development and verification. But the reality is that the most of these phases are overlapping with each other. This is due to various reasons. (1) Sometimes a new specification/requirement has to be accommodated in the middle of the design phase. Thus, we re-do some of the specification and design – an overlap between specification and design phase. If this new requirement comes during the development phase then we will see an overlap among specification, design and development phase. (2) We usually start the verification process early in development phase. In particular, unit testing is done before the complete development phase, thus providing an overlap between development and verification phase. (3) An error or a glitch in the verification/testing phase pushes the product to the design phase followed by the development and then verification phase.

This in reality instead of sending 100% time in the design flow shown above, we usually send 150% time. This project myth is shown in Figure 1(a) and the project reality is shown in Figure 1(b).

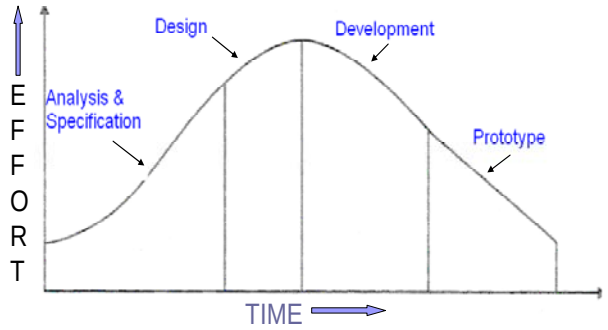

Figure 1(a): Project Effort Myth

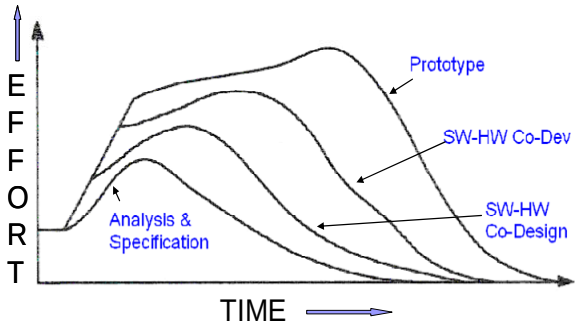

Figure 1(b): Project Effort Reality

In table 3 we have we have summarized this project reality in terms of the time spent during each phase of PDC. It can be seen from the table 3 that the final engineering time spent (in days) is about 720 for developing an embedded system for the current year.

Table 3: Product Development time with Current Design methodology

Design Phases	Percent Time Spent	Eng. Time (in Days)
Requirement Analysis	10	48
Specification	15	72
Architectural Selection	15	72
RF Design & Development	20	96
HW-SW Co-Design	15	72
HW-SW Co-Development	30	144
Hw-SW Co-Verification	30	144
Virtual Prototyping	15	72
Total	150	720

But there is always some amount of concurrency involved in design phases of PDC. For example, RF design and development can be carried out at the same time as the co-development of HW-SW. But such design concurrency is limited in its nature as these design phases are dependent upon each other. Table 4 lists our assumption on this design concurrency factor we have chosen.

Table 4: Design Concurrency Factor for next five years

Factor	1st Year	2nd Year	3rd Year	4th Year	5th Year
Concurrent Engineering Factor	1.5	1.5	1.5	1.5	1.5

We have chosen the value of this parameter to be constant as this value does not depend on the increase in complexity each year. Thus, this concurrent engineering factor must be accounted in the development time for PDC. Along with concurrent engineering factor, management and co-ordination (M&C) factor also needs to be included (by adding with the total time given in table 3) while calculating the actual product development time. This M&C factor (in table 2) is 0% for the first year, thus, will not effect the design time. Thus the total time for and embedded product development is given by equation 1.

$$Final\ \Pr oductDevelopmentTime = \frac{TotalTime + Management\ \&\ CoordinationTime}{ConcurrentEngineeringFactor} \quad \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Applying equation (1) we get the final product development time to be 480 days. We can realize that these 480 days represents a true scenario of embedded industry. This is due to the fact there are about (365-52-52-30) working days in a year. We are excluding weekends and possible leaves allocated to an engineer. Thus, a two year development time can be represented as about 465 days. This number is very close to what we have calculated from our calculation (480), thus also verifies the correctness of our methodology adopted for calculating the cost model.

We have used the above discussed methodology to extrapolate the number for next five years. These numbers are tabulated in table 5. In table 5 column TS represent the percentage of Time Spent and ET represents the Engineering time in Days.

Table 5: Predicted Product Development time with Current Design methodology

	1st Year		2nd Year		3rd Year		4th Year		5th Year	
Design Phases	**TS**	**ET**	**TS**	**ET**	**TS**	**ET**	**TS**	**ET**	**TS**	**ET**
Requirement Analysis	10	48	13	64	17	80	20	96	27	128
Specification	15	72	20	96	25	120	30	144	40	192
Architectural Selection	15	72	20	96	25	120	30	144	40	192
RF Design & Development	20	96	27	128	33	160	40	192	53	256
HW-SW Co-Design	15	72	20	96	25	120	30	144	40	192
HW-SW Co-Development	30	144	40	192	50	240	60	288	80	384
Hw-SW Co-Verification	30	144	40	192	50	240	60	288	80	384
Virtual Prototyping	15	72	20	96	25	120	30	144	40	192
Total	150	720	200	958	251	1202	300	1440	401	1922
M&C Time		0		48		120		216		384
PDT with Non-OPP		480		670		882		1104		1538

In table 5 the entries for 2nd year are calculated by using the table 1. This has been represented in form of equation 2.

$$\%TimeSpent_I = DesignPhase_I \times ConcurrenyFactor_J \quad \text{.....(2)}$$

In above equation subscript "I" represent the value in a row and subscript "J" represents the value in a column. The column values from concurrency factor (from table 1) have been multiplied with percent time spent column (from table 3) to get the final value corresponding to each design phase (table 3). The total time is given by equation 3.

$$\sum_{I=1}^{n} \%TimeSpent \quad \text{.....(3)}$$

In equation 3, the value of I=1 represent the value corresponding to requirement analysis phase and I = n represent the value corresponding to virtual prototyping design phase. Similarly, the M&C time has been calculated by equation 4.

$$M\ \&\ CTime_J = TotalEngineeringTime(inDays)_J \times M\ \&\ CFactor_J \quad \text{.....(4)}$$

In calculating M&C Time we have multiplied the corresponding total engineering time (in days) (from table 5) with corresponding M&C factor value for a year (table 2).

Final PDT for Non-OPP design methodology is calculated by using equation 5.

$$FinalPDTwithNon-OPP_J = \sum_{J=1stYear}^{5thYear} \frac{TotalEngineeringTime(inDays) + M\ \&\ CTime}{ConcurrentEngineeringFactor} \quad \text{.....(5)}$$

We have added the value of total engineering time is days (in table 5) with the M&C time spent (table 5) and divided it by the concurrent engineering factor to accommodate the overlap among various design phases in PDC. This procedure has been repeated for each year to get it final PDT for Non-OPP design methodology.

We have also calculated the final product development cost. We have represented the time of product development in terms of "Man-Hours". Man Hours have been calculated from equation 6.

$$ManHours = EngineeringTime(inDays) \times 30 \times 8 \quad \text{.....(6)}$$

For calculating the man-hours we have assumed that a team comprises of 30 engineers and each engineer works for 8 hours in a day. Similarly, we calculate the man-hours for the five year period of cost analysis. These man-hours have then been converted into the cost of product development. For calculating the cost entry in each row of table 6, we have used equation 7.

$$Cost = \frac{ManHours \times CostofEngineerPerYear}{(No.ofWorkingDaysInAYear = 365 - 52 - 52 - 31) \times (HoursPerDay = 8)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

This has been summarized in table 6.

Table 6: Product Development Cost with Current Design Methodology

	1st Year			2nd year		3rd Year		4th Year		5th Year	
	Engg. Design Time	**Man Hours**	**Cost**	**Man Hours**	**Cost**	**Man Hours**	**Cost**	**Man Hours**	**Cost**	**Man Hours**	**Cost**
Requirement Analysis	48	11520	$938,880	15,322	$1,248,710	19,238	$1,567,930	23,040	$1,877,760	30,758	$2,506,810
Specification	72	17280	$1,408,320	22,982	$1,873,066	28,858	$2,351,894	34,560	$2,816,640	46,138	$3,760,214
Architecture Selection	72	17280	$1,408,320	22,982	$1,873,066	28,858	$2,351,894	34,560	$2,816,640	46,138	$3,760,214
RF Design & Development	96	23040	$1,877,760	30,643	$2,497,421	38,477	$3,135,859	46,080	$3,755,520	61,517	$5,013,619
SW Design	72	17280	$1,408,320	22,982	$1,873,066	28,858	$2,351,894	34,560	$2,816,640	46,138	$3,760,214
HW Design	144	34560	$2,816,640	45,965	$3,746,131	57,715	$4,703,789	69,120	$5,633,280	92,275	$7,520,429
SW/HW Co-Development	144	34560	$2,816,640	45,965	$3,746,131	57,715	$4,703,789	69,120	$5,633,280	92,275	$7,520,429
SW/HW Co-Verification	72	17280	$1,408,320	22,982	$1,873,066	28,858	$2,351,894	34,560	$2,816,640	46,138	$3,760,214
Virtual Radio Prototyping	72	17280	$1,408,320	22,982	$1,873,066	28,858	$2,351,894	34,560	$2,816,640	46,138	$3,760,214
M&C Factor		0%	$0	5%		10%		20%		40%	
Total	720	190080	$1,408,320	12,640	$21,633,908	31,743	$28,457,922	57,024	$35,630,496	101,503	$49,634,830

It should be noted that the concurrent engineering factor does not affect the cost of the product development but only reduces the time of product development. This concurrent engineering factor is not included in (7) while calculating the final cost of product development.

5. Cost Model for OPP Design Methodology

We have divided the OPP cost model into SW cost model and hardware cost model. These HW/SW cost model has been further divided into component cost model, sub-system cost mode and system cost model. We are proposing a component based design methodology along with other innovations for designing future embedded systems as discussed earlier in section 2 of this paper. Such methodology will have a good amount of reuse factor. Thus, we propose to build a set of reusable components at various levels of abstractions and design phases. We will discuss design reuse parameter in more detail while describing HW and SW cost model. We further assumed that the complexity of each SW or HW component is equal to that of a digital camera in an embedded system. These components will be integrated together to form a SW/HW sub-system and these sub-systems will be combined together to form the final embedded system.

5.1 Factors for OPP Cost Model

Along with the common factors for the cost model, as discussed in previous section, we have assumed additional parameters for evolving this cost model. These parameters for methodology learning factor, concurrent engineering factor, reuse factor for software cost model, reuse factor for hardware cost model and HW and SW maintenance factor.

5.1.1. Methodology Learning Factor (MLF)

As OPP aims at introducing various new design methodologies and techniques into PDC, it will take some time for the product development team to get expertise in these areas. With time as OPP design methodology gets more matured this factor will not be dominant any more. Thus, in the beginning yeas we have represented this parameter by a bigger number and its value slowly decreases over a period of five years. This parameter gets multiplied with the final PDT increasing the cost and time to market. MLF has been shown in Table 8.

Table 8: Methodology Learning Factor

Factor	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**

Methodology Learning Factor	2.0	1.5	1.2	1.1	1.0

5.1.2. OPP Concurrent Engineering Factor

Similar to Non-OPP Design methodology, we have some design concurrent in OPP PDC as well. But due to immaturity of OPP design methodology in its recent phase of deployment, we assume that there is no design concurrency. As OPP gets more matured design bottleneck will be addressed and more design concurrency would evolve. Finally, in five years time it is expected that this parameter will have the same value as in case of Non-OPP design methodology. We have represented the value of this parameter in table 9.

Table 9: Concurrent Engineering Factor for OPP Design Methodology

Factor	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**
Concurrent Engineering Factor	1.0	1.1	1.2	1.3	1.5

This parameter gets divided by the final PDT. Thus, a larger value of this parameter will reduce PDT further. It should be noted that the concurrent engineering factor will not affect the development cost but will effect the development time only.

5.2 Software Cost and Time Feasibility Analysis for OPP Design Methodology

As discussed earlier we have proposed a component based design for embedded systems. In order to get the final cost and time analysis for embedded systems we first analyze the development time for one SW component. The SW development life cycle would include requirement analysis, specification, SW design, SW development, SW integration and SW testing & verification. We propose that each SW component will be of the same complexity as a digital camera in an embedded device. We have listed the total engineering time for developing this SW component is Table 10.

Table 10: SW Component Design Parameter

	Time (in Days)	No. of Engineers	Total Engineering Time (in Days)
Specification/Requirement	2.5	2	5
SW Design	2.5	2	5
SW Development	3	3	9
SW Integration	2	2	4
SW Verification	2.5	3	7.5
Total Component Development Time			**30.5**

In Table 10 the total engineering time has been calculated by equation 8. The total component development time is given by equation 9.

$$TotalEngineringTime(inDays) = No.ofDays \times No.ofEngineers \quad \ldots\ldots(8)$$

$$TotalComponentDevelopmentTime = \sum_{i=0}^{i=4} TotalEngineeringTime(inDays) \quad \ldots\ldots(9)$$

In (9), i=0 to i=4 represents time involved from specification phase to testing and verification phase. Using (9), we calculate the cost of component development. The cost of component is given by equation 10.

$$CostofComponentDevelopment = \sum (TotalComponentDevelopmentTime(inDays) \times CostofEngineerPerDay) \ldots(10)$$

Thus, the total cost of development of a single component is $19,892. This allows us to develop the overall cost model for SW system. For this we have further assumed some software specific parameters. These parameters are SW reuse factor, SW maintenance factor, methodology learning factor and concurrent engineering factor.

5.2.1 SW Reuse Factor

In this research, we have proposed new design techniques such as concurrency modeling, object oriented component based design, SW decomposition and layered architecture. These methodologies allow us to design reusable components independent of other components. Thus, they provide a degree of reuse. This reuse will increase as we design and develop more SW components over a period of five years. During the first year of the project there will no reuse and this parameter will be increase to 50%. Table 11 shows this value distributed over a five year period.

Table 11: Reuse Factor for Next Five Years

Factor	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**
Reuse Factor	0%	20%	30%	40%	50%

The reuse factor will obviously reduce the SW development cost and time.

5.2.2. SW Maintenance Factor

As number of SW component increases, there will be resources needed to manage and update these component libraries. SW maintenance factor represent this cost involved in managing and updating the SW library. Table 12 shows this maintenance factor

Table 12: SW Maintenance Parameter for Next Five Years

Factor	1st Year	2nd Year	3rd Year	4th Year	5th Year
SW Maintenance Factor	0%	25%	30%	35%	40%

Thus, as the SW maintenance factor increases, it will increase the cost and the time of product development.

5.2.3. Methodology Learning Factor

As per the principles of the management, it takes some finite amount of time to excel a new development process. Similarly, there will some time frame associated in adopting and exploiting OPP design methodology to get its complete potential. This factor is represented in form of methodology learning factor. Table 13 shows the predicted value of this over a period of five years.

Table 12: SW Maintenance Parameter for Next Five Years

Factor	1st Year	2nd Year	3rd Year	4th Year	5th Year
Methodology Learning Factor	4	3	2	1.5	1

Thus, as the methodology learning factor decreases over a period of five years, it will reduce the cost and the time of product development, thereby, enhancing the productivity of system design further.

5.2.4. Concurrent Engineering Factor

Similar to Non-OPP design methodology, we will be able to bring some concurrency among various design phases. We will not be able to exploit this concurrency in the beginning phase of deployment of OPP design methodology. But with time as this design methodology become matured and product development gains more expertise in this technology, we will be able develop more software components in parallel with each other. The value of this factor over a period of five years is shown in table 13.

Table 13: Concurrent Engineering Factor for Next Five Years

Factor	1st Year	2nd Year	3rd Year	4th Year	5th Year
Concurrent Engineering Factor	1	3	7	13	18

As the concurrent engineering factor increases over a period of five years, it will reduce the time of product development but will not affect the cost of the product development. The concurrent engineering factor can be accommodated by devoting more engineers in component development process but this will in turn increase the management and co-ordination time. This increase in management and co-ordination time has been incorporated in M&C factor as discussed earlier.

5.2.4. Final SW System Cost and Development Time Analysis

Final cost and development time of SW system has been calculated based on various parameters as discussed above. We have further assumed that there are about 100 SW components in a SW system. The number of these components will increase as per the system complexity factor. In addition to these components, there are some legacy components. These legacy components will have to be integrated with newly design components. We further propose that with time as we develop a set of more sophisticated libraries, these legacy components will decrease. We propose to develop SW sub-system by integrating SW components and SW system by integrating these sub-systems together. Thus, we will also have to design integration components and its interfaces. The number of integration components is given by equation 11.

$$\sum IntegrationComponents = \sum\left(\frac{NewlyDesigedComponents}{10}\right) + \sum\left(LegacyComponents\right) + 1 \ldots\ldots\ldots (11)$$

It can be realized from (11) that we propose to form a sub-system by integration 10 components. For each of the legacy component we will have to design an integration component, defining its interfaces. All these sub-systems will then be combined together to form a SW system. Thus "1" is added in (11). Total number of newly designed components is calculated by adding the integration components and the actual components. As the value of reuse factor increases from 0%

to 50% over the period of five years, we will have to develop less number of components. Similarly, we have added value of methodology learning factor, SW maintenance factor, and concurrent engineering factor in the table 14 to calculate the final development time (in Days) and cost of SW development.

Table 14: Cost and Development Analysis for SW System

	1st year	2nd Year	3rd Year	4th year	5th Year
Legacy Components	20	17	15	12	10
No. of Components	100	133	167	200	267
Reuse Factor	0%	20%	30%	40%	50%
No. of Components Reused	0	20	40	67	100
Newly Designed Components	100	113	127	133	167
Number of Integration Components	31	31	33	33	38
Total No. of Newly Designed Components	131	139	160	166	205
Total Number of Components	151	161	175	178	215
Methodology Learning factor	4.0	3.0	2.0	1.5	1.0
Development Cost	$10,423,043	$8,610,946	$6,357,261	$4,958,902	$4,071,750
Software Maintenance Factor	0%	25%	30%	35%	40%
Cost with SW Maintenance Factor	$10,423,043	$11,216,707	$8,940,545	$7,183,943	$6,055,311
No of Engineering Days	15982	17199	13709	11015	9285
No. of Days Assuming 30 Engineers	533	573	457	367	309
Concurrency Factor	1	3	7	13	18
Final Development Time(in Days)	**533**	**191**	**65**	**28**	**17**

5.3 Hardware Cost and Time Feasibility Analysis for OPP Design Methodology

Proposed hardware cost model is based on FPGA based NOC backbone for embedded systems. With rapid advancements in FPGA technology, today's FPGAs have performance comparable to the ASICs. Therefore, FPGAs may be used for designing the NOC backbone for embedded systems. Today an ASIC based NOC implementation is very expensive; it is not a feasible solution to be employed for designing embedded system. However FPGA based implementation of a NOC will provide a cost effective solution to NOC. We assumed 4 FPGA (each FPGA having two processors) based NOC implementation for the communication backbone of our embedded system. With the increase in complexity, we proposed that the number of FPGA would increase to 8 (16 processors) in five years. We assumed that hardware design phase will include specification, design capture, design simulation, synthesis, functional verification, place & route, timing/power simulation and final simulation. Considering the complexity of a component to be the same as that of a digital camera in an embedded device, we estimated that it will take about 36 engineering days to design a HW single component. Table 15 shows the hardware component development time for various FPGA design phases.

Table 15: FPGA Based Hardware Design Phase

FPGA Design Parameter	Time (Days)	No. Of Engineers	Total Time
Specification	3	3	9
Design Capture	3	2	6
Design Simulation	1	3	3
Synthesis	2	2	4
Functional Simulation	1	2	4
Place/Route in Weeks	2	3	6
Timing/Power Simulation	1	2	2
Final Simulation in Weeks	1	2	2
Total			36

Thus the total cost of development of a HW component would is given by (12), which is be around $23.5K.

$$TotalEngineringCostPerComponent = No.ofEngineeringDays \times No.ofEngineers \quad \text{.....(12)}$$

From this equation hardware cost and time analysis model has been derived. This model is discussed in table 16. In table 16, we have assumed that there are 4 sub-systems in the 1st year of hardware development phase. We further assume that an embedded system comprises of about 50 components. The number of these components will increase as per the complexity factor. In the 1st year of the hardware development phase we will have any amount of reuse factor. But this factor will increase over a period of five years. Similarly, we have added a HW maintenance factor. We have assumed the value of this factor to be constant at 25%. The number of integration components will be equal to the number of sub-systems. We have

not added the cost of integrating the sub-systems into a system at this phase. This cost is added in the final system cost model. The number of engineering days has been calculated by dividing the hardware development cost with the cost of engineering per day. For calculating the development time, we have assumed a team of 30 engineers to calculate the number of engineering days. These engineering days are then divided by the concurrency in development phase.

Table 16: Hardware Development Cost with OPP Design Methodology

Parameters	**1st Year**	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**
Number of Sub-Systems	**4**	**4**	**6**	**6**	**8**
No. of Components	50	65	82	100	133
No. of Components Per Sub-System	13	16	14	17	17
Reuse Factor	0%	30%	40%	50%	65%
No. of Components Reused	0	15	26	41	65
Newly Designed Components	50	50	56	59	68
Number of Integration Components	4	4	6	6	8
Total Number of Newly Designed Components	54	54	62	65	76
Development Cost	$1,267,826	$1,267,826	$1,455,652	$1,526,087	$1,784,348
HW Maintenance Factor	0%	25%	25%	25%	25%
HW Maintenance Cost	$0	$316,957	$316,957	$363,913	$381,522
Final Hardware Development Cost	$1,267,826	$1,584,783	$1,772,609	$1,890,000	$2,165,870
No. of Engineering Days	1944	1944	2232	2340	2736
No. of Days Assuming 30 Engineers	65	65	74	78	91
Concurrent Development	1.0	3.0	7.0	13.0	18.0
Final Development Time	65	22	11	6	5

From table 16, we drive the final cost analysis for designing 1million embedded product. For this we have assumed the cost of a single unit of FPGA in the first year to be $13.00. In can be realized from the FPGA industry that the cost of an FPGA reduces approximately at the rate of 25% every year. If these FPGAs are further hardwired, which can be an optimum solution if the product becomes successful, then it will further reduce the cost of FPGA by an additional 75%. Thus, the total cost of hardware can be calculated by adding the cost of the hardware development (from table 16) and the cost of the hardware. This has been summarized in table 17.

Table 17: Final Hardware Production Cost with OPP Design Methodology for 1M Units

	1st Year	**2nd Year**	**3rd Year**	**4th Year**	**5th Year**
No. of Units of Final Product	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000
No. of FPGA Used	4	4	6	6	8
Cost Per FPGA	$13.00	$9.75	$7.31	$5.48	$4.11
Cost of FPGA After Hardwire	$13.00	$2.44	$2.44	$2.44	$2.44
Total Cost of FPGAs in One Product	$52.00	$9.75	$14.63	$14.63	$19.50
FPGA Cost in 1M Products	52,000,000	9,750,000	14,625,000	14,625,000	19,500,000
Final Hardware Development Cost	$1,267,826	$1,584,783	$1,772,609	$1,890,000	$2,165,870
Total Hardware Cost	53,267,826	11,334,783	16,397,609	16,515,000	21,665,870

It can be realized from table 17 that the final development cost reduces drastically during the 2nd year. This is due to the cost effectiveness of the actual hardware from hardwiring the FPGAs.

5.4. System Cost Model

System design flow involves requirement analysis, specification, architectural selection, RF design & development, SW-HW co-design, SW-HW co-development, SW-HW co-verification and virtual prototyping phases. All of these steps are not strictly sequential in nature and some of them can be seen as overlapping stages.

In OPP design methodology, we propose to invest more time at requirement analysis, specification and HW-SW co-design phase. Concepts such as, executable specifications, SysML system level modeling, abstract concurrency modeling, and annotation methods, as discussed earlier will be included at these levels. Most of these concepts have been discussed in section 2 of this paper. Usually the specifications are interpreted from the requirement document, which is more or less a simple word document. Often these requirements are misinterpreted. This results in a huge penalty in terms of cycle time. Similarly there is always a boundary between HW and SW engineering. Thus, the co-design and co-development phases become the bottleneck. SystemC- A HW-SW design language is likely overcome these limitations. The concept such as test driven design will ensure the unit testing before the actual development phase, thereby reducing the time spent during the unit testing. Concurrent modeling before the design phase ensures the proper integration of components, thus reducing the

time spent during the integration phase. Table 18 lists the various design phases and the proposed time spent over the period of five years. The time spent each year on a particular design phase increases by the complexity factor.

Table 18: Product Development Cycle with OPP Design Methodology

	1st Year	**2nd Year**	**3rd year**	**4th year**	**5th Year**
Requirement Analysis	20	27	33	40	53
Specification	15	20	25	30	40
Architecture Selection	10	13	17	20	27
RF Design & Development	20	27	33	40	53
SW/HW Co-Design	60	80	100	120	160
SW/HW Co-Development	15	20	25	30	40
SW/HW Co-Verification	15	20	25	30	40
Virtual Radio Prototyping	15	20	25	8	40
Total	**170**	**226**	**284**	**318**	**454**

It can be realized from table 18 that we propose to spend more time in requirement analysis and SW/HW co-design phase. From OPP SW development cost model and OPP HW development cost model (table14 and table 16 respectively) we know that time for developing a SW system is about 533 days and time for developing a HW system is about 65 days. So the total time for developing the SW/HW will be the greater of the two values. Thus, we assume that the final time for HW/SW design will be 533 days in the 1st year. This time is corresponding to 60% of the product development time as shown in table 18. Similarly, for the second year the 80% of the time spent for designing SW HW (as shown in table 18) will be corresponding to 191 days from table 14. We can then drive time spent in other design phases for the system development life cycle by using this 60% as a reference value and equating it with their corresponding percentage of time spent. These results have been summarized in table 19.

Table 19: System Development Time for OPP Design Methodology

	1st Year	**2nd Year**	**3rd year**	**4th year**	**5th Year**
Requirement Analysis	177.58	63.70	21.76	9.41	5.73
Specification	133.18	47.77	16.32	7.06	4.30
Architecture Selection	88.79	31.85	10.88	4.71	2.87
RF Design & Development	0.00	0.00	0.00	9.41	5.73
SW/HW Co-Design	532.73	191.10	65.28	28.24	17.19
SW/HW Co-Development	133.18	47.77	16.32	7.06	4.30
SW/HW Co-Verification	133.18	47.77	16.32	7.06	4.30
Virtual Radio Prototyping	133.18	47.77	16.32	1.77	4.30
M&C Time	0.00	23.89	16.32	14.95	19.49
Total	1331.83	477.75	163.20	74.73	48.72
Methodology Learning Factor	2	1.5	1.2	1.1	1
Time After Methodology Learning factor	2663.66	716.62	195.84	82.20	48.72
Concurrency Factor	1	1.1	1.2	1.5	2.0
Time After Concurrency factor	2663.66	651.48	152.32	62.33	33.0
Management & Co-ordination (M&C) Factor	0%	5%	10%	20%	40%
Final Development Time	2663.66	684.05	167.55	74.80	46.1

In table 19, we show the total time as the summation of the time spent during each phase. As discussed earlier the concepts such as methodology learning factor, concurrency factor and M&C factor will impact the development time. The value of these factors were then included in the development time in corresponding years. This gave us the final development time for OPP design methodology.

Using data from table 19 and cost calculations for software and hardware development with OPP design methodology, we are able to calculate the cost of system design using OPP design methodology. We summarize these results in table 20.

In table 20, we have separated the SW design and HW design. This is due to the fact that these phases can be done at the same time in parallel with each other, thereby reducing the time of system development. But, while calculating the cost we will have to include the resources dedicated in SW/HW design phase. We have calculated the cost corresponding to each design phase with SW design as a reference. From table 18 we know the percentage of resources dedicated to each design phase in every year. So using the value of SW design as a reference we have calculated the cost of other design phases.

Table 20: Final System Development Cost with OPP Design Methodology

	1st Year	2nd Year	3rd Year	4th year	5th Year
Requirement Analysis	$3,474,348	$3,738,902	$2,980,182	$1,524,362	$2,018,437
Specification	$2,605,761	$2,804,177	$2,235,136	$1,143,271	$1,513,828
Architecture Selection	$1,737,174	$1,869,451	$1,490,091	$762,181	$1,009,218
RF Design & Development	$3,474,348	$3,738,902	$2,980,182	$1,524,362	$2,018,437
SW Design	$10,423,043	$11,216,707	$8,940,545	$4,573,085	$6,055,311
HW Design	$1,267,826	$1,584,783	$1,772,609	$1,890,000	$2,165,870
SW/HW Co-Development	$2,605,761	$2,804,177	$2,235,136	$1,143,271	$1,513,828
SW/HW Co-Verification	$2,605,761	$2,804,177	$2,235,136	$285,818	$1,513,828
Virtual Radio Prototyping	$2,605,761	$2,804,177	$2,235,136	$285,818	$1,513,828
Total	$30,799,783	$33,365,451	$27,104,152	$13,132,168	$19,322,584
Final Cost After Methodology Learning Factor	$61,599,565	$50,048,177	$32,524,982	$14,445,385	$19,322,584

6. Results

We developed the entire cost model for three different scenarios: best, average and worst cases. The best case assumed more optimistic parameters in terms of the number of components being used and the steepness of the curves for methodology learning and maturity factors. In the result section we are providing the results for the worst case scenarios.

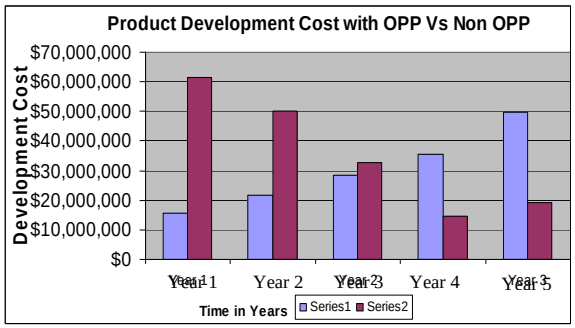

Figure 2: Cost Comparison for OPP Vs Non-OPP Design Methodologies

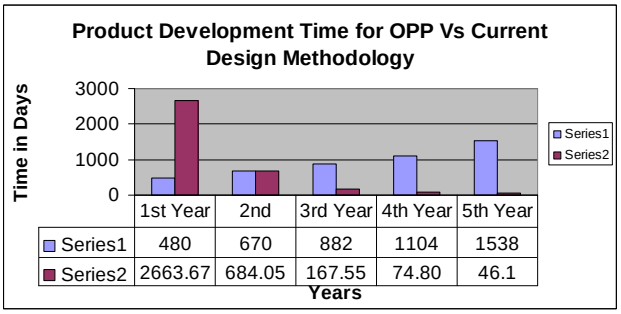

Figure 3: Development Time Comparison for OPP Vs Non OPP Design Methodologies

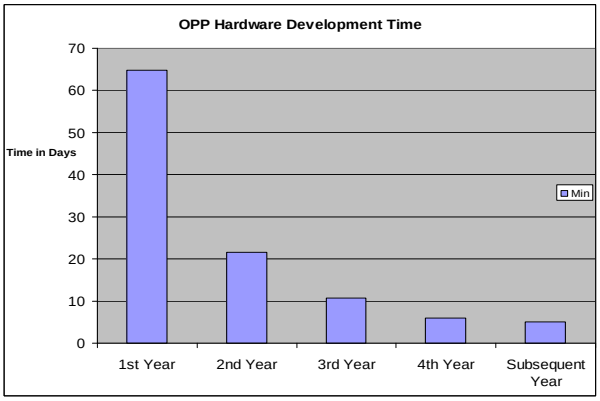

Figure 5: OPP Hardware Development Time

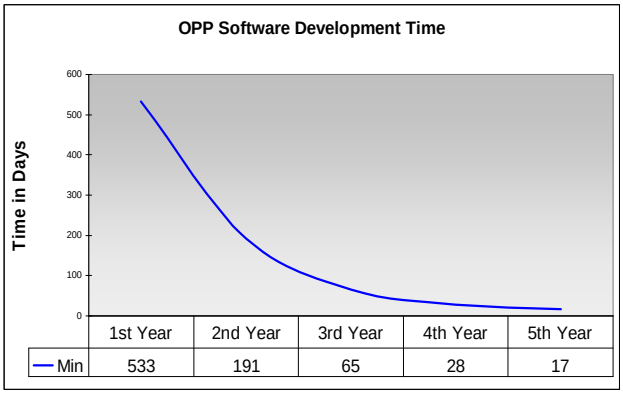

Figure 6: OPP Software Development Time

Figure 2 shows the cost comparison between the OPP and Non-OPP design methodologies. It can be seen from the figure that the cost of the OPP PDC will be more than that of the Non-OPP PDC in the first two years. This is due to the fact that in the initial two years OPP would invest more time in evolving HW/SW components and there will be a methodology

learning factor involved in using the proposed design methodology. But the value of the product development decreases giving the long term benefit. Similarly, the time of product development with OPP design methodology is about five times as compared to the current design methodology. But over the period of five years it will be ten times less than the time for product development with the current design methodology. From figure 5 and 6 we can realize that the time of hardware and software development will drastically reduce over the period of five years with OPP design methodology respectively.

6. Future work

We are working on developing and expanding this cost model to integrate RF components more fully. During this process, we expect to develop a refined methodology for design, development and integration of RF components.

7. Conclusion

We have developed a model to explore benefits of increased automation, integration, annotation, and reuse. Our results show a significant reduction in cost and time-to-market with our proposed design methodology. This may be viewed in light of a 40 fold improvement in productivity in hardware design that accrued, during 1990 to 2005, with the use of similar techniques [2].

8. Acknowledgements

We would like to acknowledge the help of a former MS student at Florida Atlantic University, Abhijit Ajmera, now with Motorola. This work was funded by iDEN, Motorola.

9. References

[1] Boehm, B.W., et al.., Software Cost Estimation with COCOMO II, Prentice Hall, 2000
[2] Kahng, A.B., Design and Design Automation Advances in the 2001 ITRS, MEDEA+ conference, 2002