AUTOMATION AND CONTROL IN ELECTRONIC INDUSTRIES

Popovic D.
University of Bremen, Germany

Keywords: design automation, computer-aided design, rapid prototyping, semiconductors production, process monitoring and control, run-to-run control, visual inspection, feature extraction, defect classification, automated test equipment, PCB inspection, IC inspection, automated assembly, assembly robots, peg-in-hole assembly, packaging and interconnections

Contents

Summary

This article presents a survey of methods, tools, and approaches to automation and control in the electronics industry, including the major areas of design, production, and manufacturing both of electronic components and of systems. Design automation approaches are first presented, and then attention is focused on the methods of design

testing and on automated test equipment (ATE). The main application fields of automation and control, semiconductor manufacturing, and automated assembly are considered in depth. The main focus is on the modeling, monitoring, and control of manufacturing processes, and on the related visual product inspection, defect detection, and defect classification. In conclusion, automated assembly is considered (including the use of assembly robots).

1. Introduction

The electronics industry is a complex branch of production engineering. It involves a large number of different production steps and other engineering activities, starting with product design, prototyping, modeling, simulation, test, inspection, and quality control, and including the monitoring and control of semiconductor manufacturing processes, as well as the assembling and packaging of electronic systems. Therefore the range of automation and control in this field of production engineering is extremely wide. In what follows, an outline is presented of some typical applications of automation and control in electronic industries.

The rapid developments in electronics, the tremendous increase in the complexity of electronic systems, and the decrease in their geometrical dimensions, have since the very beginning called for automation aids at all stages of the production of electronic systems and components. This has particularly been amplified since the globalization of, and the increase in competitive pressure in, the international electronics market, the volume of which is estimated to exceed US$1.5 billion. International competition drives up production standards, leading to both high product quality and a high price/performance ratio for the product.

This is achievable through a decrease in overall production time, a reduction in production costs, and an increase in product quality using automation strategies. For the same reasons, the design and rapid prototyping of new products are increasingly becoming very significant, because in this way the product development time is reduced, and experimentation at the extremely expensive and time-consuming production level is largely avoided. Rapid prototyping is especially important. This helps the designer to estimate the physical realization of the product and the performance to be expected from it, and includes simulation test runs of product behavior.

The major production steps include the manufacturing of the semiconductor modules that are the core parts of electronic devices. This is a complex and highly integrated production process requiring a number of successive manufacturing and test steps, until the final product—the VLSI chip—has been produced. What then remain are the assembly and packaging of the final electronic devices, which have to be both functionally tested and evaluated for performance before being delivered to the customer.

2. Design and Test Automation

We are presently witnessing an explosive development in semiconductor technology. The technology already makes it possible to pack over 100 million transistors on one

integrated circuit (IC), whose clock frequency is increasing enormously. It is therefore a huge challenge to develop design automation approaches for such complex electronic systems that will support the system designer and speed up the design process. It is expected that such approaches should integrate the design methods and tools that are already available, take into account the valid standardized design specifications, and be able to deal with up-to-date technological knowledge. Based on this, a number of development systems have been elaborated and methodologies worked out that address design problems at all levels of abstraction: that is, from system conceptualization up to the finalization of chip design. They also help carry out circuit simulation, placement, routing, and so on, up to high-level synthesis and optimization of—among other issues—the use of space, power consumption, and the handling of timing delays.

2.1. Design Approaches

A number of development tools are in use in the silicon industry to deal with particular design problems at different system levels. Currently, various design tools within the A4XX Projects for high quality design in deep submicron technology are preferable for CMOS applications. Various advanced system-level methods and tools that incorporate new design methodologies and support intellectual property reuse are also under development.

Advanced design concepts and enhanced development tools are required when dealing with deep submicron technology, because here a single dies can hold billions of transistors running at Gigahertz frequency. The inevitable difficulties created here by the physical constraints, such as device, connect, and power limits, increase the need for advanced development tools to ensure that timing and other design requirements are met, and to cope with other problems emerging with the increase of system complexity.

For instance, on the logical level of system development, traditional interconnection tools that optimize the total gate delay are not efficient enough, because they do not help optimize logic design and physical placement simultaneously. Furthermore, at the system level of development, computer-aided design (CAD) tools should determine the interconnect delay and the probability of congestion. Above all, the tools should help to reuse intellectual property, and in this way increase the reliability of system design.

Considerable difficulties, however, emerge when chips contain both circuitry and the software that runs on it, and the requirements of coherent hardware and software design need to be met simultaneously. The substantial compatibility of the simultaneously designed hardware and software components should be confirmed through verification. In the past, this was a shared responsibility, with circuit designers being responsible for the circuitry and programmers for the software, with the expectation that the design results of both experts would prove to be compatible in the later joint test at system level. For hardware design the language VHDL has been preferred, and for software implementation the C or C++ language. The use of two different formal languages in the development of complex systems, however, impedes a coherent system description. As a consequence of these developments in complex systems, a common formal language is required for the design and formal description of the systems on the chip. This needs to be capable of coping with both hardware and software design tasks throughout the

entire system development process, from formal system specification up to its automated verification. Alternatively, an existing formal language like Java could be modified to meet the requirements of both descriptions.

Many efforts have been made to develop systems to handle the codesign of hardware and software, the most valuable of them being the attempt to conceptualize a high-level formal language that is targeted at harmonic unification of the entire design process, and at the same time reflects as far as possible the existing valuable design methodologies. Thus far, the great perspectives have shown the Superlog language, based on interweaving the C language and Verlog. This should not be difficult to implement because much of Verlog was built on C bases. The resulting language could be improved considerably by the inclusion of some elements of VHDL and Java, which will enable the efficient design of combinatorial, sequential, and multi-valued logic circuits. Furthermore, the full version of Superlog also supports the efficient description of protocols, interfaces, state machines and the like.

After a method has been chosen to control the system design process, a no less difficult problem has to be solved: that of system verification and system validation. When the system to be verified is highly complex, these tasks require very sophisticated approaches. The use of standard emulation approaches for this purpose (for instance by translating the designed system into the equivalent in FPGA (field-programmable gate arrays)) creates considerable difficulties because of the huge number of gates and the very high operating clock frequency required. These approaches cannot emulate the real intended performance of the system. However, FPGAs today considerably narrow the gap between the hardware and software. They implement arbitrary combinatorial or sequential circuits, and their structural interconnection can easily be configured. Today, boards populated by FPGA chips are still attractive for the emulation and/or implementation of hardware, and the execution of the related software.

For formal system verification, both simulation programs and prototyping techniques are used. However, computer-aided prototyping using programmable FPGAs for circuit validation is preferred as a valuable step after the circuit design has been completed, and before the circuit has been manufactured. This is because system prototyping in this way, instead of system simulation, enables more authentic data to be extracted on the functional correctness and performance of the designed system. Nevertheless, for the time being no suitable approach is available that permits the full verification of complex systems. The approaches in use solely provide the designer with the possibility of checking particular system aspects, such as probable deadlock conditions, and congruency of design representation.

Simulation has been used for a long time for circuit validation. Its applicability to the validation of codesigned systems is still restricted, because only a limited number of input/output test patterns can be analyzed. There are also some additional difficulties with the building and use of an adequate system model. As the way out of this restriction symbolic simulation is recommended. This is a blend of simulation and formal verification. It could be defined as simulation based on system-level modeling. For this purpose, system behavior can be modeled using cooperating control-data-flow graphs and finite-state machines. System-level partitioning is an alternative approach. It

is more effective in balancing the system split between hardware and software components, as a basis for optimizing the overall system cost and performance.

2.2. Rapid Prototyping

Rapid prototyping is a computer-aided systems development paradigm, in which various methodologies for system design, simulation, test, and evaluation are combined, to result in a scaled-down prototype of the system on the way to full-scale system development. This prototype building is of advantage because it saves time in final system development, and considerably reduces the required experimental work while developing the system. It also permits a multiple system redesign during development at very low cost when compared with the design and redesign of a full system version.

The rapid prototyping of electronic systems is primarily related to the automated design of electronic hardware, although it has also contributed considerably to the evolution of software. The prototyping mainly encompasses the methods and tools of hardware synthesis, simulation, and programmable implementation. The objective is to reduce the time between the initial system draft and its final development. This is achieved in three successive steps. In the first step the draft simulations under various aspects of system functionality. For each aspect that is simulated, possible conceptual errors can be discovered, and the performance and implementation costs can be estimated. In the second step, the simulation results are used for synthesis of the relevant electronic hardware. In the third and final step, fast programmable versions of the synthesized electronic hardware are implemented and tested, before final selection of the best version to be fabricated.

Simulation is frequently used as a design verification tool. Both analog and digital electronic circuits can be simulated and their functioning verified. The simulation of analog circuits corresponds to the simulation of dynamic systems, and can be implemented using the methods and tools of systems engineering. The simulation of digital circuits is simplified by observing the binary voltage changes at circuit output in response to the binary changes of signals at circuit inputs. The simulation models used for this purpose could be simplified timing models, enabling the study of circuit behavior for specific parallel or sequential input signal combinations. With each step in the simulation the system designer can confirm (or not) the expected proper circuit behavior. Here, two simulation approaches are to be distinguished:

- Simulation of combinatorial digital circuits, or logic circuit simulation. In this type of simulation, the user selects and inserts the signal combinations to the circuit inputs; and
- Simulation of sequential (asynchronous) logic circuits. For this an event-driven simulation is preferred, in which the new simulation steps are triggered by individual changes of signals within the circuit within prescribed time slots.

Powerful hardware simulation languages are available for the building and implementation of simulation models. A keyboard and monitor are used to facilitate interaction with the model and to observe the simulation results. VHDL (very high-speed integrated circuits hardware description language), a hardware design language

proposed by the US Department of Defense and standardized as IEEE Standard 1076, is predominantly used as a hardware simulation language. The language supports an iterated top-down simulation–synthesis methodology at the architectural, behavioral, and structural abstraction levels, by mixing the data flow and the behavioral and structural system description. This enables each system module to be individually represented on the various abstraction levels required for its simulation, synthesis, testing, and so on. Because of its wide international acceptability, VHDL can be used in the majority of available computer and software simulation environments. An analog version of VHDL has also been launched, designed for the simulation of analog and hybrid analog–digital circuits.

The main VHDL elements are *processes*, structured by combinations of related *functions* and *procedures*, and *entities*, representing the specified *interfaces*, *signals*, and user-defined *parameters*. The processes run concurrently during the system simulation, allowing the simulation of parallel and event-driven phenomena.

A remarkable alternative to VHDL is Verlog, a hardware simulation language that became IEEE Standard 1364 in 1995. This language is very similar to VHDL, although more restricted. Some vendors deliver Verlog along with a VHDL compiler to run on the same computer.
The system development stages of the rapid prototyping methodology, from the initial system concept up to its programmable implementation, include a number of hierarchical levels. Each level involves a particular abstraction pattern and its validation by simulation. Examples of typical hierarchical levels are:

- *High-level synthesis*, which starts with modeling the initial behavioral system in a simulation language. The model includes the main hardware functions to be implemented, along with the design constraints. These could for example be local or global execution time restrictions, or cost limits. In this level the modeled system behavior is translated into a possible register transfer level system description, to be processed in the next development stage.
- *Register transfer level synthesis* mainly deals with the optimization of the controller and the minimization of the associated logic. For the controller synthesis, the concepts of finite state machines are used for preference. The output of this level is an optimized gate level design and an inter-chip partitioning draft for easier programmable implementation and testing.
- *Physical level synthesis*, which maps the signal connections of the gate level draft into programmable implementations of minimized delays, which are appropriate for the use of field programmable logic circuits. At this development stage the synthesis results at gate level are mapped onto the cells of such circuits, and the corresponding cell interconnection programming pattern is produced.

Programmable logic circuits were introduced some time ago to support the reconfigurable implementation of digital circuit prototypes, and in this way to reduce total development costs. Depending on their primary application field, the programmable logic circuits used can be PALs (programmable arrays logic), PLDs (programmable logic devices), or FPGAs (field programmable gate arrays). The latter is

the most frequently used in rapid prototyping.

FPGAs are predominantly used for the functional verification of microelectronic systems. They can be structured in a number of different ways: for example, as rows of logic cells, cell and embedded arrays, and hierarchical logic blocks. They use a number of different programming technologies, such as fuses and anti-fuses, EPROMs and EEPROMs, SRAMs and FLASHs. The most recent trend appears to be the use of boards with what are known as field programmable interconnection circuits or devices (FPICs).

The designer of electronic hardware systems can save a significant amount of time through rapid prototyping as described here, but it is possible to make an additional time saving by using dedicated rapid prototyping tools instead of starting the system design process by its formal description. For instance, a rapid prototyping engine for multiprocessor design has been elaborated, suitable for the rapid prototyping of multiprocessor systems using low-cost hardware emulation. The engine largely comprises FPGAs and other off-the-shelf components, and enables a high-speed system emulation in comparison with programmed simulation.

2.3. Intelligent Design and Test Tools

The last three decades of the twentieth century saw attempts to use the methods and tools of artificial intelligence to automate most different phases of systems design and production in the electronics industry. It was recognized very early on that knowledge-based approaches could help a great deal with system design, verification, test, inspection, fault diagnosis, and so on. A variety of dedicated expert systems have been developed, as computer-aided tools to solve specific automation problems, particularly to address various aspects of the systems design and fabrication of very large scale integration (VLSI) circuits. This was necessary because with the increase of VLSI circuit complexity, traditional CAD tools approached the limits of their applicability.

Before the design process is initiated, it is necessary to specify the functions that the system is to perform, and the given performance specifications and design constraints. The design process itself then runs top-down, beginning with the conversion of the high-level functional specification into adequately interconnected functional units. The specifications of individual functional units are then translated into electronic circuits, and circuit analysis is carried out in order to evaluate the correctness, quality, and acceptability of the design before the circuit is made. Behavioral, logical, timing, or circuit simulation tools can be used for this purpose.

A number of the expert system tools that have been developed for automating the design of VLSI circuits are briefly presented below.

- CRITTER, an expert system that assists in checking the correctness, timing, robustness, and speed analysis of circuits, based on circuit diagrams defined through the circuit input–output specifications using frames.
- DAA, an expert system that translates the algorithmic data flow and transcribes the presentation of a VLSI system into a hardware allocation, in the form of a

technology-independent list of registers, operators, data paths, and control signals.

- PALLADIO, a circuit design environment system that assists the design and test of new VLSIs, particularly of nMOS circuits, using interactive graphic editors, a rule editor, and a circuit simulator to refine the design specifications at different levels of abstraction.
- PEACE, an expert system that supports electronic circuit design by performing circuit analysis based on a functional description of the basic circuit components, connection rules for complex circuit interconnections, functional and topological circuit transformations, and so on.
- REDESIGN, an expert system that helps redesign digital circuits to meet redefined functional specifications by making local changes within the circuits.
- SADD, an expert system that enables the design and testing of digital circuits using a frame-based model, with knowledge of component characteristics and circuit behavior.

Dedicated expert systems have also been developed as design automation tools for integrated hardware and software design. By combining expert knowledge and external algorithms, they aim to cover most of the major design requirements, including performance, reliability, and cost at circuit level, and timing constraints at system level. Such expert systems are also designed to generate automatically the check and test program related to the designed circuitry. This releases the hardware designer from the need to become involved in programming. Finally, by coupling the layout software and simulation package, the expert system is capable of generating automatically the PCB layout and carrying out the system simulation.

In addition to the design tools described above, a number of intelligent evaluation tools have also been elaborated. Among the best-known are:

- DFT, an intelligent "design for testability" tool that checks for rule violations, analyzes those it detects, and corrects the design to remove them using a level-sensitive scan technique.
- DIALOG, an expert system that analyzes VLSI circuits to discover possible design errors, using knowledge about the frequent design errors made in specific design technologies.
- Hitest, a knowledge-based test generation system operating on a frame-based representation of expert knowledge, such as the functionality knowledge of higher-level modules like flip-flops and counters.

Finally, a series of intelligent tools has been developed for automatic troubleshooting and error diagnostics of electronic circuits, including:

- FOREST, an intelligent tool that isolates and diagnoses faults in electronic circuits using automatic test equipment diagnostic software, based on experimental expert knowledge, library knowledge and circuit diagrams, and heuristic knowledge gathered in the process of electronic troubleshooting.

-
-
-

Bibliography

Card J.P., Sniderman D.L., and Klimasauskas C. (1997). Dynamic neural control for a plasma etch process. *IEEE Transactions on Neural Networks* **8**(4), 883–901. [Experimental results are presented on using a cascade neural network and a policy-iteration optimization routine for dynamic control of a plasma etch process.]

Chan S.P. (1998). Robotic Assembly in electronics manufacturing systems. *Industrial and Manufacturing Systems* (ed. C.T. Leondes), pp. 145–179. San Diego, CA: Academic Press. [An introduction to the approaches of robot-based manufacturing processes in electronic industry, particularly in PCB assembly using neural networks.]

Chen R. and Spanos C.J. (1992). Self-learning fuzzy modeling of semiconductor processing equipment. *Fuzzy Logic Technology and Applications* (ed. R.J.Marks II), pp. 267–273. New York: IEEE Technical Activities Board. [A qualitative equipment model for low-pressure chemical vapour deposition, based on a fuzzy representation of input–output relationships and on the use of self-tuning membership functions is presented along with the results of its simulation test.]

Chou P.B., et al. (1997). Automatic defect classification for semiconductor manufacturing. *Machine Vision and Applications* ???(9), 201–214. [This paper describes the planning and realization of a machine-vision-based system for automated defect classification of manufactured semiconductor chips at various manufacturing steps.]

Hurwitz A.M., Moyne J., and del Castillo E.(2000). *Run to Run Control in Semiconductor Manufacturing*, 384 pp. Boca Raton, FL: CRC Press. [This represents the comprehensive edge for quality control and manufacturing improvement].

IEEE Micro. (1998). Chip–Package Codesign, Special Issue on New Route in System Integration, **18**(4), 7–59. [Future substrate and packaging technologies, as well as the methods of their modeling, simulation and design are presented.]

IEEE Spectrum. (1999). The 100-million Transistor IC: Design, Fabrication, Packaging, Testing, Special Issue, **36**(7), 22–73. [This is an up-to-date review of problems and their solutions in automated design, manufacturing, assembly, and test of emerging technology components and devices.]

Limanond S., Si J., and Tsakalis K. (1998). Monitoring and control of semiconductor manufacturing processes. *IEEE Control Systems* **18**(6), 46–58. [The paper provides a survey of methods and concepts of model-based automation of monitoring and control of semiconductor manufacturing processes, and describes in depth the implementation and evaluation of a neural optimal etch time controller.]

Pau L.F. (1990). *Computer Vision for Electronics Manufacturing*, 324 pp. New York: Plenum Press. [The author gives a comprehensive description of methods and metrological tools of vision-based inspection of PCBs, ICs, and other semiconductor devices in electronic manufacturing.]

Postula A. and Christodoulou. (1999). Rapid prototyping of mechanical and electronic subsystems of mechatronic products. *Mechatronics in Engineering Design and Product Development* (eds D. Popovic and L. Vlacic), pp. 359–412, New York: Marcel Dekker. [Presents an overview of methods, tools, and technology used in rapid prototyping of electrical and mechanical component of mechatronic systems, that include the use of design methodology, programmable hardware technology, and hardware simulation.]

Biographical Sketch

Professor D. Popovic received his M.Sc. (Dipl.-Ing.) from the University of Belgrade and his Ph.D. (Dr.-Ing.) from the Technical University of Berlin. He has been with the University of Bremen since 1972 as Chair of Process Control Computer Engineering. In 1982 he established the Institute of Automation Technology and has headed it since the very beginning. Before coming to the University of Bremen he worked for more than 10 years with AEG-Telefunken in Berlin and Bayer AG in Leverkusen, in charge of the application of computers to industrial plants automation.

Professor Popovic is the author and editor of several books, including *Distributed Computer Control for Industrial Automation, Methods and Tools for Applied Artificial Intelligence,* and *Mechatronic Approach to Process and Product Design.* He has also contributed eight chapters to various books in the areas of expert systems, neural networks in systems control, and communication links for industrial automation His field of research includes, besides process control, the application of intelligent technology to texture analysis, video data compression, computer vision and intelligent robotics, where he has over 140 publications.