A CAD tool for Automatic *Core* Interface Generation in a SoPC Environment

Julio A. de Oliviera Filho, Manoel E. de Lima
Centro de Informática, Universidade Federal de Pernambuco
jaof@cin.ufpe.br, mel@cin.ufpe.br

Abstract

System on a Programmable Chip (SoPC) has become a reality, facing design of complex circuits into a single programmable device, supporting different cores for microprocessors, interface, bus, etc. However, the automatic inclusion of new general cores from different providers via a standard bus still needs a reliable interface mechanism to guarantee the correct protocol conversion and performance. This work presents a CAD tool methodology to cope this problem based on a Petri Net protocol conversion approach in a high level behavioral specification, focusing on bus planning and core integration.

1. Introduction

Chip gate count fast increasing has led to entire systems being implemented on a single die (SoPC), and so, design complexity has grown proportionally making difficult to achieve short time-to-market needs. To cope such problems, emergent technology methodologies and CAD tools emphasizing system-level synthesis and reusability have been proposed, forcing fundamental changes in the way systems are designed.

Reusability approach is mainly based on the use of pre-designed IP blocks(henceforth called *cores*) and has become essential in order to build the required complexity in a short time-to-market. However, in practice, the task of quickly assembling an SoPC has not yet become reality for various reasons, including:

- Lack of efficient tools for *core* integration into SoPC. Nowadays, *core* integration on SoPC design is mainly a manual and error-prone activity, forcing designers to deal directly with functionality, interfaces and electrical characteristics of complex and custom *cores*.
- Physical design complexity. Putting *cores* together can be a challenging task as unforeseen effects such as noise and coupled capacities can affect performance.
- Design for reusability should be concerned to integrate IPs from different vendors, but there are no established standards or efficient interface synthesis tools. Moreover, various interface issues, e.g. timing, may cause systems to fail even if the *cores* are individually correct.
- Interface between hardware and software affects directly time-to-market because, in major design frameworks, the integration among those is done later when hardware part is more stable.

Some groups efforts, like VSI Alliance[1] are oriented towards facilitating integration of *cores* through standardization. SoPC bus standards, such as AMBA[13], WishBone[14] and AVALON[12] are as well arising in market to ease *core* interchange and reuse. However, integration task today is a complex process because it requires the designer to understand details in hundred of pins in various *cores* and generating correct additional logic for their interconnection. While design methodologies migrate to system-level algorithmic and architectural bases and hardware-software co-design technologies, it is imperative to build tools focused on bus planning and block integration [2].

This work addresses the implementation of a CAD tool methodology for *core* interface automatic generation and *core* integration in a SoPC design, based on a Petri Net approach. From the description of timing diagrams, bus specification, pins functionality and some system information it is possible to automatically synthesize adequate interface process circuit. Related works are presented in the section 2. Section 3 explains main streams of the framework and the Section 4 discusses some initial conclusions.

2. Related Work

Several research efforts treat interface automatic generation as transducers synthesis. It is indeed an efficient approach, however, simple protocol conversion on transducers circuit hardly addresses system level information like type conversion and data width matching.

Daniel D. Gajski[3] proposes a method for generating interface process based on HDL description of two communicating protocols. HDL descriptions, however, makes difficult to estimate circuit issues (like area and power) and formally verify process functionality.

Cortadella[4] et. al. uses a formal theory and mathematical model based on a Petri Net for interface specification. In his work, techniques for synthesis, analysis and formal verification are being largely studied and can lead interface process generation task through an asynchronous circuit design approach. Bill Lin[5] addresses automatic protocol conversion and Passerone[8] proposes interface process construction through a state machine approach.

Other works analyze *core* integration into system level. Bergamaschi[6] discusses relevant info from system specification that should be available during *core* integration task. The same is extensively discussed by Henry Chang et. al.[2] while proposing a methodology for SoPC design based on reusability and standardization. Both works show how this problem is being addressed in strong market groups like IBM[7] and VSI Alliance[1]. None of these works, however, unify interface automatic generation and system-level information into a *core* integration-driven SoPC design framework.

3. CAD Tool Methodology

Core interface protocols are typically documented as some sort of timing diagrams in data sheets and technical manuals. For this reason a slightly different type of timing diagram, called Signal Transition Graph (STG), is used as the input for the proposed CAD tool. This type of annotated timing diagram was introduced by Chu[9] and is equivalent to a free-choice Petri Net model, where transitions represent rising and falling signal edges as well as data assertion/de-assertion. Unlike timing diagrams, STGs may express choice and non-determinism in a concurrent view. Figure 1 depicts a two phase writing operation described as an STG and its equivalent Petri Net graph. In the figure, data assertion(de-assertion) is represented as *D# (D*)*, a signal name followed by *+(-)* means rising(falling) transition on that signal and the input signals are underlined. After data assertion, a strobe signal is enabled to pulse and data can be de-asserted so forth.

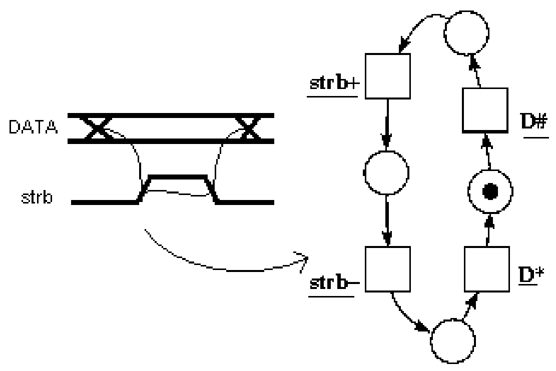

Figure 1. – STG

Based on this approach, the Figure 2 shows the CAD tool design flow for interconnecting two different modules interface protocol. In the graphic, the destination module has a two phase protocol interface(2a) while the source one uses a four phase protocol(2b) and so, a direct connection between the two entities is not possible. From system specification the designer can build an STG describing only the necessary communication tasks, as removing unnecessary operations leads to power and area saving. A STG for each communicating module is provided.

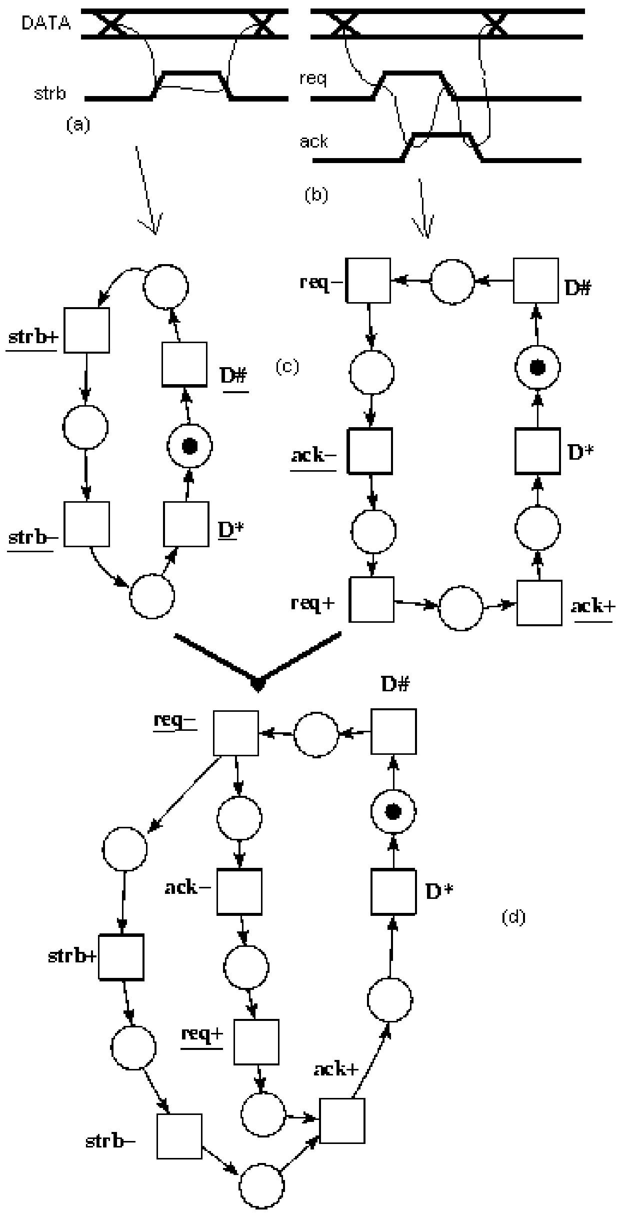

Figure 2 – CAD Tool workflow

The CAD tool accepts STGs as inputs and automatically translate them into Petri Net notations(2c).

The Petri Nets can then be adequately manipulated to systematically introduce information available from high-level specification. Examples for those information are sequences of data transfers, data ports interconnection, address decoding and data-width mismatches. These aspects are introduced through sequential transformations on Petri Net structure.

Once the Petri Nets describing interface protocols plus system information are ready for both modules, they are joined into one unique net in a two step stage. Initially a each Petri Net of each protocol is mirrored. A mirror operation consists in re-building the same net but making input(output) signals as outputs(inputs). After that, the protocols are then joined through a parallel composition operation[5] between the mirrored nets. The composed single graph represents the interface process between the two communicating modules as depicted in figure (2d). Figure 3 shows a mirror operation example.

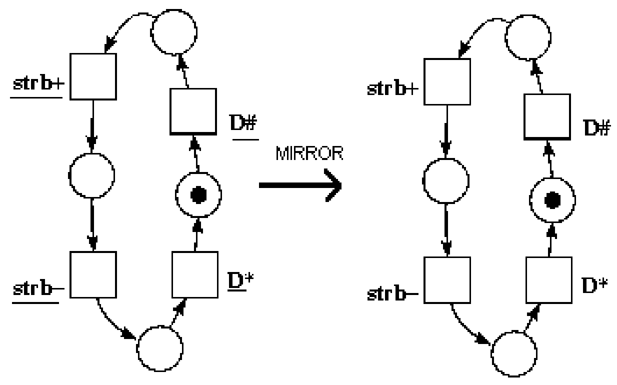

Figure 3 – MIRROR Operation

The generated Petri Net representation is then pre-processed to ensure that some required properties, such as *boundedness, consistency, persistency and completeness*, are verified. *Boundedness* is used to guarantee a finite state graph for the Petri Nets. *Consistency* is applied to make sure every rising(falling) transition on a signal is alternated. *Completeness* ensures there are no states with the same encoding and different behavior on output, and finally, *persistency* avoids logic hazards during circuit operation. Other properties can be of interest as well, e.g., absence of deadlocks and fairness. These properties are important to garantee an harzard free implementation of the circuit. If the above properties are satisfied, then the specification can be implemented as a *speed-independent* circuit [4].

The resulting Petri Net is then translated into a synthetisizable VHDL. The goal of circuit synthesis from the Petri Net description, is to derive a gate netlist that implements the interface behavior defined by the specification. The VHDL component is generated by attaching structurally the *core* code with the interface process. The new component is then ready for plug-and-play into the system. Figure 4 shows the conversion and synthesis results.

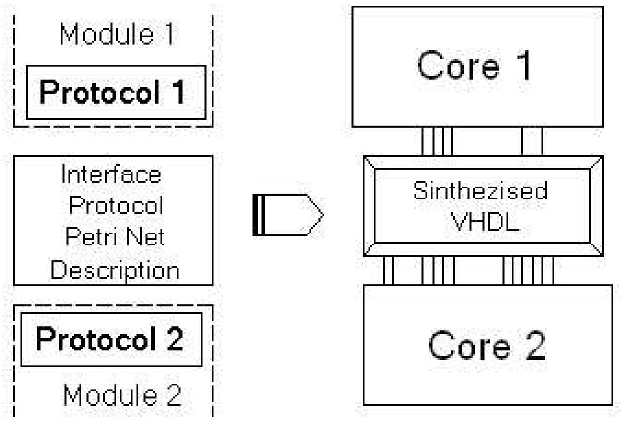

Figure 4 – CAD Tool Result

This method can be applied between any two communicating modules using different protocols. In this particular work, we intend to emphasize block integration onto SoPC standard bus architecture, such as AVALON and AMBA, wide spread architectures in processor/peripheral and bus-driven systems.

4. Conclusions

The CAD tool methodology proposal for automatic *core* interface generation is being implemented as a plug-in for the Altera´s SOPC Builder[10] in the Quartus II[11] management environment. A real case study is under development. The methodology aims at *core* integration into design by automatic protocol match between *core* and bus in a system level specification. The CAD tool will be able to gather information from the system specification and systematically introduce it into interface process generation based on a Petri Net approach.

5. References

[1] *VSI Alliance Architecture Document*, Version 1.0, VSI Alliance, 1997.
[2] H. Chang et. al., *Surviving the SoC Revolution – A Guide to Plataform Based Design*, Kluwer Academic Publishers, 1999.
[3] D.D.Gajski and S. Narayan, *Interfacing Incompatible Protocols using Interface Process Generation*, 32nd ACM/IEEE Design Automation Conference, 1995
[4] J.Cortadella, M.Kishinevsky, et. al., *Hardware and Petri Nets: Application to Asynchronous Circuit Design*,

21st International Conference on Application and Theory of Petri Nets, 2000.
[5] G. G. Jong , B. Lin, *A Communicating Petri Net Model for the Design of Concurrent Asynchronous Modules*, 31st ACM/IEEE Design Automatic Conference, 1994.
[6] RA.Bergamaschi,WR.Lee,*Designing Systems on Chip using Cores*, DAC2000, LosAngeles,CA.
[7] *Blue Logic Technology*, IBM.
[8] R. Passerone, J.A. Rowson, *Automatic Synthesis of Interfaces between Incompatible Protocol*., DAC98, San Francisco,CA.
[9] T. Chu. *Synthesis of Self-timed VLSI Circuits from Graph Theoretic Specifications*. PhD thesis, Department of EECS, Massachusetts Institute of Technology, September 1987.
[10] Altera Inc., *SOPC Builder*, version 2.0, January 2002.
[11] Altera Inc., *QUARTUS II Manual*, http://www.altera.com
[12] Altera Inc., *AVALON Bus Specification – Reference Manual*, version 2.0, January, 2002.
[13] ARM Inc., *Amba 2.0 Specification*, http://www.arm.com
[14] Silicore Corporation, *WISHBONE, Revision B.1 Specification,* http://www.silicore.net/wishbone.htm