Project Nr 3216

Turning the Formal Verification of VLSI Hardware into Reality[1]

L. Claesen[2], D. Borrionel[3], H. Eveking[4], J.L. Paillet[5], P. Prinetto[6]

Abstract.

This paper presents an overview of the different aspects in the area of the formal verification of VLSI hardware. For particular aspects of the problem area the adequate approaches are being addressed. In this respect an overview of major directions and achievements in the area of formal hardware verification as under research in the CHARME ESPRIT Basic Research Action are presented. All partners are convinced that formal verification, given the appropriate methodologies, algorithms and formalisms, will find its place in actual CAD systems for industrial hardware designs. Research results include among others a link-up of formal verification tools to VHDL as well as the demonstrated Mi formal verification of actual VLSI chips of over 32000 transistors from the layout up to high level algorithmic specifications.

1. Goals of Formal Hardware Verification: Why is it needed?

The constant evolution in the microelectronics technology continuously allows to integrate larger and larger systems in integrated circuits and systems. This has its consequences in the fact that complicated chips of over 1 million transistors are realizable and that semi-custom approaches of standard cells and gate arrays have emerged to an enabling technology, not only for the classical electronics systems industries but also for innovative SME's. Electronics systems emerge in all aspects of every day life such as consumer electronics, telecommunication, HDTV, speech and image processing, computing systems, automotive applications etc... Instead of being *technology limited*, complex electronic systems have become *design limited*. It is indeed crucial that these complex systems, as required by industries incorporating electronics in their products, are *first time right*. Design errors should be detected as soon as possible in the design phase, because erroneous designs will introduce considerable delays in the introduction of innovative products on the market. These on their turn induce unacceptable losses in profit.

The classical way in which digital systems are being evaluated for design correctness is by a huge amount of simulation experiments. It is however well known that, except for trivial and obvious examples, exhaustive simulation covering all possible patterns is impossible to perform due to the problem of combinatorial explosion.

Formal design and verification techniques [4] attempt to address the design problem in an analytic way in order to obtain mathematical guaranteed correctness with respect to the modeling method used. Automatic synthesis from high level specifications [1,21

1 Research sponsored by the EEC under the ESPRIT Basic Research Action 3216: CHARME

2 IMEC / Kath. Univ. Leuven, Kapeldreef 75, B-3001 Leuven Belgium

3 Lab. IMAG-ARTEMIS, Inst. B.P. 53X, F-38402 Grenoble Cedex, France

4 Inst. fuer Datentechn., Merckstrasse 25, Techn. Univ. Darmstadt, D-6100 Darsmtadt, F.R.G.

5 Université de Provence, CASE Y, 3, Place Victor Hugo, 13331 Marseille Cedex 3, France

6 Polit. di Torino, Dip. di Autom. e Inf., Corso Duca degli Abruzzi, 24, I-10129 Turin, Italy

is an approach that can already provide some solutions for specific design aspects and abstraction levels. However, due to the unbeatable human insight in the underlying design problems, there will always be a large human and manual contribution in the design of complex digital systems, which need to be verified. Even in the case of so called "correctness-by-construction" methods, formal verification methods will enable the discovery of software bugs in the synthesis systems at hand by cross checking the results of the synthesis process.

Formal methods have already a long tradition and constitute a number of aspects and different approaches in the area of software development. In fact a lot of European research in this area is sponsored in ESPRIT projects [3]. The correctness of hardware designs have much more impact on direct costs involved in iteration cycles (due to VLSI processing etc.) and late introduction of products into the market. This motivates even more the need for formal verification in hardware than it is already in software design. The ongoing research in hardware verification, as is the goal in the CHARME project, is driven by a need to verify complex digital systems in as automatic a way as possible. This is required to foster acceptance in the electronic CAD commmunity.

The goal of the ESPRIT CHARME Basic Research Action is to investigate, evaluate and prototype promising approaches that can help guarantee digital hardware correctness. For this purpose a grouping of researchers with backgrounds in electrical engineering, computer science and mathematics has been formed. These different backgrounds allow to put a number of different approaches and view points together in order to develop appropriate techniques and methodologies in order to address the problem of hardware correctness in the best way. All partners in the action are convinced that formal hardware verification can be converted into actual CAD systems and design practice. Therefore prototyping and application of the ideas to actual electronic design problems is ultimately important. Only the head lines of the research in CHARME are described in this paper.

Formal methods address a large number of design areas and levels of abstraction and encompass several different approaches to tackle these problems. Therefore an overview of the major *design aspects and levels of abstraction* is given in section 2. Based on this design aspect and abstraction level classification a description of individual formal verification techniques addressing specific areas is given in section 3. In order to facilitate the use of specific verification techniques, *methodologies* and/or *design rules* could result in "Design for Verifiability" in similar lines as what has been achieved by "Design for Testability" [6] as described in section 4. In section 5 the major ideas for the future direction of the research in formal hardware verification are indicated, followed by the conclusions in section 6.

2. Design Aspects, Levels of Abstraction

Format Methods is a term that is used to cover several meanings, related to either different *aspects,* different *levels of design abstraction* as well as different underlying *formalisms* and *methodologies*. In order to clarify this situation a problem oriented, overview of design aspects and abstraction levels is given in this section. This classification is important because the different aspects and levels of abstraction often have given rise to specific developments of methods and algorithms for formal verification. An overview according to formalisms is given in [4]

In VLSI hardware design, errors can be introduced at several places such as layout rule violations, circuitry etc... For synchronous digital VLSI designs the verification of the correct functionality can be subdivided in to the aspects of timing-, electrical- and

behavioral correctness verification. In this paper we mainly concentrate on the correct implementation versus specification behavioral verification.

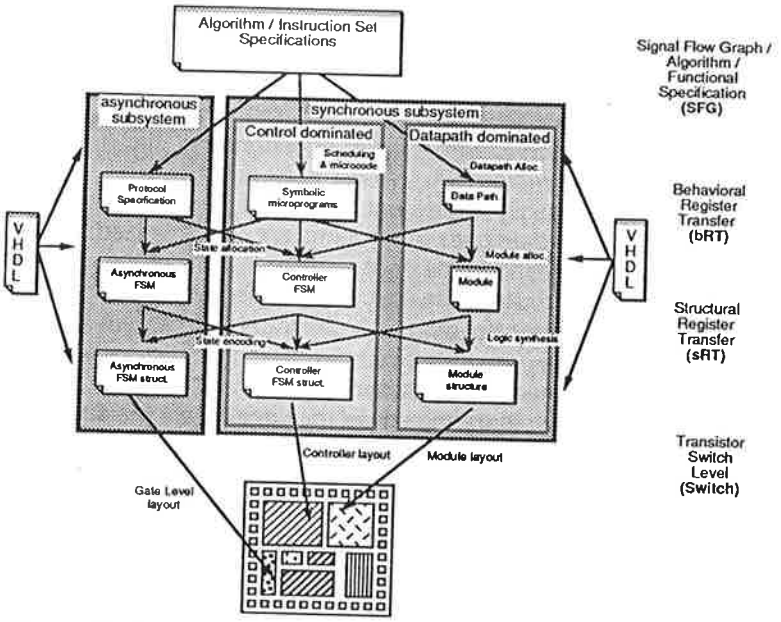

Figure 1: Aspects (horizontal) and Levels of Abstraction (vertical) in Hardware Design.

Figure 1 provides an overview of current digital system design, including the specific aspects when VLSI implementation is targeted. It is assumed that digital system design starts from a *formal* specification at the behavioral or signal flow graph level. Such specifications usually take the form of software models in classical programming languages such as FORTRAN or C. For digital system specifications at these high levels, several dedicated designer oriented system and hardware description languages as well as more mathematical languages have emerged. VHDL [5] is a standard system and hardware description language (HDL) that is gaininng acceptance in industry and is supported by CAD vendors with compilers, simulators and logic synthesis systems. VHDL is a language that can not be overlooked anymore. It allows to describe digital systems at several levels of abstraction, using several description styles, namely the behavioral, structural and dataflow styles. Thus, the behavioral specification as well as the circuit implementation can be given in VHDL. For all these reasons, we believe that VHDL is a good candidate as input language for a Formal Proof environment.

.1 Synchronous - Asynchronous Subsystems

In digital hardware systems, a major subdivision of *design aspects* is the subdivision of *asynchronous* and *synchronous* subsystems. To be able to manage the design process and for increasing testability [6], the largest part of current digital designs are synchronous.

.1.1 *Asynchronous Subsystems.*

The interfaces to the externals of a system or other integrated circuits are most often realized by asynchronous subsystems. The asynchronous interfaces have to imple-

ment communication protocols among digital systems. They form the bridge between the asynchronous (interface) world and the synchronous regions on the chip. Such asynchronous circuitry usually consists of small parts of the global circuitry, but nevertheless have to implement the secure communication with the outside world.

2.1.2 Synchronous Subsystems.

For the synchronous subsystems a major subdivision of circuitry in *control dominated-* and *data path dominated* subsystems is possible (see fig. 1). The data path dominated subsystems usually implement the vector operations such as additions, multiplications, etc.. These data path subsystems are characterized by the fact that the circuitry is generated in relation to the chosen word lengths of the digital words being processed. In integrated circuits these are usually implemented using structured layout techniques. Typical applications of data path dominated circuits are for example ALU's, ACU's Multipliers etc... They are heavily employed in the data paths of computers as well as in digital signal processing applications [2].

Control dominated subsystems implement the control over the data path dominated subsystems as well as implement the digital control of an application at hand. Several of the current ASIC applications fall in this category. Control dominated systems are usually much more random in nature than data path dominated subsystems, and are usually implemented using random logic (currently often done as standard cells or gate arrays).

2.2 Levels of Design Abstraction.

An other classification of design aspects in fig. 1 is according to the levels of abstraction (in space and in time): (1) signal flow graph / algorithmic / instruction set level: **SFG**, (2) behavioral register transfer level: **bRT**, (3) structural register transfer level: **sRT**, (4) MOS transistor switch level: Switch, as described further on.

2.2.1 Signal Flow Graph / Algorithmic / Instruction Set Level.

At the highest level of design abstraction, the signal flow graph defining the behavior of the algorithm to be synthesized in hardware is considered. At the SFG level it is not specified *how* the algorithm is implemented in hardware. This could in fact be done in numerous ways (e.g. bit-serial, micro-code processor type of architecture (with several variants), bit-parallel or any combination of these). It is independent of how operations are performed, either on dedicated hardware blocks or on general purpose ALU's. It should be noticed, that very often the topology of the SFG will not directly correspond to the topology of the synthesized architectures.

In the case of processor design, the highest level of design abstraction is the specification of the instruction set (reference manual). This defines the processor-memory system as it is seen by the software programmer. The memory is seen as an array of adressable words of a given length, and only the internal registers of the processor which are made visible to the machine language are represented at this level. The internal structure of the processor, the processormemory detailed exchanges, are not part of this level of specification. On the contrary, the emphasis is on the definition of the operation codes, addressing modes, and on the result of the execution of each individual instruction, all in symbolic form.

2.2.2 The Behavioral Register Transfer Levels (bRT-level).

In high level synthesis [1,2] the first steps consist of transforming the SFG level specification into a *data-path* and a *schedule*. In the behavioral register transfer level, the specific data path operators, such as the amount of ALU's, multipliers etc. and the bus interconnection, are known from the data path allocation. The allocation of variables to registers, and operations to data path operations over specific machine cycles is done in the scheduling. At the behavioral register transfer level, specific registers and data path hardware blocks are known. The micro-code to control the data paths, and the state encoding is in *symbolic form*.

2.2.3 The Structural Register Transfer Level (sRT-level).

At the structural register transfer level, the interconnection of logic building blocks, registers and state variables are modelled in the same way as they are implemented in the ultimate hardware. Starting from the schedule as available at the bRT-level, the controler is synthesized, starting from a generic structure. A specific state assignment has to be done. The data path hardware is realized by a library of parameterized building blocks. At sRT level, a logic description is available which corresponds to the structure and state encoding in the actual hardware implementation. In this paper, no distinction is made between the sRT-level representation and a gate level representation, as they are both very close together.

2.2.4 The MOS Transistor Switch Level (Switch-level).

When targeting towards MOS VLSI implementation of the system at hand, all digital circuitry has to be implemented in terms of MOS transistors. Transistors can be described in general at the circuit level using non-linear device models, that accurately model the voltage and current relationships at the terminals. For synchronous digital circuits, the switch level abstraction, that abstracts MOS transistors as switches and circuit nodes with a number of different strengths [7] has proven its usefulness.

. The Formal Verification Map.

Given the outline of design aspects and levels of abstraction of fig. 1 as described section 2, it is now better possible to describe the *Formal Verification Map* as indicated figure 2. Here we indicate the most important developments and relate formal erification work (by the numbers in circles on figure 2 and near the titles in the text) nd achievements in CHARME that address specific design aspects and levels of bstraction. This description starts from the lowest levels of abstraction as it is there at the most progress towards automated formal verification has been made.

1 Transistor Switch Level Analysis. (1)

After the generation of the mask layouts it is possible to extract the transistor etworks as they are actually implemented on an integrated circuit. This is a good epresentation for the actual circuits and their behavior as they will ultimately be realized n silicon. In order to be able to capture all possible design errors (also in geometry nd interconnections) this representation is a very good one to start the formal rification from. When abstracting to the behavioral modeling, the transistor switch vel representation [7] is currently very well accepted to perform ultimate simulations

on in order to check the circuit functionality. Symbolic analysis techniques of switch level circuits [9,13] based on the solution of a number of systems of Boolean equations [8] have been developed. In the CHARME project similar techniques have been realized [11,12] in the BOTRYS program. An alternative approach based on the formulation of the transistor switch level model in predicate logic has also been worked out in CHARME [14] in the SWAN program.

These approaches have been used for the verification between the sRT and the switch level.

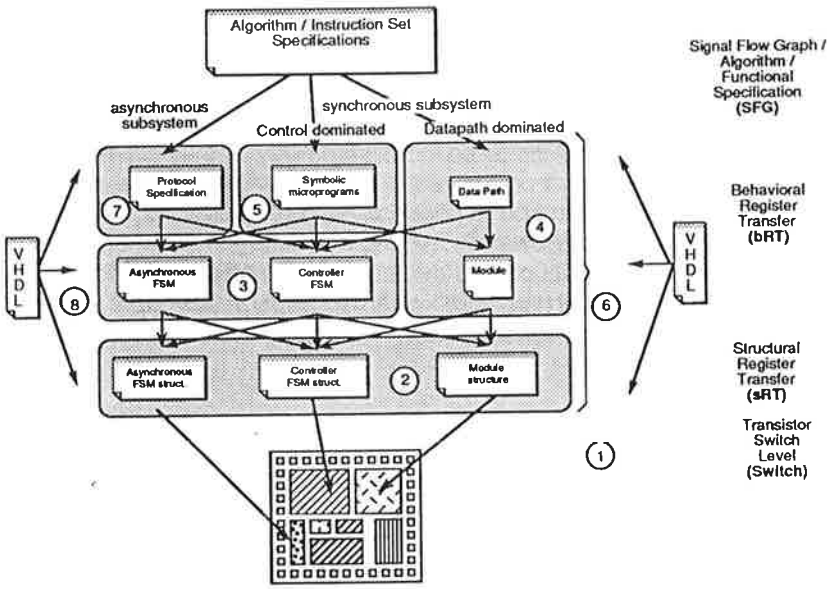

Figure 2: The Format Verification Map. The numbers in circles refer to descriptions of formal verification techniques in the text.

3.2 Combinatorial Logic Blocks. (2)

For the verification of different design representations at the sRT level, it is possible, due to the knowledge and use of the same state encoding of registers in specification and implementation to reduce the problem of formal verification to the comparison of the Boolean functions of the combinatorial logic blocks. For the comparison of Boolean functions, tautology checkers can be used. A comparison on the same benchmark set [17] of over 10 tautology checking algorithms, from all over the world including algorithms as developed by CHARME partners has been made. From this it has become clear that the methods based on the concept of OBDD's (Ordered Binary Decision Diagrams) [19] and their improved derivatives [20,21,22] are clearly superior to the other existing methods in their performance. The methods of OBDD's are currently recognized as the most efficient approaches for Boolean function comparison. OBDD's are used as the basic representation formalism of Boolean formulas within most of the research in the CHARME project [10,57,18]. The comparison of representations at the sRT levels ((structural) register transfer level & gate level) [20] has been used for

checking the correctness of *the individual combinatorial building blocks* in chips of up to 300,000 transistors [23] and have currently found industrial application in Bull.

For the verification of different representations at the sRT level in the assumption of the same state encoding, the LOVERT [57] program has been developed and coupled to VHDL. Implementation verifications of a 32-bit ALU and small microprocessors has been done in cpu seconds.

3.3 Finite State Machine Applications. (3)

The comparison of the Boolean functions of combinatorial blocks by means of tautology checkers (e.g. OBDD's) is only directly possible when the same state encoding is used in the two representations. In several applications the correspondance in the encoding of the two representations of digital circuits is not directly known. E.g. a controller can be defined at the bRT level with symbolic states and is implemented at a lower sRT level with specific states. To cope with this problem general algorithms for the comparison of two finite state machines, for which the state encoding and their correspondances is not known have been developed.

A major breakthrough in the complexities of comparisons of FSM's has been achieved with the symbolic representations (instead of specific enumeration) of the visited states [27]. The achievements in these comparisons have lead to verifications of FSM's with more than 1020 states, which corresponds roughly to some 60 state variables [29,30]. This means that small controler like circults can be verified by this. Based on these symbolic techniques, a program called EPOS [65], for the verification of FSM's as black boxes has been developed in CHARME. In addition specialized algorithms [25,261 for FSM verification on the product machine have been developed in CHARME. The two algorithms differ in the following points: algorithm #I: forward time processing and explicit enumeration to dynamically build the product machine; algorithm #2: reverse time processing, function representation as cubes, cube extraction by means of a Podem-like implicit enumeration procedure.

The verification of complete hardware designs, which most often have much more than 60 state variables, can due to complexity problems not yet be verified by these techniques. As no single technique is clearly superior for all applications, further research for the appropriate combination of algorithms and exploitation of the circuit structure is required here.

3.4 Parameterized Hardware Modules. (4)

In contrast to control dominated circuits, data path dominated circuits are chararcterized by much more regularity in their implementations. They are often implemented in an iterative or recursive way as e.g. 32-bit ALU. VLSI module libraries for such data path elements are usually realized as parameterized module generators [2]. The hardware modules generated by such module generators are characterized by *potential complexity* and by *parameterizability*: The size of the combinatorial blocks as generated by such module generators can give rise to unmanageble complexities for checking their correctness by tautology checking methods as described in subsection 3.2. Several data path dominated design problems expose regularity and are defined in a parameterized way. Starting from a generic description several design instances can be generated. This makes them very suitable for the *formal proofs by induction*, for which general purpose theorem provers can be used very well.

Within CHARME, formal correctness proofs of parameterized module generators [31] as well as regular hardware structures [32,33] based on the Boyer-Moore [60] theorem prover, have been worked out successfully.

The formal proof by means of the Boyer-Moore theorem prover [60] has been successfully integrated with a module generation environment as used by the CATHEDRAL silicon compiler [31]. This has up to now allowed the discovery of more than 25 design and specification bugs that were previously uncovered by traditional simulations on the designs.

Other theorem provers and proof assistants have been investigated for hardware verification and compared: OTTER [34,35], HOL[7] [59,36,37] and OBJ3 [38,62].

From the experience of the CHARME partners in the usage of general purpose theorem provers it has become clear that such tools have their specific strengths and weaknesses in comparison to more dedicated approaches such as those based on OBDD's and FSM verification as explained in subsections 3.2 and 3.3. General purpose provers have the advantage of *a uniform formalism*, the concept of *abstraction* and *proofs by induction*. The uniform formalism has the advantage that it allows reasoning in the formalism itself. But it has the disadvantage of being too general because they usually support the reasoning in some branch of mathematics instead of with the concepts and objects under design. To be useful in design this requires a policy for use of such provers. Even for theorem provers that have some automatic decision procedures, it has been experienced that a *lot of user interaction* and *system expertise* is required in the use of theorem provers. This means that in the application of theorem provers in hardware design, these tools should only be used where this investment can pay off. The aspect of *proofs by induction* is best exploited in parameterized hardware modules as well as in the correct definition of synthesis primitives. The aspect of abstraction can best be used at the higher level of design abstraction. Methodologies for design which rely on theorem provers should include the usage of theorem provers only at places where they do not require intervention of the day to day hardware designers.

3.5 Instruction Sets. (5)

As far as special purpose devices are concerned, between the SFG/Algorithmic level and the bRT level, control dominated subsystems require scheduling of the tasks to be executed in the hardware (data paths, 1/0 units, memories, etc.). This task is called scheduling [1,2].

On the other hand, with respect to microprocessors, the "instruction set" level is the highest level of abstraction and therefore an "instruction set" description can be considered as the microprocessor specification. The microprocessor implementation, that has to be checked versus this specification, can be described at various less abstract levels, from the "micro-sequence" level to the "data path" level [39,40]. Starting from the "instruction set" level, the proof consists in verifying, among two descriptions given at two adjacent levels, that the more detailed one is a correct implementation for the more abstract one. The lowest levels of description are written in VHDL, but for the highest ones, no appropriate single HDL exists. Therefore, in the CHARME project, a functional semantics [40] based on the **P**-calculus [64] has been defined for each of these highest levels [39,40]; these definitions have been made in accordance with the formalisms and the principles of the tools that are used to achieve the proofs. Because

7 cooperation with the CHEOPS ESPRIT-BRA 3215

of the complexity of this semantics, the realization of an associated hardware description language is not desirable; rather, a user-friendly interactive editor can help the designer in describing his/her microprocessor.

Such a special purpose interactive tool (with windows and menus), called μSPEED, has been developed for the specification and symbolic verification of (instruction sets of) microprocessors [41]. This tool constructs automatically the appropriate functional models of the microprocessors. This methodology has been used to successfully formulate the instruction sets of commercial microprocessors.

3.6 System Level Verification. (6)

In cooperation with research efforts in the CHEOPS project, a new system verification methodology called *SFG-Tracing* has been defined [42]. This verification methodology aims at the formal verification of designs across levels of abstraction. It is based on the observation that higher levels of abstraction are less detailed in their specifications (in terms of hardware and in terms of time instances). Therefore *SFG-Tracing* uses the higher level specification as a starting point and relies on the partitioning of this high level specification. In case of an SFG representation (but others are possible as well), the partitioning results in a number of boundary signal values, which are called reference *signals*. In the *SFG-Tracing methodology* it is required that the *mapping functions* in space and in time of these reference signals with respect to the lower level implementation are known. This methodology has resulted in the *full verification* of the transistor circuits as extracted from the layout with respect to the high level specification of a 32.000 transistor modem chip [58] as synthesized by CATHEDRAL-2 (see fig. 3).

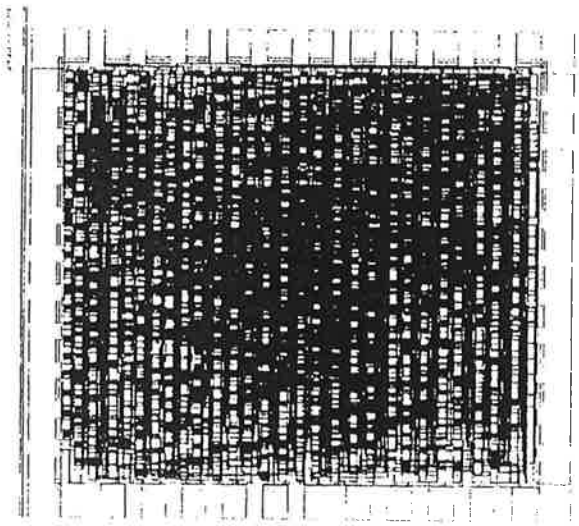

Figure 3: Chip layout of a 32000 transistor modem receiver pulse shaper and equalizer chip as synthesized by CATHEDRAL-2. This chip has 3 ALU's of 14 bits, register files, a multi-branch controller, micro sequencer, and testability circuitry. This chip has been fully verified w.r.t. high level specification using the *SFG- Tracing* methodology.

The complete interaction of sequential systems is captured in the *SFG-Tracing* verification methodology in contrast to only the individual combinatorial blocks at the sRT level as the methods explained in subsection 3.2. Due to the systematic partitioning of the specifications, *SFG-Tracing* does also not suffer from the complexity limits (around 60 state variables or 10^{20} states) as is the case in black box FSM verification

methods as explained in subsection 3.3. The chip in fig.3 contains 852 latches corresponding to 10^{256} states.

3.7 Asynchronous Subsystems. (7)

As explained under 2.1.1 almost all integrated circuits conumunicate with one another via asynchronous interfaces. The valid communication among such subsystems therefore needs verification. Whereas synchronous subsystems are more naturally described in *value-based* formalisms, asynchronous subsystems lend themselves more naturally to *event-based* formalisms. This is motivated by the asynchronous communication protocols.

In CHARME the *event-based* formalism CIRCAL [44] is being implemented for the formal verification of asynchronous subsystems [45] as well as its use for the integration in the synchronous circuits. CIRCAL is based on the concept of process algebras as it has been developed in the area of software engineering. Abstraction mechanisms for such an event based formalism have been worked out in CHARME [46]. Formal verification results are presented in [47].

3.8 Interface to VHDL as a Hardware Description Language. (8)

As the methodologies for formal verification are still under development, investigation, prototyping and comparison, currently all of the research efforts in CHARME as depicted in the *Formal Verification Map* of figure 2 are based on in-house hardware description languages and specific formalisms. This is motivated by the availability of previous research results and tools, the concentration on finding appropriate solutions of the design verification problem and by the available benchmarks.

However, to guarantee that what is being verified is indeed what has been designed, and gain acceptance in the community of hardware designers, formal verification should take as input the same user interface descriptions which are used for the other design tasks. VHDL [5] tends to become a "lingua franca" for hardware description in a significant part of the western world. Due to the considerable interest raised in the community by VHDL-based software, it is clear that dedicated efforts to concentrate on the formal verification of VHDL descriptions are required.

In CHARME, research work is being conducted to that goal. Unfortunately, VHDL is both a complex language, and a language for which no formal semantic definition is provided. Our pragmatic attitude has consisted in selecting a significant VHDL subset for which we have defined the semantics in terms of a formally manipulable model, and writing a translator from VHDL to the input format of provers for this model (this work has taken advantage of the translation principles presented in [61]). This work has led to the implementation of a prototype of Formal Proof Environment for VHDL, which is called PREVAIL [48,50], and which checks the functional equivalence between two "architecture" bodies, describing alternative implementation hypotheses, or different description levels, for the same design "entity", where one is considered as the specification, and the other one is the implementation. In order to keep the proof problem within manageable size, we take the well known "divide and conquer" strategy, and have identified modelling levels and circuit types, together with a preferred description style in VHDL, for which particular proof tools appear to be efficient. More specifically, this is done in the current status of our PREVAIL environment by defining:

- A syntactic subset of VHDL for Zero-delay combinational circuits; for this subset, a restricted and simplified semantics is associated to the VHDL architecture.
- A syntactic subset and description styles for the synchronous sequential circuits, and their associated functional semantics.

Nevertheless, the current status of this prototype does not allow the processing of the VHDL timing constructs such as "stable" (except for the master clock), "last-event", and thus avoids reasoning on asynchronous primitives. To that goal, recent work concentrates on the formal definition of the semantics of the VHDL timing constructs, and on the verification of associated properties (given in an informal way in the LRM) by means of the Boyer-Moore Theorem Prover [49].

The preliminary conclusions of all that work are that some concepts (such as memorization, FSM's) must be clearly defined in VHDL in order to make formal reasoning more manageable. This has already given rise to a number of recommendations for the 1992 ballot on the new VHDL standard.

4 Design for Verifiability.

Given the fact that formal verification still needed to go a long way to become useful in practice, the CHARME partners decided in 1988 to define guidelines under which formal verification of hardware could become practical. In analogy with *Design for Testability* (DfT) [6], this has been called *Design for Verifiability* by the partners.

DfV such as also DfT has to take into account the possibilities of the technology enabling formal verification. With respect to DfV there are basically two perspectives to look at the problem. One aspect is to define a number of rules or constraints to which designs have to adhere in order to become verifiable [51,53]. An other aspect is to concentrate on the verification *methodology* to be introduced in the design process [52,54].

5 Directions for Future Research.

The Formal Verification Map represents an overall classification of aspects and levels of abstraction on which formal verification tools are being worked out. As indicated in the previous section large progress has been made in nearly all of the classes in the last few years. However still a lot of efforts are required to extend the possibilities of formal verification as well as to introduce formal verification methods in the every day industrial hardware design process.

In the future, further consideration is necessary in order to automate the formal verification process. This requires further elaboration of the basic technology of Boolean comparison as outlined in section 3.2. Multipliers as often occur in VLSI hardware are still a problem [24] when represented as OBDD's. Initial approaches in this direction have been investigated already [55,63]. Although algorithms exist for generating the orders of variables in OBDD's [21,22] further research is required to exploit the regularity and structure of high level specifications in generating such orders.

In the automatic verification of combinatorial functions, further algorithms are required for exploiting the vector aspects in the expressions. Initial techniques in this direction have been worked out in CHARME [57].

Because the generated mask layouts are one of the formal hardware representations that are the nearest as a model to the actual circuits being realized, the formal verification starting from the layout extracted transistor circuits is extremely important. Therefore,

on the switch level of abstraction, future efforts have to concentrate on the efficient symbolic analysis and evaluation of transistor circuits with over 1.000.000 transistors.

As stated previously, the validation of the usage of the switch-level model deserves further attention in future research.

The validation of external properties of specifications, as also used in synthesis should be worked out further. Current approaches in model checking only concentrate on the verification of systems with around 60 state variables [29,30], which is still far from actual circuits. To cope with realistic complexities, more of the structure at hand in hardware specifications has to be exploited. In the industrial practice synthesis from high level specifications is becoming more and more accepted. This allows designers to make fast tradeoffs between different implementation alternatives. It should however be made sure that these specifications still meet the original requirements and specifications.

As general purpose theorem provers require a large amount of human expertise and interaction their usage should be restricted to those places where their use can really enlarge the quality of designs at hand as well as where "the regular designer" can be hidden from their intervention. This can be mainly exploited in the formal design of parameterized library entries as well as alternative synthesis primitives in libraries [56]. The aspect of abstraction is being exploited in transformational design systems.

Formal languages as used in theorem provers [59,60] can be used for the consistent formal definition of operator primitives that are used in CAD environments for purposes of *simulation*, *verification* and *synthesis*.

System level verification methodologies such as *SFG-Tracing* are as a methodology not restricted to the levels of abstraction to which they are currently used in the CHARME and CHEOPS projects. Indeed alternative representations such as VHDL and even 'C-code' models require future investigation.

In the implementation of complex systems, trade-offs have to be made on which parts to be made in hard- and which parts to be made in software. Future research has to concentrate on making this migration possible in an *efficient* and *correct* way.

VHDL is the HDL for the years to come. Even though reservations could be made with respect to certain aspects of the language, formal verification tools will have to adopt languages such as VHDL in order to be able to introduce formal verification methodologies in the actual hardware design trajectory.

In comparison to the synchronous subsystems, the aspect of the correct synthesis and verification of asynchronous subsystems needs further elaboration, because the correctness of large systems interconnected via asynchronous interfaces are critically dependent on the correctness of these interfaces for the correctness of the global systems.

Several of these future aspects will the be target of the CHARME-II project.

6. Conclusions.

In this paper an overview has been given of the broad field of formal verification in relation to design *aspects* and *abstraction levels*. This is necessary in order to understand and to be able to compare the specific approaches to address specific problem areas. A large progress has been made in the whole formal verification field. Having a mixed background cooperation in CHARME of electrical engineers, computer scientists and mathematicians has enabled to put together the appropriate ingredients to migrate formal verification from theory in industrial practice. This is already illustrated by the practical results of the full verification of actual VLSI chips from the transistor

level as extracted from the layout with respect to the high level specifications. This is *the largest full verification of a complete integrated circuit done thus far*. Previous approaches in the formal verification of e.g. microprocessors have all skipped important levels in the design abstractions. Also the adoption of designer oriented description languages such as VHDL are a key to introducing formal verification methods in the industrial practice.

In a fast growing field as formal design verification, the CHARME partners are stimulating the communication of researchers in the field by organizing state-of-the-art public workshops. These have already been organized in Darmstadt, Glasgow, Grenoble, Leuven and Turin. These meetings assist in the further comparison and synergy of promising approaches in the area of formal verification.

7. Acknowledgements.

The authors hereby thank the EEC for sponsoring this focused research project, which has enabled further progress and superior results, that would not have been possible in individual efforts only. They also would like to acknowledge the direct or indirect contributions of the following researchers to the implementatious and elaborations of the ideas and results being developed inside the CHARME project: M.Allemand, C.Angelo, A.Bailey, C.Bayol, P.Camurati, P.Cockshott, H.Collavizza, D. Deharbe, P. De Vijt, M.Genoe, M. Gilli, S.Hoereth, B.Huber, C. Le Faou, W.Lempens, W.Mao, T.Margaria, G.McCaskill, G.Milne, L.Pierre, W.Ploegaerts, F.Proesmans, A.Salem, H. Samsom, M. Sonza Reorda, U.Schellin, J.Vandenbergh, D.Verkest, E.Verlind. The authors thank the reviewers for their constructive comments.

References

[1] M.C. McFarland, A.C. Parker, R. Camposano, "The High-Level Synthesis of Digital Systems", *Proceedings of the IEEE,* Vol. 78, No. 2, February 1990, pp.301-318.

[21 H. De Man, J. Rabaey, P. Six, L. Claesen, "Cathedral-II: A silicon compiler for digital signal processing", *IEEE Design & Test of Computers,* December 1986, Vol. 3, No. 6, pp.73-85.

[3] J.N. Reed, A.W. Roscoe, "Technology Study: Formal Methods for the Development of Computer Systems", *EuroTechnology*, Issue No.9, April 1991, Blackwell Professional Information Services, ISSN 0959-7735, pp. 12-14.

[4] P. Camurati, P. Prinetto, "Formal verification of hardware correctness: introduction and survey of current research", *IEEE Computer*, July 1989, pp. 8-19.

[5] -, "IEEE Standard VHDL Language Reference Manual", IEEE Standard 1076-1987.

[6] H. Fujiwara, "Logic Testing and Design for Testability", *Computer System Series,* The MIT Press, ISBN 0-262-06096-5, 1985.

[7] R.E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems", *IEEE Transactions on Computers*, Vol. C-33, No.2, February 1984, pp. 160-177.

[8] R.E. Bryant, "Algorithmic aspects of symbolic switch network analysis", *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6, No. 4, July 1987, pp. 618-633.

[9] R.E. Bryant, "Boolean Analysis of MOS Circuits", *IEEE Transactions on Computer-Aided Design,* Vol. CAD-6, No. 4, July 1987, pp. 634-649.

[10] S. Hoereth, "Improving the performance of a BDD-based tautology checker", *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies,* ed. P.Prinetto, P. Camurati, Turin, June 12-14, 1991.

[11] P. Herrebout, "BOTRYS: A program for the symbolic analysis of MOS circuits at the switch level", Thesis IMEC - Katholieke Universiteit Leuven Belgium, July 1988.

[12] W. Lempens, "Symbolic analysis of digital MOS circuits at the switch level", Thesis IMEC Katholieke Universiteit Leuven Belgium, July 1989.

[13] R.E. Bryant, D. Beatty, K. Brace, K. Cho, T. Sheffer, "COSMOS: A Compiled Simulator for MOS Circuits", 24th Design Automation Conference, pp. 9-16, 1987.

[14] H. Eveking, "Behavioral consistency between register-transfer and switch-level descriptions", *Proc. IFIP TC-10 Working Conf. on Design Methodologies for VLSI and Computer Architecture*, pages 183-202, North-Holland, 1988.

[15] H. Eveking, "Behavioral verification of synchronous systems" in G. Milne and P.A. Subrahmanyam editors, *Formal Aspects of VLSI Design*, pages 137-152, North-Holland, 1985.

[16] H. Eveking, "Axiomatizing hardware description languages", *International Journal of VLSI Design*, Vol. 2, Nr. 3, 1990.

[17] D. Verkest, L. Claesen, "Special Benchmark Session on Tautology Checking", *Format VLSI Correctness Verification*, ISBN 0 444 88688 5, North-Holland Elsevier Science Publishers, 1990, p.81-82.

[18] C. Bayol, J.-L. Paillet, "Using TACHE for proving circuits", *Formal VLSI Correctness Verification*, ed. L.Claesen, ISBN 0 444 88688 5, North-Holland Elsevier Science Publishers, 1990, p.83-87.

[19] R.E. Bryant, "Graph Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, Vol. C-35 No. 8, August 1986, pp. 667-691.

[20] J.C. Madre, J.P. Billon, "Proving Circuit Correctness using Formal Comparison Between Expected and Extracted Behavior", *Proc. of the 25th Design Automation Conference*, 1988.

[21] M. Fujita, H. Fujisawa, N. Kawato, "Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams", *Proc. IEEE ICCAD-88 Conference*, 1988, pp. 2-5.

[22] S. Malik, A.R. Wang, R.K.Brayton, A. Sangiovanni-Vincentelli, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment", *Proc. IEEE ICCAD-88 Conference*, 1988, pp. 6-9.

[23] F. Anceau, *in panel session: Formal Hardware Verification: Myth or Reality*, EDAC-91, Amsterdam, February 25-28, 1991.

[24] R.E. Bryant, "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication", *report Carnegie Mellon University*, September 27, 1988. .

[25] P. Camurati, M. Gilli, P. Prinetto, M. Sonza Reodra, "Model Checking and Graph Theory in sequential ATPG", *Workshop on Computer-Aided Verification*, June 1990, Rutgers, NJ (USA).

[26] P. Camurati, e.a., "The Product Machine and Implicit Enumeration to prove FSMs Correct", *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies*, June 12-14, 1991, Turin, Italy, pp. 305-319.

[27] O. Coudert, C. Berthet, J-C. Madre, "Verification of Sequential Machines Using Functional Vectors", *Formal VLSI Correctness Verification*, ISBN 0 444 88688 5, North-Holland Elsevier Science Publishers, 1990, p.179-196.

[28] H. Eveking, "Experience in Designing Formally Verifiable HDL's", *Proc. Computer Hardware Description Languages and their Applications CHDL-91*, ed. D. Borrione, R. Waxman, Marseille, April 22-24, 1991, pp. 301.

[29] S. Bose, A. Fisher, "Automatic Verification of Synchronous Circuits using Symbolic Logic Simulation and Temporal Logic", *Formal VLSI Correctness Verification*, ISBN 0 444 88688 5, NorthHolland Elsevier Science Publishers, 1990, p.151-158.

[30] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, "Sequential Circuit Verification Using Symbolic Model Checking", *Proc. Design Automation Conference, DAC-90,* June 24-28, 1990, pp. 46-51.

[31] D. Verkest, L. Claesen, H. De Man, "Correctness proofs of parameterized hardware modules in the Cathedral-II synthesis environment", *Proc. European Design Automation Conference EDAC-90,* Glasgow 12-15 March 1990.

[32] L. Pierre, "The Formal Proof of Sequential Circuits described in CASCADE using the Boyer-Moore Theorem Prover", and "The Formal Proof of the "Min-max" sequential benchmark described in CASCADE using the Boyer-Moore Theorem Prover", *Formal VLSI Correctness Verification*, ISBN 0 444 88688 5, North-Holland Elsevier Science Publishers, 1990, p.309-348.

[33] L. Pierre, "One Aspect of Mechanizing Formal Proof of Hardware: The Generalization of Partial Specifications", *Proc. 1991 International Workshop on Format Methods in VLSI Design*, ed. P.A. Subrahmanyam, ACM/SIGDA, Miami Florida, January 9-11, 1991.

[34] P. Camurati, T. Margaria, P. Prinetto "Use of the OTTER theorem prover for the formal verification of hardware", Euromicro'90, August 1990, Amsterdam, (The Netherlands).

[35] P. Camurati, T. Margaria, P. Prinetto, "Resolution-based correctness proofs of synchronous circuits", *Proc. European Design Automation Conference EDAC-91*, IEEE Computer Science Press, Amsterdam, 25-28 February 1991, pp. 11-15.

[36] C.Angelo, L.Claesen, H.De Man, "A Methodology for Proving Correctness of Parameterized Hardware Modules in HOL", *proc. Tenth International Symposium on Computer Hardware Description Languages and their Applications*, CHDL-91, Marseille, April 22-24.

[37] C. Angelo, D. Verkest, L. Claesen, H. De Man, "On the comparison of HOL and Boyer-Moore for formal hardware verification", *proc. Advanced Workshop on Correct Hardware Design Methodologies,* ed. P.Camurati, P.Prinetto, Turin, June 12-14.

[38] H. Collavizza, L. Pierre, "Basic Verification Techniques - Evaluation of General Provers", *ESPRIT CHARME Report: UP-2.A.2.-OI,* 30 June 1990.

[39] H. Collavizza, "Functional Semantics of Microprocessors at the Micro-Program level and Correspondance with the Machine Instruction Level", *proc. of the EDAC Conf.,* Scotland, 12-15 March 1990, pp. 52-56.

[40] J.L. Paillet, "Functional Semantics of Microprocessors at the Machine Instruction Level", *Proc. 9th IFIP Int. Conf. CHDL,* North-Holland, Washington D.C., June 1989.

[41] D. Borrione, H. Collavizza, C. Le Faou, "μSPEED: a Framework for Specifying and Verifying Microprocessors", *Proc. 1991 International Workshop on Formal Methods in VLSI Design*, ACM/SIGDA, Miami, January 9-11, 1991.

[42] L.Claesen, F.Proesmans, E.Verlind, H.De Man, "SFG-Tracing: a Methodology for the Automatic Verification of MOS Transistor Level Implementations from High Level Behavioral Specifications", *Proceedings A CM-SIGDA International Workshop on Formal Methods in VLSI Design,* ed. P.A. Subrahmanyam, January 9-11, 1991.

[43] M. Genoe, L. Claesen, E. Proesmans, E. Verlind, H. De Man, "Illustration of the *SFG-Tracing* Multi-Level Behavioral Verification Methodology, by the Correctness Proof of a High to Low Level Synthesis Application in CATHEDRAL-11", *Proc. IEEE ICCD-91,* Conference, Cambridge MA, October 14-16, 1991.

[44] G.J. Milne, "Circal and the representation of communication, concurrency and time", *ACM Trans. on Programming Languages and Systems*, 7(2), 1985.

[45] A. Bailey, G. Milne, "Using CIRCAL to Analyse Sutherland's Asynchronous Micropipeline Design Style", *Dept. of Computer Science Research Report*, HDV-13-91, University of Strathclyde, UK, May 1991.

[46] A. Bailey, "Abstraction Mechanisms for Hardware Verification: Formalisation in a Process Algebra", *Proc. CHDL-91,* editors: D. Borrione, R. Waxman, Marseille France, April 22-24,1991.

[47] A. Bailey, G.A. McCaskill, J. McIntosh, G.J. Milne, "The description and automated verification of digital circuits in Circal", Proc. *Advanced Research Workshop on Correct Hardware Design Methodologies,* ed. P.Prinetto, P.Camurati, Turin, June 12-14, 1991.

[48] D. Borrionc, A. Salem, "Design For Verifiability - Automatic Formal Verification of VHDL descriptions", *ESPRIT CHARME report UP-I.C-01,* 30 June 1990.

[49] A. Salem, D. Borrione, "Formal Semantics of VHDL Timing Constructs", *Proc. EURO-VHDL'91,* Stockholm, 8-11 September 1991.

[50] D. Borrione, L. Pierre, A. Salem, "PREVAIL: A Proof Environment for VHDL Descriptions", *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies,* ed. P. Prinetto, P. Camurati, Turin, June 12-14, 1991.

[51] G. Milne, "Design for Verifiability", In *Proc. Workshop on Hardware Specification, Verification and Synthesis: Mathematical Aspects*, Cornell Univ. 1989.

[52] H. Eveking, "Priliminary Concepts of Design for Verifiability", *ESPRIT CHARME Report THDI.C-01*, July 20, 1990.

[53] P. Camurati, P. Prinetto, "Design for Verifiability and Design for Testability: limiting designers' freedom to achieve what?", *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies,* ed. P. Prinetto, P. Camurati, Turin, June 12-14, 1991.

[54] L. Claesen, M. Genoe, E. Verlind, F. Proesmans, H. De Man, "*SFG-Tracing:* a methodology of Design for Verifiability", Proc. *Advanced Research Workshop on Correct Hardware Design Methodologies,* ed. P. Prinetto, P. Camurati, Turin, June 12-14, 1991.

[55] H. Simonis, "Formal Verification of Multipliers", *Formal VLSI Correctness Verification*, ISBN 0 444 88688 5, North-Holland Elsevier Science Publishers, 1990, p.267-286.

[56] D. Verkest, J. Vandenbergh, L. Claesen, H. De Man, "Formal Design and Verification Strategy of Parameterized Hardware Modules", *Proc. Advanced Research Workshop on Correct Hardware Design Methodologies,* ed. P. Prinetto, P. Camurati, Turin, June 12-14, 1991.

[57] A. Bratch, H. Eveking, H.-J. Faerber, J. Pinder, U. Schellin, "LOVERT - A Logic Verifier of Register Transfer Level Descriptions", *Formal VLSI Correctness Verification*, ISBN 0 444 88688 5, North-Holland Elsevier Science Publishers, 1990, p.247-256.

[58] J.Vanhoof, I.Bolsens, S.De Troch, E.Blokken, H.De Man, "Evaluation of high-level design decisions using the Cathedral-11 silicon compiler to prototype a DSP ASIC", *Proceedings, IFIP Workshop on High Level and Logic Synthesis,* ed. G. Saucier, Paris, 30 May-1 June 1990.

[59] M.J.C. Gordon, "HOL: A Proof Generating System for Higer-Order Logic", *in "VLSI Specification, Verification and Synthesis"* ' editors: G. Birtwistle and P.A. Subrahmanyam, Kluwer 1987.

[60] R.S. Boyer, J.S. Moore, "A Computational Logic Handbook", *Academic Press*, Boston, 1988.

[61] L. Pierre, "From a HDL Description to Formal Proof Systems : Principles and Mechanization", *Proc. 10th IFIP Int. Conf. CHDL'91,* Ed. D.Borrione & R.Waxman, Marseille, 22-24 April 1991.

[62] J. Goguen, "OBJ as a Theorem Prover with Applications to Hardware Verification", *Technical report SRI-CSL-88-4R2, Computer Science Laboratory, SRI International*, Menlo Park, August 1988.

[63] J. Burch, "Using BDDs to Verify Multipliers", *Proc. 28th ACM/IEEE Design Automation Conference*, San Francisco (CA), 17-21 June 1991.

[64] J.L. Paillet, "A Functional Model for Descriptions and Specifications of Digital Devices", *Proc. IFIP Int. Working Conf. "From HDL Descriptions to Guaranteed Correct Circuit Designs"*, Grenoble, September 1986.

[65] D. Borrione, D. Deharbe, H. Eveking, St. Horeth, "Application of a BDD-package to the verification of HDL-descriptions", *Proc. Advanced Research Workshop on Correct Hardware Design Methodology,* Turin, 12-14 June 1991.