Original Article

INNOVATIONS IN ENGINEERING AND TECHNOLOGY DRIVING ELECTRONIC DESIGN AUTOMATION

Solomon Chijioke Okafor

Department of Electronic & Computer Engineering, Faculty of Engineering, University of Port Harcourt, Nigeria

DOI: https://doi.org/10.5281/zenodo.17131957

Abstract: Computer Aided Design (CAD) also known as Ele ctronic Design Automation (EDA) is used for the design of microelectronic circuits and employs layers of abstraction. This innovation has been in place since the introduction and invention of Integrated Circuits (IC). It has had an extraordinary effect on everyday human life with the development of conveniences in the software and Hardware system design in cell Phones, Global Positioning Systems (GPS), navigation systems, music players, and personal data assistants (PDAs). CAD techniques have reduced the level of complexity within a given layer of abstraction that is limited due to limitations of human senses. Certain Design rules such as Maximum loading of a signal, minimum space or maximum mask overlap during fabrication to avoid shorts or achieve connection and set up / hold time for some pre-designed blocks. Event Driven Simulation (EDS) is often as a modeling system containing multiple agents that interact with discrete events. In fact, almost everything and every daily task have been influenced by, and in some cases are direct results of EDA/CAD. As engineers, the role of EDA/CAD cannot be overemphasized as there has been significant and exponential growth of circuit capacity with respect to critical challenges for EDA/CAD. The most noteworthy inventions have been the microprocessor and the personal computer (PC), their progression in terms of performance and features, and the subsequent development of smaller, portable implementations such as the notebook computer. In this article, the role of EDA/CAD are pointed out in Engineering and Technology the role played in Designing and Developing Software Components (SWC), Hardware Components (HWC) and Interfacing Components (IFC) for easy Electronic Design Automation Adventures. As a result, the computer has become an essential tool and part of everyday life – to the extent that current automobiles, including safety features in particular, are controlled by the multiple microprocessors and ICs.

Keywords: Electronic Design Automation, System Design Methodology, Synthesis, CAD, Engineering, Technology.

Introduction

Design automation or computer-aided design (CAD) for microelectronic circuits has emerged since the creation of the integrated circuits (IC). It has played a crucial role to enable the rapid development of hardware and

software systems in the past several decades. CAD techniques are the key driving forces behind the reduction of circuit design time and the optimization of the circuit quality. Meanwhile, the exponential growth of circuit capacity driven by the Moore's law1 prompts new and critical challenges for CAD techniques. In this chapter we will introduce the fundamentals of design automation as an engineering field. We begin with several important processor technologies and several existing IC technologies. We then present a typical CAD flow covering all the major steps in the design cycle. We also cover some important topics such as verification and TCAD. Finally, we introduce some new trends in the design automation field.
(Deming Chen, dchen@illinois.edu)
Design Automation of Electronic Systems or Electronic Design Automation (EDA) creates software tools for Computer Aided Design (CAD) of electronic systems like the printed circuit boards (PCBs) and integrated circuits (ICs). According to the International Technology Roadmap for Semiconductors, the IC technology scaling driven by the Moore's Law will continue to evolve and dominate the semiconductor industry for at least another 10 years. This will lead to over 14 billion transistors integrated on a single chip in the 18nm technology by the year 2018 (http://www.itrs.net). EDA can work on digital circuits and analog circuits. It is the engineering science that derives software and hardware tools for the designs of integrated circuits and systems based on abstraction, design methodologies, and software implementations of sophisticated algorithms for verification and synthesis. What makes EDA unique is the continuous interplay between theory and applications. Computer Science, mathematics, and physics offer the foundations upon which EDA rests. However, EDA specialists must also be able to leverage their application domain knowledge to solve abstract problems that are well known to be intractable in general. Indeed, EDA is a cornucopia of interesting problems, some solved and some that are just appearing on the scene where prominent results come from the tight interaction of mathematicians, computer scientists, physicists, and engineers. The relevance of EDA is also proven by the existence of a vibrant community of EDA companies and universities active in this field. It is believed that the future has much to offer to young researchers and engineers. The Goal of this special issue is two-fold: First, to provide an overview of and a perspective on the evolution of EDA and then, to offer a perspective on some of the principal avenues of future development.
EDA started as a field in the 1960s when researchers of some leading academic and industrial labs conceived the first computer-aided design (CAD) tools for supporting engineers in the analysis and layout of circuits and boards whose complexity was growing dramatically. Since the 1940's, there have been significant milestones in the IC industry, for example, in 1947 Bardeen a Brattain & Shochly invented the Transistor, SONY introduced the first ever transistor-based radio in 1952, IC's was then invented by KILBY in 1958. Also in 1971 intel announced 4-bits 4004 microprocessors (2250 transistors) and in 1991 ARM introduced its first embeddable RISC IP core (chip less IC design) and today over 30 million transistor are produced per person (1 billion/person by 2008) and many more billion through the EDA.
Electronic Design Automation (EDA) has been an immensely successful field, to manage the exponential increase in our capability to implement integrated circuits (ICs) that currently incorporate billions of Transistors. At the same time, a fostered and used theories in computation and modeling, successfully combining theory and practice. EDA has completely transformed the way that electronic engineers design and manufacture integrated circuits. It was one of the earliest to engage in interdisciplinary collaboration, where the computer scientists and engineers in EDA successfully collaborated with electrical engineers, physicists, and chemists, theoretical computer scientists, applied mathematics and optimization experts, and application domain specialists.
This reports contains an overview of the EDA area, its funding history, a discussion of some major challenges for the future, related emerging technologies and how EDA experience may help in developing these technologies, Educational aspects and challenges, EDA's relation with CS theories and how this collaboration can be

resurrected and finally a series of recommendation on what might be done to promote EDA and help with the serious challenges it faces in the future.

The Role of Engineers in Electronic Design Automation (EDA)

Engineers are solely concerned with design of systems but not limited to it as they can also be part of implementation and maintenance as the need arise. Since the advent of what could be best called modern engineering, there has been tremendous development and innovation due to breaking up special field into smaller field for easy research and development. For instance in electrical engineering is broken into electrical, electronic, telecommunication, radio, computer, satellites communication etc. Societies and or organizations of engineers from the early engineering field was first formed in England and then USA whose goal was to enhance and promote research and development of engineering e.g. in 1771, 1828, 1846, 1871 and 1880 sees the formation of The Society of Engineers, Institute of Civil Engineers, Institute of Mechanical Engineers, Institute of Electrical Engineers and Institute of Chemical Engineers respectively. All these achievement is aimed at improving the quality of life and makes life easier. The nineteenth witnessed a remarkable advancement in the growth of engineering field as it gives rise to classification and specialization in the engineering profession. It create special field like Mining, Civil, Electrical, Mechanical, Chemical etc. presently, this new era has witnessed sub-field like Robotic, Environmental, System, Software Genetic, Mechatronic and ICT Engineering respectively (Oko & Abam, 2012) . Engineers are those that received professional training in the university in any field of engineering and licensed to practice engineering in their field. Electrical / electronic engineers are engineers who utilizes non-linear and active electrical components such as semiconductors devices especially transistors, diodes and integrated circuits to design electronic circuits devices, microprocessors, microcontrollers and other systems. Electronics is a subfield within the wider electrical engineering academic course but denotes a broad engineering field that covers subfield such as analog electronics, digital electronics, consumer electronics, embedded system and power electronics.

Electronics engineering deals with implementation of application, principles and algorithms developed within many related fields for examples solid state physics, radio engineering, telecommunications, control systems, signal processing system engineering, computer engineering, instrumentation engineering, electric power control, robotics and many others. The institution of electrical and electronic engineers (IEEE) is one of the most important and influential organizations for electronics engineers.

The Role of Technologists in Electronic Design Automation (EDA)

Electronic technology is intricately –woven into many sectors of industry which affects our daily lives. Every year, new and executory communications in wired, wireless and satellite services impact devices and machines which change the way people lives, work and play. It’s a dynamic environment that requires professionals to sustain its progress. Technicians or technologists are skilled persons in the engineering field by venture of formal education to analyses and design some engineering components and technological processes.

Wired phone and cable TV, cellular, broadband, mobile internet and satellite TV are all impacted by electronic engineering technology. The investment in automated manufacturing also is changing the demands for skilled workforce. Increasing demand for these services and create the need for technicians with skills to assist these growing sectors of the world economic. Engineering technologists play a critical role, serving as a nexus between engineers and technicians. From conception to design, development, testing and production, they are essential to the production process. In general, the distinction between an engineers and technologists emanates primarily from differences in their education. Engineering programs are geared toward the development of conceptual and design skills whereas engineering technology programs are oriented toward the application of designs. The balance between theory and practice gives physical insight and understanding, a balance between the world of the engineers and the world of the technologists. So engineers and technologists are the primary engine of economic growth and provide the key to unlocking any country's potential. Hence countries that want to grow or develop must invest significantly in science and technology. This is achieved by developing the talent, the human capacity required to complete in a globally competitive world.

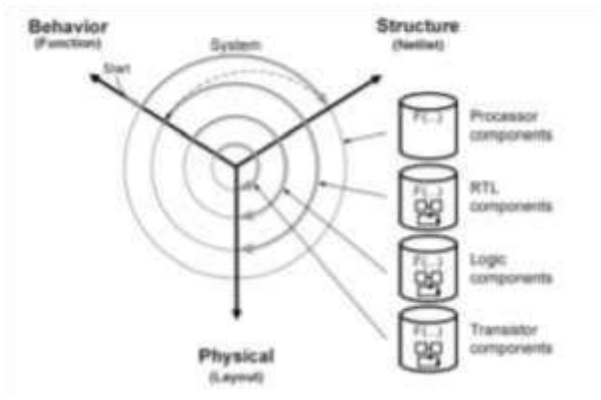
EDA

The current Wikipedia definition of EDA is "the category of tools for designing and producing electronic systems ranging from printed circuits boards (PCBs) to integrated circuits. This is sometimes referred to as ECAD (Electronic Computer-Aided Design) or just CAD. Let us emphasize the following three aspects of EDA:

- EDA consists of a collection of methodologies, algorithms and tools, which assists and automate the design, verification and testing of electronic system.
- It embodies a general methodology that seeks to successively refine a high-level description to low-level detailed physical implementation for designs ranging from integrated circuits (including system-on-chips), to printed circuit boards (PCBs) and electronic systems.
- It involves modeling, synthesis, and verification at every level of abstraction (Chen et al 2008 & Liu 2007).

The second and third aspects of EDA in this definition can be applied easily to different application fields, other than the designing of electronic systems. Design methodologies have evolved together with manufacturing technology, design complexity, and design automation. Improvements in technology have increased design complexity to the point that designers are no longer capable of making complex designs manually. To solve this problem, design automation tools, also known as computer-aided design (CAD) tools, were introduced. In order to make CAD tools more efficient and design algorithms more manageable, design-automation researchers as well as tool developers were forced to introduce more stringent design rules, parameterize components and minimize component libraries (Chen 2006 & Chang 2004). As design complexities continued to increase, tool developers created new design abstraction levels and tried to use the same design strategy from the circuit level, to the logic level, to the processor level, and finally to the system level (Gajski et al 2009). Some of the system design methodologies are shown in Fig 1. They have libraries of transistors, RTL, and Processor components. Components in each of the libraries are used to build the components in the libraries in the next abstraction level. On the circuits level, transistors are used to develop circuits and there layouts for the basic logic components such as gates, flip-flops, bus drivers etc. The components become standard cells with their functionality; structures and layout are stored in the logic component library for use in the logic level in Fig 1a. File register-transfer components are created on the logic level including registers, register files, ALU, multipliers and components for processor micro architecture that implements Boolean expression like FSM and FSMD models. The Bottom-Up model started before CAD tools invention. It is important to note that the Bottom-Up and the Top-Down model do not perform a component or system layout not until the entire design is finished. A Top-down methodology begins MoC and generates from it a system platform or system structure in which all component has its parameters and its required metric values defined but not its structures or layout shown in Fig 1b. The Meet-in-the-middle methodology [124, 160] is mostly used by designers today just to take advantage of the benefits of both Bottom-up and top-down methodologies while also minimizing the draw backs. The meet-in-the-middles methodology applies a top-down methodology to higher abstraction levels and the bottom-up methodology to lower abstraction level [124,160]. The difference between these two models is based on how the styles meet. Another possibility for a meet-in-the-middle methodology would be to perform system layout with logic components or standard cells, as shown in Fig 1d. As with the first meet-in-the-middle methodology as described, this one starts with a MoC and synthesizes the system platform with virtual Pes and CEs. Those PEs and CEs are then synthesized with RTL components, which themselves are further synthesized with logic components. In the platform methodology,

it is much product oriented [165]. System design usually starts with an already-designed platform, usually one defined by a well-known platform supplier or defined locally inside the company as shown in
Fig1. Such platforms may have already some standard components, such as memories and standard processors with well-defined layouts. The system platform may also be upgraded with the addition of custom components that will be synthesized with processor and RTL synthesis tools, after which the layout of these custom components can be obtained through standard cells. The platform methodology can be upgraded to a system-level methodology by introduction of standard architecture cells and re-targetable compilers. An
Architecture cell contains a parametrize-able programmable processor. The parameters include number, type and size of components, component connectivity, and the number of pipeline stages in the functional units, controller and the data-path. Such a standard architecture cells can be pre-synthesized with standard cells and inserted into the library of Processor components or generated on demand. A typical system-level
Methodology based on such architecture cells is shown in Fig 1f. Field-Programmable-Gate-
Array (FPGA) methodology is based on the
FPGA substrate, which consists of a multitude of 4-bit ROM cells called Lookup Tables (LUTs). These LUTs can implement any 4-variable Boolean function.
Therefore, in this methodology, every RTL component in the RTL component library must be decomposed into these 4-variable functions. Then, the Processor components are synthesized out of available RTL components shown in Fig 1g. In other words, a FPGA methodology shown in Fig 1g., uses a top-down methodology on both the System and Processor levels, in which standard and custom PEs and CEs are all expressed in terms of LUTs. A system design starts by mapping an application onto a given platform and then synthesizing custom components down to RTL components which are defined in terms of LUTs. Standard processors components in the Processor library are already defined in terms of LUTs. Once all components in the platform are defined, we flatten the whole design to LUTs and BRAMs and perform the placement and routing with the tools provided by FPGA suppliers.



Original Article

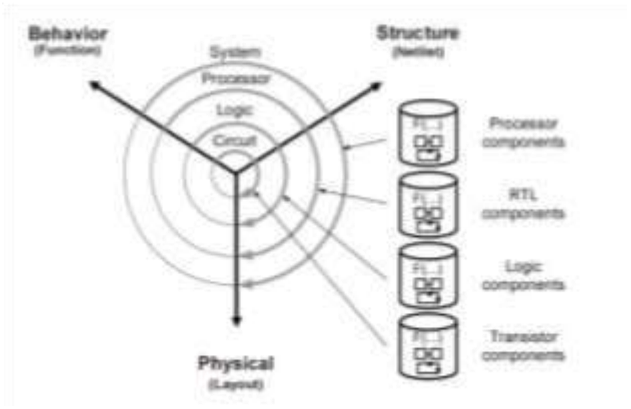


Fig 1a Bottom-up methodology.

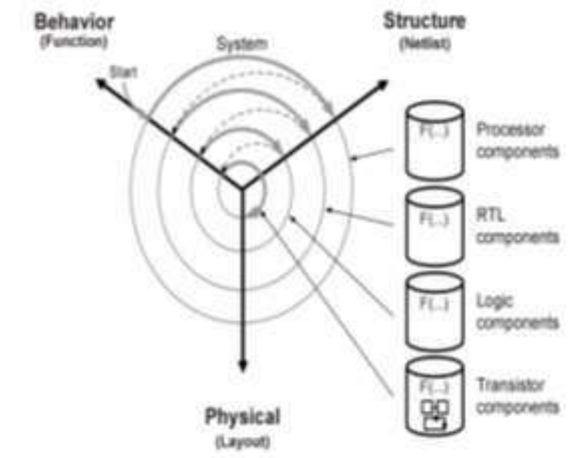


Fig 1b Top-down methodology.

Fig 1c Meet-in-middle methodology (option 1).

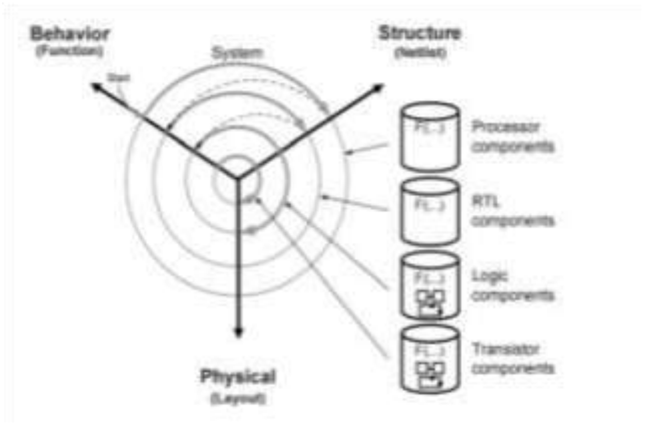


Fig 1d Meet -in-middle methodology (option 2).

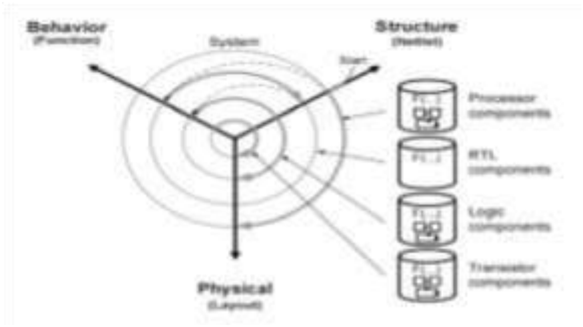


Fig 1e, Platform methodology.

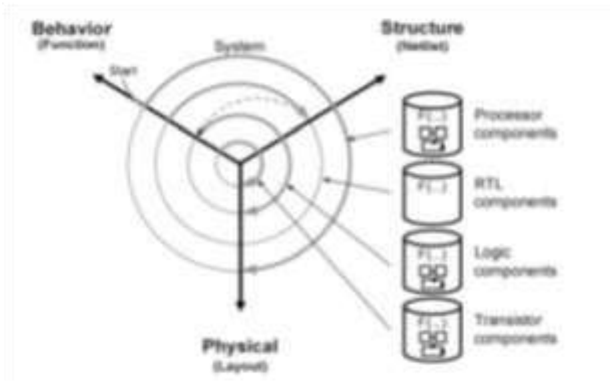


Fig 1f System methodology.

Original Article

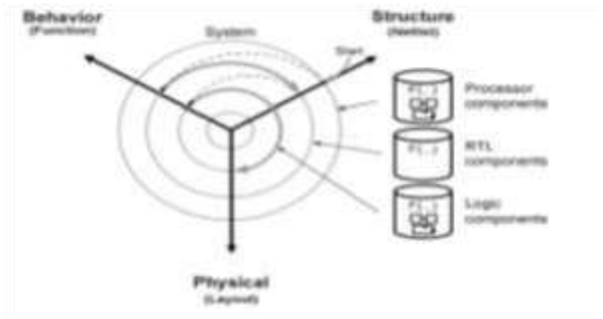


Fig 1g FPGA methodology.

The system level synthesis and the processor synthesis layout are shown in Fig 2a and 2b. As shown in Fig 2a, system-level synthesis starts with an application written in some MoC such as a set of sequential and parallel processes communicating through message-passing channels. Such a MoC must execute on a platform of multiple standard and custom processors connected through an arbitrary network. This type of platform can be defined partially or completely after estimating some characteristics of the application in terms of performance, cost, power, utilization, configurability, and other considerations. Platform definition can be done manually or automatically. Once the platform is defined, an application must be partitioned and each partition assigned to a processor or IP in the platform. In order to verify that the application executes on the platform and satisfies all the requirements, we need to generate a simulate-able and possibly verifiable model such as a timed Transaction-Level Model (TLM). After simulation, the design can be optimized if it does not satisfy the requirements by changing the platform, the application code, or the algorithms used in that code. We can also change the mapping of the application to the platform. For example, we can minimize external communication by grouping heavily communicating processes and assigning the whole group to one processor. It is also possible to assign performance-demanding processes to different processors or specialized IPs, or to pipeline performance-demanding processes if possible. After obtaining a satisfactory application code, platform, and mapping, each component can now be synthesis. Three types of component such as the software components (SWC), Hardware component (HWC) and the interfacing component (IFC) are needed for scheduling of processes, communication and Interfacing across all platforms. In the Processor Synthesis, the components are synthesized as standard processors, custom processors, and custom hardware units, which are sometimes called IPs. The standard and custom processors are usually defined by their instruction sets. Custom processors can be also defined by the algorithm or the programming language code that they execute. They are programmable so that new algorithms and the code can be added or existing one modified. Custom hardware units or IP are usually not programmable. They are used as accelerators to execute special functions for a particular application, such as multimedia applications as shown in Fig 2b. The synthesis process starts with a given Specification in a programming language, which is compiled into some Tool model such as CDFG or a FSMD or a three-address code. This formal model can be used for Estimation of the future processor architecture and its metrics. It can be also used for some partial or complete allocation, binding, and scheduling. Processor synthesis, sometimes called High-Level Synthesis (HLS), takes the formal model and performs Allocation, Binding and Scheduling. The Allocation task selects necessary and sufficient components from the RTL component library and defines their connectivity. The Binding task defines binding of variables to registers, register files, and memories, operations to specific functional units and register-to-register transfers to specific buses. Scheduling assigns operations and register transfers to clock cycle (Gajski et al 2009).

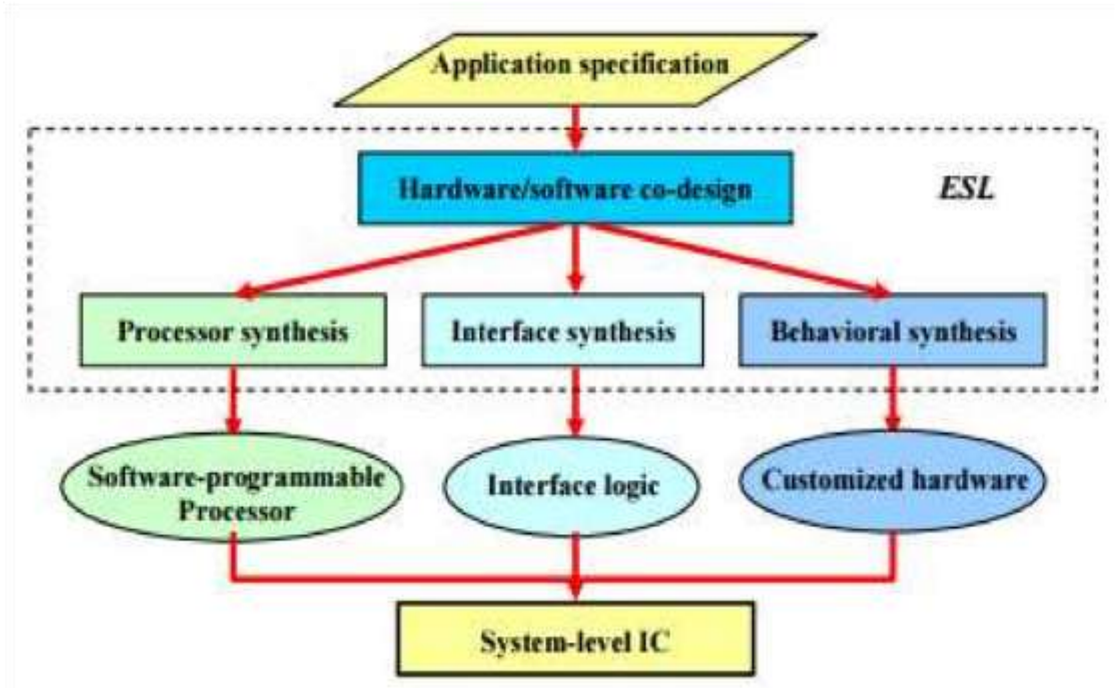


Fig 3. ESL (Electronic System-Level) design flow.

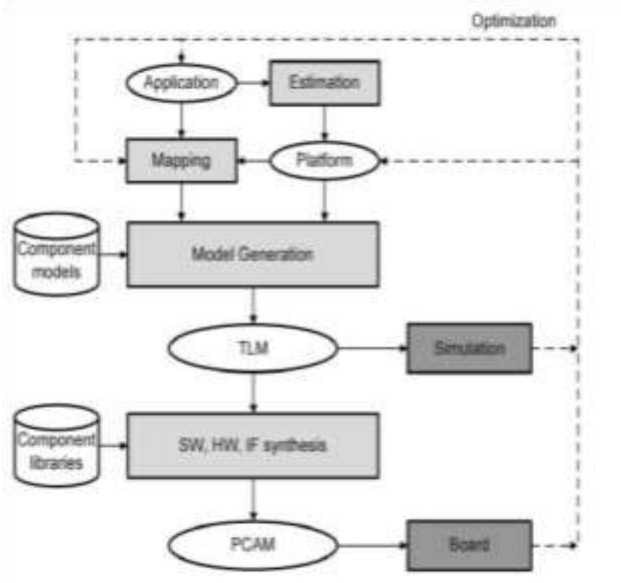


Fig 2a System-Level Synthesis

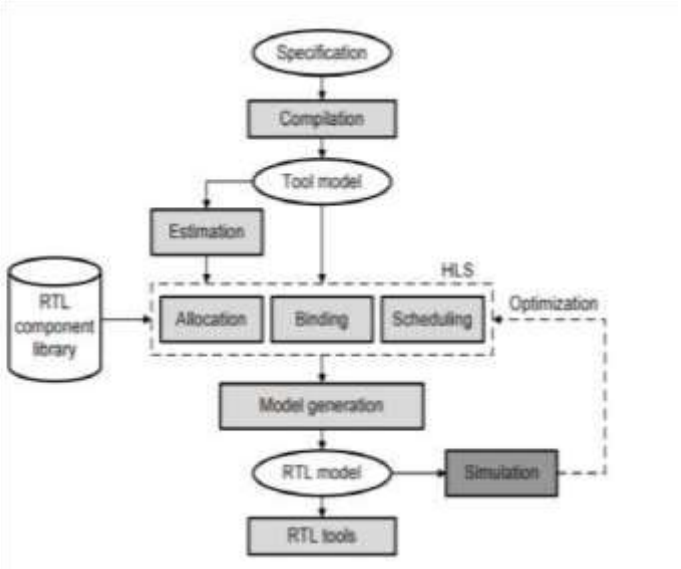


Fig 2b Processor Synthesis.

Original Article

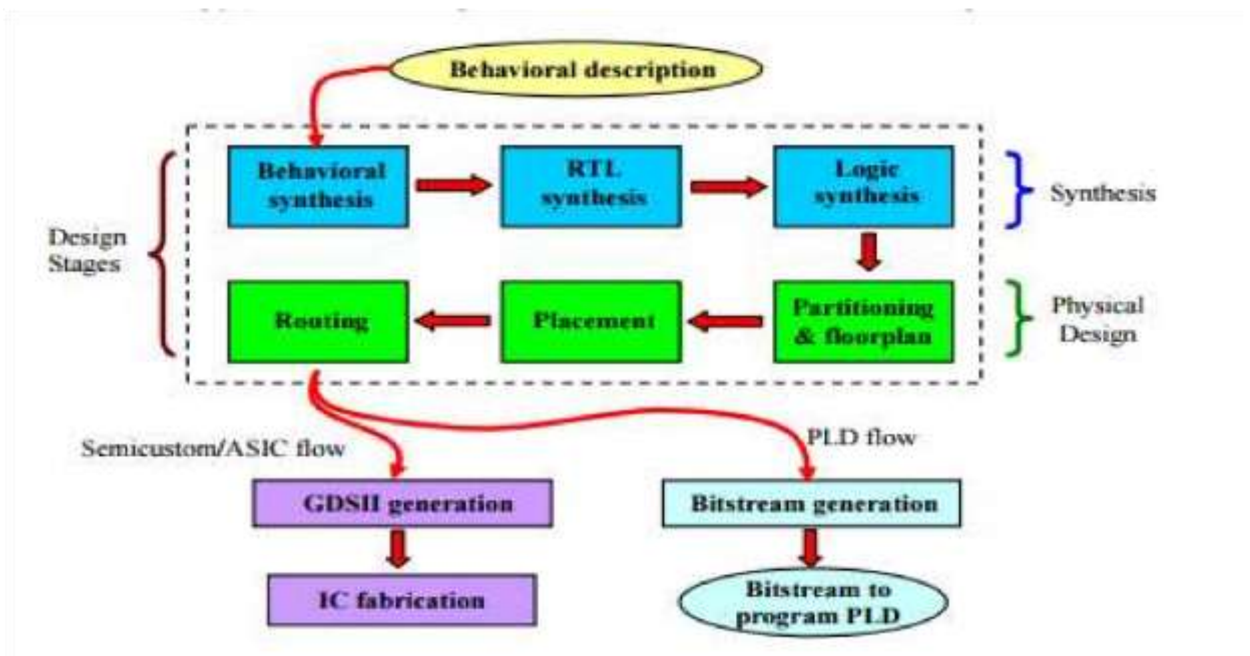


Fig 4. Atypical design flow.

Fig 3 and Fig 4 Shows the ESL design flow and atypical design flow. It includes the Behavior synthesis used for mapping of a behavioral description of a circuit into cycle-accurate RTL design that consists of a data path and a control unit. A data path is composed of three types of components called the functional unit, storage units and interconnection units. The RTL synthesis performs optimizations on the register-transfer-level design. While the logic synthesis is used for the generation of structural view of the logic-level implementation of the design. The partitioning and floorplan is in the physical design with many stages including partitioning, floorplan, placement and routing. Partitioning is required for multi-million-gate design. As a result, placement is a key step in the physical design flow for determining the positions of physical object like the logic blocks and logic cell up to millions objects and the shape of each object is predetermine and fixed. Immediately after placement is the routing stage which determines the geometric layouts of the nets to connect all the logic blocks and logic cells together. Verification and simulation is then done to ascertain the workability of the system under the design process and the testing and commissioning is performed.

Evolution of EDA

Although the history of design automation algorithms such as Kaminghan and Lin's bipartitioning heuristics and Kenneth Hall's r-dimensional quadratic location procedures predates the VLSI era, it was not until the early 1980s that the mystique surrounding VLSI chip design and fabrication was unveiled be Mead and Conway. This led to the cultivation of VLSI education in universities and to the flourishing of research in VLSI system design and electronic design automation (EDA) which, in turn, created automation tools for logic synthesis, layout generation, circuit simulation, design verification, reliability, modeling, chip testing, debugging, and yield analysis (Tan & Shi 2003).

One way of characterizing the advancement of VLSI automation tools is to demarcate them into different eras depending on their chief optimization criterion- namely, area, and timing, power, reliability, and Nano-scale issues. In the early 1980s, silicon area was at a premium, and the underlying theoretical foundation of layout tools was rooted in graph theory (min-cut, quadricsectioning), convex optimization (geometric programming), physical laws (force-directed relaxation methods), circuit theory (energy minimization in resistive networks), and spectral methods (eigenvalue-based partitioning). In the mid-1980s, stringent timing constraints motivated the development of theoretical underpinning for interconnect delay models by using linear algebra and matrix (differential quadrature method, finite difference), frequency domain solvers (method of Characteristics, FDFD), and other theoretical techniques.

In the early 1990s, energy optimization requirement led to the adaption and extension of finite element methods (FEM) and gridded techniques to solve large second-order PDE associated with the heat equation. The need for integration of full-chip thermal analysis with chip layout packages was realized by applying heat Fourier and discrete cosine transformation techniques to accelerate the multi-layer Green's function solvers over the quadruple integral spaces. In the late 1990s, higher reliability and quality assurance demands led to the development of technologycentric EDA tools to ensure correctness of the system design by using Model checking, SAT solver, static analysis, and other mainstream CS theories. Multi-scale/multilevel optimization techniques also received a lot of attention in the late 1990s, especially in physical design, to cope with the rapid increase in design complexity.

Positive Effects of EDA

The positive effects of EDA include:

- EDA will not go away and cannot stagnate. It is absolutely necessary for the support of the design of complex systems such as micro-chips. EDA expertise is needed by EDA software companies and by large system-design houses. Start-ups will continue to be valuable in finding niche applications and researching solutions, and nurturing core EDA technologies as well as emerging ones. These should see a resurgence as the economy improves.
- Cooperation between industry researchers and developers and university faculty and students remains very high, probably among the highest among any discipline within computer science/engineering.
- As technology shrinks, the problems get harder, so not less but more EDA activity is required. This increased complexity puts more emphasis on modeling, model reduction, abstraction, etc. – techniques in which expert EDA personnel are well versed.
- EDA engineers are well paid, apparently better than most other types of engineers. Moreover, if the trend of fewer students entering the field continues, then supply and demand will assert itself; demand and salaries will increase.
- EDA training in its various disciplines, including complex and large problem solving, will be valuable as new growth areas come into play – such as Nano-technologies, biology and other life science application.

Negative Effects of EDA

The negative effects of EDA include:

- Currently, many EDA companies are hurting financially, and job opportunities are down.
- Venture Capital for Start-ups in EDA has decreased significantly. These have been a vital component of EDA and have served as major centers of research and development and employment of PhDs.
- Faculty positions in EDA are tight, aggravated by the difficulty of obtaining funding to support research and students.
- Student interest in EDA as a career has decreased in recent years.
- There are reduced industrial research efforts in EDA with the dissolution of high-quality research groups at Cadence and Synopsys, while the large system design companies have throttled back on the research components of their activities.
- Transition of academic research to industry is much harder than it used to be. Technologies are more complex and it is harder to get new ideas into the sophisticated and mature software offered by EDA vendors.

Recommendations

Based on my findings, I believe if the following points are taken into consideration, EDA programs will improve in the near future.

1. Funding of Research Programs by all levels of governments.
2. Enhanced Education Programs in affected institutions of higher learnings.
3. Collaboration with industries and Institutes for possible Innovations and Inventions of new packages for much easier designs, developments and implementations.

Conclusion

The sophistication and Complexity of current electronic systems, including printed circuit boards (PCBS) & integrated circuits (ICS) are a direct result of Electronic designs automation (EDA). Conversely, EDA is highly dependent on the power & performance of ICS. Such as Microprocessors and RAMS used to construct the computers which the EDA software is executed. As a result, EDA is used to develop the next generation of ICS, which, in turn, are used to develop & execute the next generation of EDA, and so on in an ever-advancing progression of features & capabilities.

References

C.O.C. Oko & D.P.S. Abam, (2012) 2rd Ed. *Engineering Professional Practice and Procedures*, University of Port Harcourt Press, University of Port Harcourt Nigeria.

C.W. Liu and Y.W. Chang (2007), Power/Ground Network and floor plan co-synthesis for fast design convergence, *IEEE Trans on Computer-Aided Design*, 26(4), 293-704.

Chang Y.W. Chang and S.P. Lin, MR 2004: A New Framework of Multilevel full chip routing *IEEE Trans on Computer-Aided Design*, 23(5), pp. 793-800.

Deming Chen dchen@illinois.edu retrieved 5/6/2017

Gajski, D.D.; Abdi, S.; Gerslauer, A.; Schirner, G (2009), *Embedded System Design, Modelling, Synthesis and Verification*, XXV, 352 p, Hardcover. ISBN: 978-1-4419-0503-1, http://www.springer.com/978-1-4419-0503-1.

S. X.D Tan and C.J.R Shi, (2003). *Efficient Very Large Sale Integration Power/Ground Network Sizing Based on Equivalent Circuit Modeling, IEEE Trans on Computer-Aided Design*, 22(3), pp. 277-284.

International Technology Roadmap for Semiconductors, *http://www.itrs.net/.*

T.C Chen, Y.W. Chang (2006), Modern Floor planning based on B* Trees and Fast Simulated Annealing, *IEEE Trans on Computer Aided Design, 25(4), pp. 637-650, April 2006*

T.C Chen, Z.W. Jiang, T.C Hsu, H.C. Chen, Y.W. Chang, (2008) NTUplace3, An Analytical Placer for Large Scale Mixed Size Design With Preplaced Blocks and Density

Constraints, *IEEE Trans on Computer-Aided Design, 27(7), pp. 1228-1240.*

Y.W. Chang, T.C Chen, and H.Y. Chen (2007), *Physical Design for System on a Chip in Essential Issues in SOC Design, Y.L, Lin, Editor, Springer, Boston.*