

# ChipCflow - A Dynamic Dataflow Machine Compiler to convert C into a dataflow graph

Kelton<sup>b</sup>, Valentin<sup>a</sup> Jorge<sup>a</sup>, Joelmir<sup>a</sup>

<sup>a</sup> *University of Sao Paulo, Sao Carlos, Sao Paulo, Brasil*

<sup>b</sup> *Centro Paula Souza, Ourinhos, Sao Paulo, Brasil*

---

## Abstract

In order to convert High Level Language (HLL) into hardware, a Control Dataflow Graph (CDFG) is a fundamental element to be used. Related to this, Dataflow Architecture, can be obtained directly from the CDFG. The ChipCflow project is described as a system to convert HLL into a dynamic dataflow graph to be executed in dynamic reconfigurable hardware, exploring the dynamic reconfiguration. The ChipCflow consists of various parts: the compiler to convert the C program into a dataflow graph; the operators and its instances; the tagged-token; and the matching data. In this paper, a C compiler to convert C into a dataflow graph is described. Some results are presented in order to show a proof-of-concept for the project.

*Key words:* C Compiler; Dynamic Dataflow Architecture; Dynamic Reconfigurable Hardware; Tagged-token; Matching-Data.

---

## 1 Introduction

A Dataflow Architecture is an architecture where a natural parallelism is present. This kind of architecture was first researched in the 1970s and was discontinued in the 1990s (4; 6; 9). With the advance of technology of micro-electronics, the Field Programable Gate Array (FPGA) has been used, mainly because of its flexibility, the facilities to implement complex systems and intrinsic parallelism. Thus, dataflow architecture is a topic which has come to light again (5; 8), especially because of the reconfigurable architecture, which is totally based on FPGAs. On the other hand, much work is being done to covert high level language as a C language into hardware, in order to help engineers to project their systems using a high level of abstraction as well as a digital logic level. In particular, the ChipCflow project is a system where a C program is initially converted into a Dynamic Dataflow graph, followed by its execution in Reconfigurable Hardware. Its flow diagram is shown in Figure

1. As can be clearly seen in Figure 1, the ChipCflow system begins in a host machine where a C program is edited, to be converted into a control dataflow graph (CDFG) generating a CDFG object program. The CDFG object program is converted into a VHDL where modules of CDFG are accessed from a data base of VHDL modules. After generating the complete VHDL program, an EDA tool to convert the VHDL program into a bitstream and to download it to a FPGA is used.

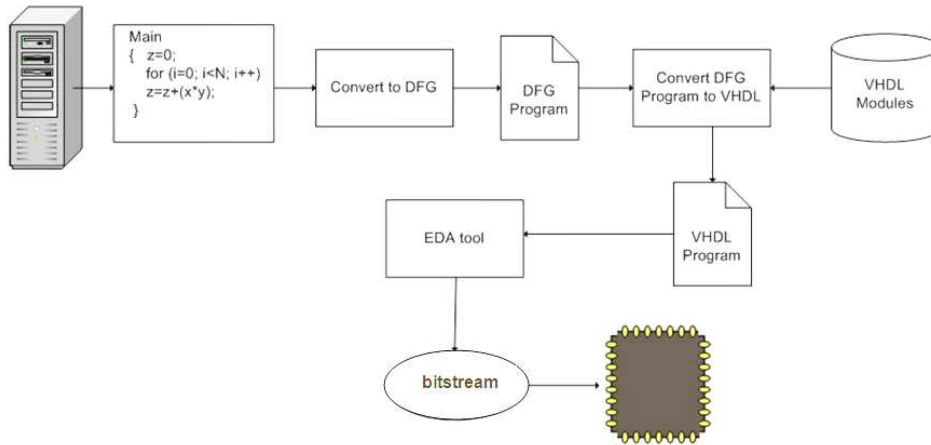


Fig. 1. The Flow Diagram for ChipCflow tool.

## 2 The Operators and C statements implemented in Dataflow Graphs

The operators to be used in the ChipCflow project are: "decider", "non deterministic merge", "deterministic merge", "branch", "copy" and "operator". They are described in Figure 2.

As can be clearly seen in Figure 2, the "decider" operator will be used to generate a control signal "TRUE" or "FALSE" after to execute a boolean operation as "<, >, =, !=, >=, <=" . The "non deterministic merge" will be used to forward a item of data coming into the operator. Otherwise, in "deterministic merge" the forward depends on the control signal. The "branch" operator will be used to forward a item of data through the "TRUE" or "FALSE" arc. The "copy" operator will duplicate a item of data. Finally an "operator" will be used to generate a result after to execute arithmetic operations as "+, -, \*, /, \*\*".

The dataflow graph of the *While* statement was implemented using these operators and is described in Figure 3. In Figure 3 there are two branch operator; two deterministic merge; five copy; one decider with boolean operator ">" and two operators with arithmetic operation "+".

In the next section the basic structure for the C compiler and some examples

of graphs are presented.

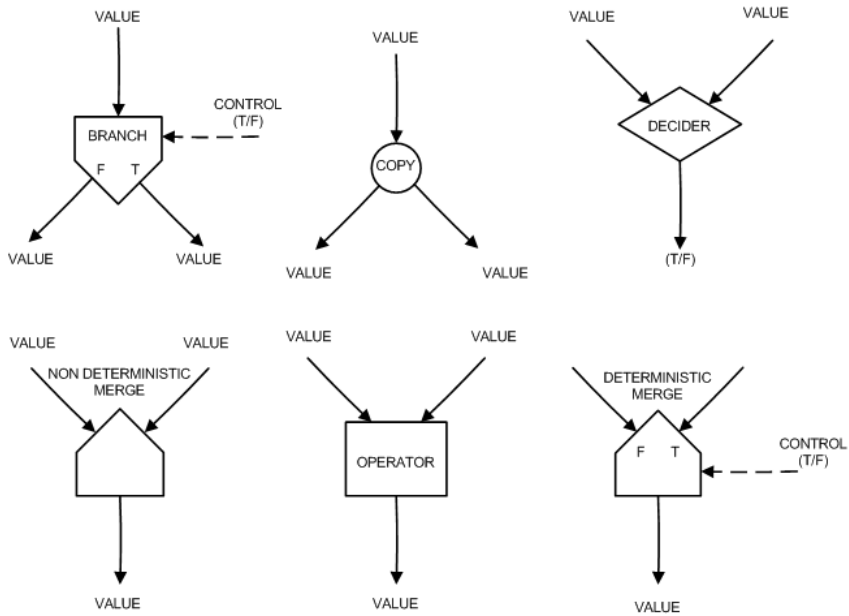


Fig. 2. Operators of the Dynamic Dataflow Model.

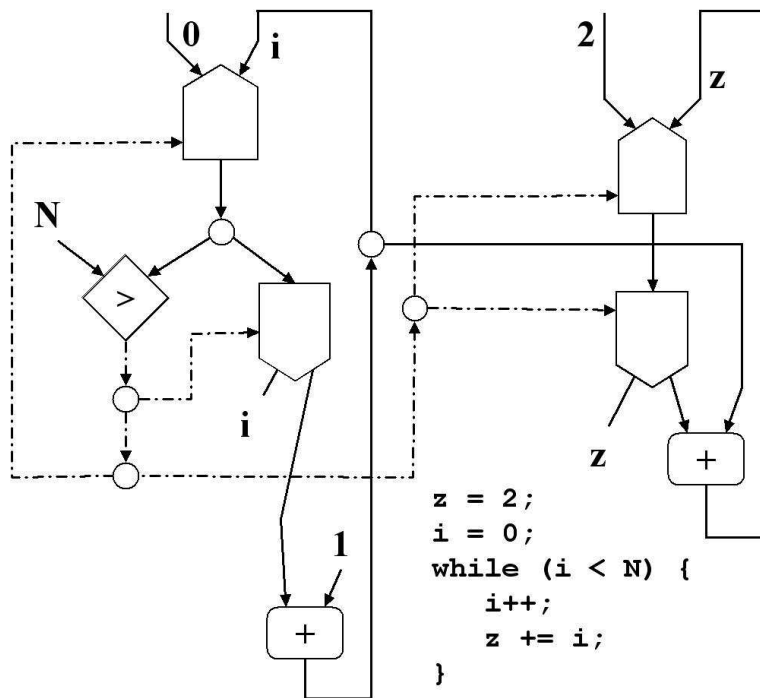


Fig. 3. Graph extracted from a while command.

### 3 The C Compiler to generate dataflow graph

The compiler structure was implemented in C++ and it is made up of two main parts: lexical analysis and VHDL code generator. The lexical analysis part is performed over the original C code and for each letter, number and reserved word, a code scanning generate a token. After the lexical analysis, and code scanning, a file with all the tokens are generated to be converted into a VHDL file.

#### 3.1 Generating a Binary Mapping File after Lexical Analysis

To generate a Binary Mapping file, a token was defined and its format is described in the Figure 4. As can be clearly seen in the Figure 4, the first 4-bits of the token has been used to identify the operator; the second, the thirty and the fourth 5-bits have been used to identify the three inputs (*a, b and c*) of the operator; finally the sixth and the seventh 5-bits to identify the outputs (*s and z*) of the operator. This is a generic template for the operator with three inputs and two outputs signal, however there are operators which less than three inputs signals and just one output signal.

Operator	Input a	Input b	Input c	Output s	Output z
----------	---------	---------	---------	----------	----------

Fig. 4. The format of token.

An example of the dataflow graph and its packets of bits for a *While* C command is describes in Figure 5. It is shown in the Figure 5 that each operator has a set of bits to identify its function, as well as each arc has a set of bits to identify its interconnections. In particular, in the left top of the figure has an operator with the code "0001" and its arcs "00000" (*value "0"*), "00001" (*value "i"*), "00010" (*a control signal*) and "00011" (*the output signal*), corresponding to three input signals and one output signal respectively.

The packet of bits for this particular operator can be clearly seen in the first packet of bits in the Figure 6, that is accorded on the format described in Figure 4. The "xxxxx" in the packet of bits represent an arc with no connection signal. Thus, a file with these packets of bits, is a binary representation for a dataflow graph extracted from a while C statement in the C pre-compiler.

The next step is to use the binary representation to identify the VHDL operators, what components will be used and what instances and their interconnections will be generated for the execution of the VHDL program in the ISE Xilinx platform. This step is described in the nest item.

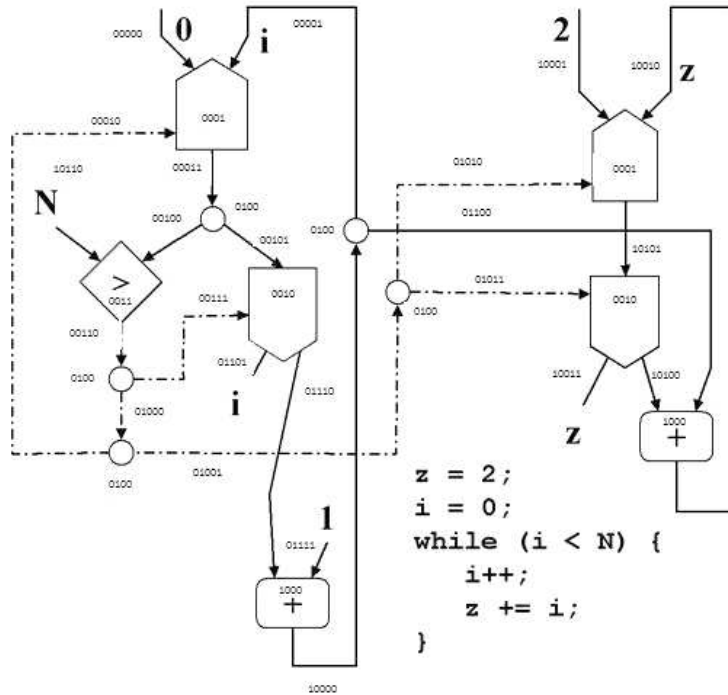


Fig. 5. A While C command and its tags.

```

000100000000010001000011XXXXX
000110001100100101010101XXXXX
00100010100111XXXXX0110101110
00101010101011XXXXX1001110100
00111011000100XXXXX00110XXXXX
10000111001111XXXXX10000XXXXX
10001010001100XXXXX10010XXXXX
010000011XXXXXXXXXX0010000101
010000110XXXXXXXXXX0100000111
010001000XXXXXXXXXX0001001001
01001000XXXXXXXXXX0000101100
010001001XXXXXXXXXX0101001011

```

Fig. 6. The Binary Mapping File generated for While C command.

### 3.2 Generating a VHDL file

The operators described for the ChipCflow project were implemented and tested and are described in (11). In order to generate a VHDL program, the file with the binary mapping is converted using all the components of the operators, that is already implemented, their instances and interconnections.



## 4 Conclusion

Research to convert High Level Language (HLL) into hardware has put forward various possibilities mainly with the flexibility and capacity of the reconfigurable architectures. A Control Dataflow Graph (CDFG) is a fundamental element in this process. Otherwise, a Dataflow Architecture, which was the focus in the 1980s, can be obtained directly from the CDFG. In particular, dynamic dataflow architecture can be generated in order to produce a high level of parallelism. In this paper, the ChipCflow project was described as a system to convert HLL into a dynamic dataflow graph to be executed in dynamic reconfigurable hardware, exploring the dynamic reconfiguration. The operator, which is the main element in the dataflow graph, was implemented, and spent *6ns* to execute all the process. The simulation results demonstrate the proof of concept for the operator. The next steps of the ChipCflow project are to implement the complete model of instances and generate an analysis with benchmarks to verify the impact of this approach.

## References

- [1] Ali, F. M. and Das, A. S.Azme, Hardware-software co-synthesis of hard real-time systems with reconfigurable FPGAs, *ELSEVIER - Computer and Electrical Engineering*(2004), volume=30,pg 471-489
- [2] Arnold, J, The SPLASH 2 Software Environment,*IEEE Workshop on FPGAs for Custom Computing Machines*,(1993),88-93
- [3] ARNOLD, J. and D. Buell and E. Davis, SPLASH 2, *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures* (1992), 316-324
- [4] Arvind, Dataflow: Passing the token, *ISCA Keynote* (2005)
- [5] Capelli, A, A Dataflow Control Unit for C-to-Configurable Pipelines Compilation Flow, *IEEE Symposium on Field-Programmable Custom Computing Machines FCCM'04* (2004)
- [6] Dennis, J. B., A preliminary architecture for a basic dataflow processor, *Proceedings of the 2nd Annual Symposium on Computer Architecture*, (1975)
- [7] Silva, J.L., Executing Algorithms for Dynamic Dataflow Reconfigurable Hardware - A Purpose for Matching Data *The 6th IEEE International Workshop System-on-Chip for Real-Time Applications - IWSOC'06*,(2006)
- [8] Swanson, S., Wavescalar, *36th Annual International Symposium on Microarchitecture* (2003)
- [9] Veen, A. H., Dataflow Machine Architecture, *ACM Computing Surveys*, n.4, (1986), v.18, pp 365-396
- [10] Astolfi, V. F. A., Silva, J. L., Execution of algorithms using a Dynamic Dataflow Model for Reconfigurable Hardware - Commands in Dataflow Graph., *The 3th IEEE Southern Conference on Programmable Logic - SPL2007*, Mar del Plata, Argentina. pp 225-230. (2007)

- [11] Silva, J. L., Correia, V. M., The Fibonacci Sequence implemented in the ChipCflow Machine., *The 12th IEEE International Conferences on Computational Science and Engineering (CSE-2009)*, Vancouver, Canada, (2009).
- [12] Davis D., Beeravolu S and Jaganathan, R, Hardware/Software Codesign for Platform FPGAs - Xilinx *Xilinx*, (2005)

**Silva** Jorge received Ph.D. in Electrical Computer Engineering, from the University of Campinas, Campinas, in 1992 M.S. degrees in Computer Science, in 1986, from Univeristy of Sao Paulo, Sao Carlos, and a B.S. degree in Computer Science from Federal University of Sao Carlos, Sao Carlos, in 1978. In 1979, he was a professor at the Federal University of Sao Carlos. From July 1999 he moved to University Euripides of Marilia. In July 2005 he moved to University of Sao Paulo, where he is presently a Professor. His research interests include Reconfigurable Computer, High performance Dataflow Architectures, and Embedded Systems design.