Towards novel format for representation of polymorphic circuits

Adam Crha, Václav Šimek, Richard Růžička
Faculty of Information Technology, Brno University of Technology
Božetěchova 2, Brno, 612 66, Czech republic
icrha@fit.vutbr.cz, simekv@fit.vutbr.cz, ruzicka@fit.vutbr.cz

***Abstract*—This paper is focused on the introduction of a novel format for representation of complex polymorphic circuits. Core of this newly presented format is based on the exploitation of And-Inverter Graph (AIG) scheme with a number of extensions. Then, synthesis flow can exploit considerable advantages in terms of the capability to handle at ease more extensive circuit structures involving hundreds of gates and employ subsequent optimization techniques resulting in an improved area-efficiency performance. The actual format notation, explanation of its visual features appearance meaning and comparison with selected conventional approaches are presented. Finally, accomplished experimental results and their analysis is provided.**

***Index Terms*—Polymorphic electronics, synthesis techniques, And-Inverter graph, circuit representation.**

I. Introduction

Contemporary state-of-the-art paradigms, design approaches and fabrication procedures are getting closer to the ultimate technological constraints, which are associated with the physical foundations of the widespread conventional CMOS process. A specific approach how to address these peculiar needs, at least in certain situations, could be based upon the principles of so-called polymorphic or multifunctional electronics [1].

Main idea behind the polymorphic electronics approach is connected with a circuit structure that is able to perform more than one intended function. It is characteristic that the interconnection of the circuit components (gates) remains unchanged in contrast to the conventional electronics where the function must be explicitly hard-wired.

Unfortunately, conventional design methods and algorithms are not directly applicable for a design of polymorphic circuits without the need to implement major changes. Besides various design and synthesis techniques themselves, one of the key aspect is also defined by the availability of suitable format that makes it feasible to efficiently capture the structure of even complex polymorphic circuits involving hundreds of gates. It is essential for convenient handling of those circuits from the perspective of synthesis tasks, minimization operations, simulation of their behaviour and many other.

II. Concept of polymorphic electronics

It is important to emphasize at this point that all the required circuit functions are designed in a fully intentional manner rather than, for example, as a specific fault condition caused by exceeding certain operating parameters of the circuit. State of the environment, where such circuitry is going to be deployed, can be accurately expressed through a physical quantity with a direct impact on the electrical properties of circuit building elements. Then, it is possible to clearly determine the actual function to be realized by that circuit according to the specific value of a relevant parameter [2]. Such behaviour is useful for circuits that must adapt itself to unfriendly environment, e.g. by imposing restriction of power consumption [3].

III. Elements of And-Inverter Graph scheme

One of the most ubiquitous schemes used for representation of a conventional digital circuit is known as And-Inverter graph (AIG). In fact, its core principle is based on an acyclic network of nodes and edges, where node is two-input AND gate and edge behaves like a negation of the logic signal passing through it between nodes.

A. Toolset for operation with AIGs

The term AIGER [4] denotes a format and, in the same time, also set of utilities for And-Inverter Graphs processing, which was developed at Johannes Kepler University in Linz. AIGER has been presented to the general audience at the Alpine Verification Meeting 2006 in Ascona. Main idea was to provide a simple, compact file format for a model checking purposes. In fact, the specification of AIGER format is available in two, slightly different version: an ASCII and a binary. Each version is conceived with the aim to accommodate somewhat different purpose.

IV. Proposal of novel format

With the aim to keep the complex nature of polymorphic circuits synthesis at a reasonable level, number of permissible modes for each node within AIG representation was strictly limited to the number of two. It means the final polymorphic circuit can ultimately work only in just two different operating modes, where the actual mode is switched by state of the environment. In general, a resulting behaviour of the circuit built in AIG can be affected only by two aspects - interconnection and edges (wire or inverter).

One of the possible ways how to enhance the capabilities of AIGs involves definition of new edge types. Thanks to the polymorphism it is possible to change behaviour of gates and also inverters. AIG contains only AND gates, however, any other function can be built from AND gates and their appropriate interconnection. This idea has resulted into the

no.	Description			Conventional solution			Polymorphic solution		Improvement	
#	Description	Circuit 1	Circuit 2	Circuit 1	Circuit 2	SUM	AIG	PAIG	Conv. vs PAIG.	AIG vs PAIG
				[ANDs]	[ANDs]	[ANDs]	[ANDs]	[ANDs]	[%]	[%]
1	2-bit ALU	Logic ALU	Arithmetic ALU	9	7	16	18	10	44.44	37.50
2	2-bit Adder	SUM	Carry	4	6	10	13	7	46.15	30.00
3	Cellular tr.function	Rule 30	Rule 100	4	4	8	13	6	53.85	25.00
4	GRAY/BCD Coder	Gray	BCD	16	7	23	26	16	38.46	30.43
5	Self-checking adder	Carry	Carry	4	4	8	8	4	50.00	50.00

TABLE I
RESULTS COMPARISON OF CONVENTIONAL AIG REPRESENTATIONS AND PAIG REPRESENTATIONS.

extended variety of edge types - from the initial two types to total of four types now:

1) Normal edge - wire.
2) Inverted edge - inverter.
3) Polymorphic edge 1:
   - In mode 1 - wire,
   - in mode 2 - inverter.
4) Polymorphic edge 2:
   - In mode 1 - inverter,
   - in mode 2 - wire.

The ordinary AIGER format works only with unsigned integers, where the even reference indexes are treated as ”wires” and odd are being seen as ”inverters”. At the beginning it is necessary to inform AIGER parser about the intention to use the extended format. Format identifier in header must contain ”paag” string. Then, the extension for polymorphic circuits will be correctly recognized. Extending AIGER to work with **signed integers** is necessary for the support of new edge types - polymorphic edges. In fact, ordinary edges are staying unchanged, while the polymorphic edges have negative prefix before their object index. Following example highlights the proposed extension:

- Polymorphic edge 1 (mode 1 = wire, mode 2 = inverter) will be noted as **negative even** integer.
- Polymorphic edge 2 (mode 1 = inverter, mode 2 = wire) will be noted as **negative odd** integer.

V. EXPERIMENTS AND DEMONSTRATION

For the purpose of demonstrating properties of the newly proposed format for polymorphic circuits representation five different experiments were prepared in total. These experiments clearly show the efficiency of polymorphic circuits handling using new PAIG (paag) format in comparison with conventional solution based on standard AIGER format without additional modifications.

The table I summarizes the results of all the conducted experiments. The table is separated into two main columns: Conventional solution and polymorphic solution. The conventional solution contains number of used AND gates with conventional technology and AIG representation. The polymorphic solution contains number of ANDs required by polymorphic technology. An AIG column contains number of ANDs required in case of using virtual polymorphism (basic AIG representation). A column PAIG denotes number of used ANDs in PAIG/PAAG required by the newly proposed format of circuit representation. Third column shows improvement in a percentage ratio between conventional solution vs polymorphic solution and also the comparison of virtualized polymorphism to PAIG/PAAG representation. The proposed PAIG/PAAG representation scheme can provide average saving of 34.59% AND gates.

CONCLUSIONS

Novel format for representation of polymorphic circuits using AIG was proposed in this contribution. It is an extension of AIGER format, which is fully supported in well known tools as ABC and similar ones. Several experiments show that the novel format turns out to be very effective approach to the representation of polymorphic circuits, including very complex variants.

Future research directions are aiming at the development of PAIG tools package for synthesis of more complex polymorphic circuits, including polymorphic variant of the original rewriting and structural hashing operation [5]. This tool will be based on PAIG/PAAG format while preserving the backward compatibility with AIG format. The preliminary tests indicate the perspective to obtain very good results in case of more complex circuits.

ACKNOWLEDGMENT

This work was generously supported by the grant FIT-S-17-3994 of Brno University of Technology, “Advanced parallel and embedded computer systems” (2017-2019). Another support was also provided by The Ministry of Education, Youth and Sports of the Czech Republic from the National Program of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602.

REFERENCES

[1] A. Stoica, R. Zebulum, and D. Keymeulen, “Polymorphic electronics. Proc. of Evolvable Systems: From Biology to Hardware Conference,” vol. 2210 of LNCS, pp. 291–302, 2001.

[2] A. Stoica and R. Zebulum, “Multifunctional logic gate controlled by temperature,” in *NASA Tech Briefs*. California Institute of Technology, 2005, p. 18, NPO-30795.

[3] R. Růžička, “Gracefully degrading circuit controllers based on polytronics,” in *Proc. of 13th Euromicro Conference on Digital System Design*. IEEE Computer Society, 2010, pp. 809–812.

[4] A. Biere, K. Heljanko, and S. Wieringa, “AIGER 1.9 and beyond,” FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, Tech. Rep., 2011.

[5] A. Mishchenko, S. Chatterjee, and R. Brayton, “Dag-aware aig rewriting: a fresh look at combinational logic synthesis,” in *2006 43rd ACM/IEEE Design Automation Conference*, July 2006, pp. 532–535.