IP Core Synthesis in a Cloud

Anatolij Sergiyenko , Valerij Simonenko, Yourij Vinogradov, Konstantin Gluchenko

National Technical University of Ukraine, pr.Peremogy,37, 03056 Kiev, Ukraine

aser@comsys.kpi.ua

Abstract. *The approach to design of the system for IP core synthesis in a cloud is proposed, which based on the XML data representation and graph drawing which uses SVG. The approach is proven in a framework, which is intended for the SDF algorithm graph input, its graphical editing and sending to a cloud. The result of the framework operation is the optimized pipelined IP core, which is described by VHDL, and is ready to be modeled and synthesized using traditional CAD tools.*

Keywords

FPGA, SVG, XML, SDF, VHDL, IP core, cloud computing, algorithm mapping.

1 Introduction

The modern technology of the integral circuit design is based on the reuse of the intellectual property (IP) cores. The proper IP core must be correct, reliable, adaptable, to be built in a new project without problems. Therefore, the design of a new IP core is hard and responsible task. The soft IP core is represented by a description of the unit at the RTL level by the Verilog, or VHDL language. Such a core can be attached to any project, which is implemented by any technology including FPGA configuring.

The process of IP core design has a set of stages. Firstly, the algorithm of the IP core behavior is searched and described by the high level hardware description language in the RTL style. Then this description is modeled using the proper logic simulator. The final IP core is synthesized, placed and routed to estimate its parameters, which are optimized. The two last stages are usually computation intensive. This process is usually repeated iteratively [1]. Last time the initial algorithm is described at the system level and then is compiled to the RTL description. But this compilation affords a lot of optimization efforts to satisfy a set of requirements to the resulting IP core. Therefore this process is computation intensive as well [2].

Last time many companies, which provide CAD tools for the integral circuit design, propose their service, which is placed in a cloud [3],[4]. Among them Aldec (logic simulation), Nimbic (3D simulation), Tabula (FPGA synthesis), Cadence (reference flow), Synopsys (logic simulation in AWS), and others have a success. The IP core design, which is based on the cloud computing, provides a set of benefits like cost minimization, ready-to-use and reliable infrastructure, data security, etc. But the present cloud services deal only with IP core verification, placing and routing, project handling. In the presentation an approach is proposed, which provides the IP core high level synthesis as well.

2 Synthesis of pipelined IP cores

During the high-level synthesis the recursive algorithm is usually represented by the synchronous data flow graph (SDF). The nodes of SDF, which are named as actors, represent the algorithm operators. The edges represent variable movings between actors with the FIFO buffering. During a single cycle of SDF modeling each actor generates and accepts a variable (token) set, which quantity is stable in each cycle [5],[6],[7].

Such programming tools like AccelDSP, or System Generator are widely spreaded, which help to generate the FPGA configuration of the DSP computing system. Here the initial algorithm is represented by SDF using Matlab Simulink package [8]. But these tools are effective only for acyclic SDF, for example, FIR-filters.

In [9],[10] a method for mapping SDF into parallel computing structures is proposed. It is based on placing the graph in the multidimensional index space and mapping it in subspace of structures and subspace of events. This method implements simultaneously the steps of resource selection, operator scheduling, and resource assignment. As a result, it optimises much better the structure of the computing system.

In this method the initial data for the design are recurrent algorithm, represented by SDF, and cost function. The SDF with a single sampling frequency i.e. the homogeneous SDF is considered. The computing system structure consists of a set of procesing units (Pus), connected to each other according to the structure graph. A single PU consists of ALU of given type (multiplier, adder, etc.) with the result register, or FIFO buffer on its output and with the input data multiplexers on its inputs. PU calculates the operation no longer than a single clock cycle. Such a simple PU can be naturally implemented in the FPGA resources including special hardware like DSP48 units in Xilinx FPGA devices.

Firstly, SDF is represented in the three dimensional index space as the algorithm configuration $\mathsf{K}_G = (K,D,A)$, where K – is the matrix of vectors-nodes $\boldsymbol{K}_i$, representing algorithm operators, D is the matrix of vectors-edges $\boldsymbol{D}_j$, representing variable movings between operators, A is the incident matrix of SDF. The coordinates of the node $\boldsymbol{K}_i = <k_i,s_i,t_i>$ are equal to the operator type, PU number s_i, in which the operator is implemented, and clock cycle t_i, when the operator result is stored in the register.

Equivalent structural solutions are represented by equivalent algorithm configurations, which differ in its matrices K, and D. The matrix K means some correct structure solution, and the matrix D can be derived from the equation $D = KA$. The optimum structural solution finding consists in the search of the matrix K, which minimizes the given cost criteria. For example, if the genetic optimization approach is used then the matrix K serves as a genome of the population representative.

A set of relations, definitions, and theorems help to find the effective solutions. For example, the algorithm configuration is correct if there is none couple of equal vectors in the matrix K, i.e.

$$\forall\, \boldsymbol{K}_i,\boldsymbol{K}_j\ (\boldsymbol{K}_i \neq \boldsymbol{K}_j,\ i \neq j\ ).$$

The operator schedule is correct if the operators that are mapped in a single PU are implemented in different clock cycles, i.e.

$$\forall\, \boldsymbol{K}_i,\boldsymbol{K}_j (k_i=k_j,\ s_i=s_j\ ) \Rightarrow t_i \not\equiv t_j \bmod L.$$

Here a modulo operation represents the cyclic nature of SDF modeling. This helps to derive the cyclic operator scheduling.

An effective algorithm configuration is searched in two steps. On the first step SDF nodes and edges are placed in the three dimensional space as sets of vectors $\boldsymbol{K}_i$ and $\boldsymbol{D}_j$ with respect to all conditions mentioned above. Then the PU number is minimized, when the number of nodes, mapped into a single PU, approaches to L.

On the second step the configuration balancing is implemented. During this procedure the acyclic subgraph of SDF is considered, and in each edge the delay nodes are added. These nodes represent a delay, or storing for a single clock cycle, or a register. In the resulting balanced configuration all the edges except edges of the feedbacks are equal to $\boldsymbol{D}_j=<a_j,b_j,1>$ or $\boldsymbol{D}_j=<a_j,b_j,0>$. The configuration nodes form columns, and the distance between adjacent columns is equal to one clock cycle. The balanced configuration is optimized by mutual interchanges of nodes belonging to a single column. By this process, the register and multiplexor number is minimized. The other methods like resynchronization, left edge scheduling are used as well.

The computing system structure and the resulting schedule are derived by splitting the algorithm configuration K_G to the structure configuration K_S and event configuration, which have the same incidence matrix A. The vectors of the structure configuration are equal to $<k_i,s_i>$, i.e. to the type and number of PU, and the vectors of event configuration are equal to $<t_i>$, i.e. to the clock cycle, in which the mapped operator is implemented. In such a way the algorithm configuration is mapped into the structure subspace and into the event subspace.

This method is well suited for the pipelined IP core design. The resulting IP core can be optimized both manually and automatically using different heuristic and exact combinatorial optimization methods. It was selected for the implementation in a cloud.

3 System for the IP core design

The usual software as a service cloud system is built according to the client-server paradigm. In the client side the internet browser serves for initial data input and result representation, and in the server side the special software solves the given user problem. The IP core design system must provide the interactive mode of operation. This means that the user has not to waist the time waiting to the response of the system.

The main source of delays in the cloud computing are the delays of transactions in the communication channels and delays of solving of computation intensive tasks. To minimize these delays it is preferable to distinguish two kinds of tasks. One task provides the input and output of initial data and results, which affords frequent interactions with the user, and therefore, it is implemented in the client side. Another task realizes the computational intensive IP core synthesis with optimization in the server processors. The data movings between these tasks have the delay, which is

dependent on the communication channel delays and on the cloud computer speed, and it can be much longer than the delay of interactions with the user in the client side.

To realize the method for mapping SDF into parallel computing structures using a cloud the following features are proposed. The initial, intermediate, and resulting data are represented by the XML files. The XML format provides the standardized representation both algorithms and resulting structures, and is native for the cloud computations. The internet browsers in the client side and server engines as well as the libraries of programming languages provide effective parsing of XML files. Many CAD tools already use XML format to store and expand their data [11].

It is useful to represent graphically both algorithms and resulting structures using the scalable vector graphic (SVG) format. This format is standardized, and is implemented in most of Internet browsers. It can be built in the XML files as well. There are many free tools and libraries, which support this format. The effectiveness of the SVG format in the electronic CAD tools, which use clouds, is shown in [12].

The working language to describe the CAD tools is Java. This language is usual one both in client and in server sides. It can provide parallel computations in the cloud computer as well, which serves to minimize the response time. There are free libraries in Java to implement both XML and SVG formats.

At present, the client side of the system for mapping SDF into IP cores is developed. The framework, which intended for the SDF graph input, its graphical editing and sending to a cloud is written in Java. Both algoritm and resulting structure are stored in XML files. The component library is stored in XML files as well. The components of the library are designed in the component editor, which provides the VHDL description with generic constants and different data representations. Among them integers, fixed point and real data representations are provided.

The framework can translate the XML description into the VHDL synthesable model, which can be modeled and synthesized by usual CAD tools provided by different companies. The present limitation consists in that, that the SDF graph is optimized only by hand using the relations, definitions, and theorems mentioned above.

In the Fig.1 an example of the SDF of a IIR filter, and respective structure are shown. This algorithm calculates a simple equation: $y_i = x_i - x_{i-1} + y_{i-1} + y_{i-2}$. In the Fig.1 the empty circle, colored circle represent input and output nodes, sign ◐ represents a node of the feedback link, respectively. Rectangles mean addition operators, straigt lines and dotted arrows represent usual data and delayed data exchange, respectively. The vertical lines mean the clock cycles, and horizontal ones sign the resources. The time marks are given by modulo L, which show the iterative nature of the algorithm.

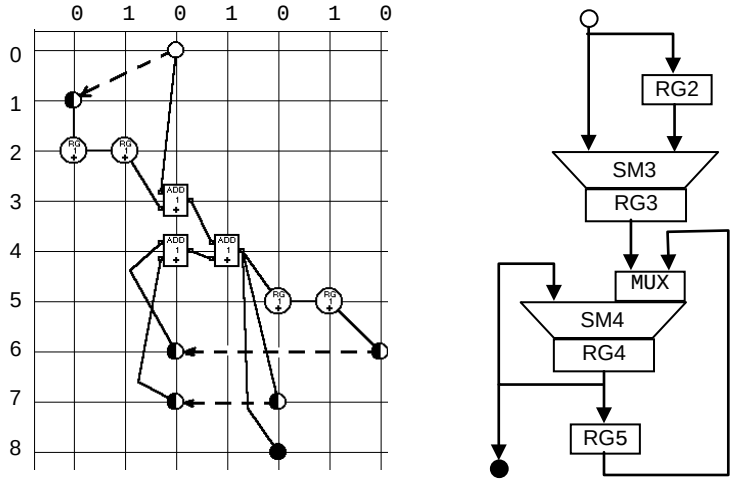

Fig.1. SDF of balanced algorithm configuration, and respective IIR filter structure by the clock period L=2

The resulting structure in Fig.1 is given for the reference because the result of the synthesis is the synthesable VHDL description. As one can see, the algorithm configuration in the Fig.1 is optimized one, because the resource loading is well balanced, and the algorithm is implemented in the pipelined structure with the minimized clock cycle.

This framework was used for development of a set of IP cores for DSP applications. Among them FFT and DCT processors, IIR filters are, which showed exellent characteristics when they are configured in FPGA [13], [14]. The projects of these IP cores are stored in the opencores.org site for the free use.

4 Conclusion and future works

The approach to design the system for IP core synthesis in a cloud is proposed, which is based on the XML data representation and graph drawing which uses SVG. The approach is proven in a framework, which is intended for the SDF graph input, its graphical editing and sending to a cloud. The result of the framework operation is the optimized pipelined IP core, which is described by VHDL, and is ready to be modeled and synthesized using traditional CAD tools.

The program, which automatically optimizes the SDF description, and is running in a cloud, will be designed in the next step of our work. The program will implement a set of optimization techniques, among them, left edge scheduling, branch and bound, genetic approach, and other combinatorial optimization techniques will take place. The cloud computing is planned to be implemented in the HP Blade server C3000, installed in NTUU „KPI“.

References

[1] C. Maxfield: The Design Warrior’s Guide to FPGAs. *Newnes, Elsevier*, 542 p. 2004.
[2] High-Level Synthesis. from Algorithm to Digital Circuit: *Coussy, P., Morawiec, A.(Eds.), Springer*. 2008.
[3] A.Vakali, L.C.Jain: New Directions in Web Data Management. Springer-Verlag, Berlin Heidelberg, 347p. 2011.
[4] J.S. Chee, C. Franklin: Cloud Computing. Technologies and Strategies of the Ubiquitous Data Center. *CRC Press, Taylor & Francis Group*, 263 p. 2010.
[5] S. S. Bhattacharyya, R. Leupers, P.Marwedel: Software Synthesis and Code Generation for Signal Processing Systems. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 47(9): 849-875. 2000.
[6] The Systhesis Approach to Digital System Design. *Ed. P. Michel, U.Lauther, P.Duzy, Kluwer Academic Pub.*, 1992.
[7] А.М. Сергиенко, В.П. Симоненко: Алгоритмические модели обработки потоков данных. *Электронное моделирование*. 30(6), 49–60. 2008.
[8] System Generator for DSP. Getting Started Guide. August, 2007, -85p. Available at http://www.xilinx.com.
[9] A. Sergiyenko, O. Maslennikow: Mapping DSP algorithms into FPGA. *Proc. IEEE Int. Symposium on Parallel Computing in Electrical Engineering, PARELEC'06. –Poland: Bialystok, 13-17 Sept., 2006*. 208-213. 2006.
[10] Сергиенко А.М. Отображение периодических алгоритмов в программируемые логические интегральные схемы / А.М. Сергиенко, В.П. Симоненко // Электронное моделирование. –2007. –Т.29. –№2. –С. 49–61.
[11] V.Berman: Standards: The P1685 IP-XACT IP Metadata Standard, in*: Design & Test of Computers, IEEE*, 23(4): 316–317, 2006.
[12] S. Dawson, P.S. Sezer: SVG for Remote EDA Schematic Representation. *SVG Open 2004, 3rd Conf. on Scalable Vector Graphics.* Available at http://www.svgopen.org/2004/papers/SVGforEDASchematics/ .
[13] А.М.Сергиенко, Т.М. Лесик: Динамически перестраиваемые цифровые фильтры на ПЛИС. *Электронное моделирование.* 32(6):47-56. 2010.
[14] А.М. Сергієнко, В.Л. Лепеха, Т.М. Лесик: Спецпроцесори для двовимірного дискретного косинусного перетворення. *Вісник НТУУ “КПІ”:“Інформатика і обчислювальна техніка”, зб. наук. праць*. 47, 49-52. 2007.