

A Logic Based Hardware Development Environment

S Belkacemi, K Benkrid and D Crookes

School of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN

Abstract

This paper presents a logic based approach to hardware abstraction and composition based on the logic programming language Prolog. This is an attempt to satisfy the dual requirement of abstract hardware design and hardware efficiency. Central to this approach is a hardware description environment called HIDE, which provides more abstract and elegant hardware descriptions and compositions than are possible in traditional hardware description languages such as VHDL or Verilog. The environment enables highly scaleable and parameterised composition of blocks using a small set of constructors e.g. 'horizontal' and 'vertical' for 2D circuit abstractions and the novel 'above' constructor for 3D circuit compositions. It also generates pre-placed configurations in EDIF (and VHDL) format for Xilinx FPGAs.

1. Introduction

FPGA programming model is often hardware-oriented rather than application-oriented. Although it is possible to use the well-established hardware description languages (e.g. VHDL), these are not very suitable for describing structural circuits.

This paper presents a logic-based approach to circuit description and composition. Central to this approach based on a Prolog [1] based high-level hardware description environment, called HIDE, which is being developed at Queen's University Belfast. Central to HIDE is a high-level hardware description notation that allows for structured, parameterised and recursive circuit descriptions in two and three dimensions. Pre-placed FPGA configurations in EDIF format (and VHDL) are generated automatically from such descriptions. This paper presents our logic based approach in the context of an implementation for Xilinx Virtex FPGAs family.

2. The HIDE notation

The main component of a HIDE configuration description is a block, which is an abstract representation of a circuit. Starting from primitive basic building blocks, compound blocks can be assembled. These can in turn be used as basic blocks for other compound blocks until the final block representing the desired circuit is obtained.

In a two-dimensional plane, a block is a rectangular, non-overlapping block. At the outer edges of this block, there are input and output ports and control signals (see Figure 1).

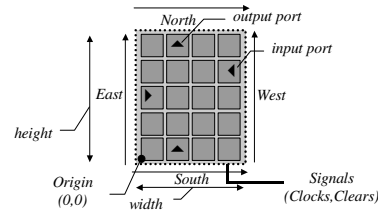


Figure 1. Block representation

Circuits can be combined either vertically or horizontally. Figure 2 illustrates the five basic constructors in HIDE (**vertical**, **horizontal**, **v_seq**, **h_seq** and **offset**) [2].

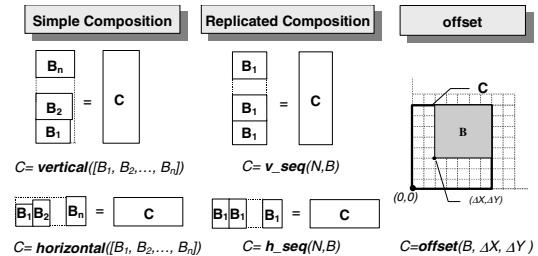


Figure 2. HIDE's basic constructors

During sub-block composition, the ports on the corresponding sides of adjacent sub-blocks are connected together. Alternatively, the user can connect the ports explicitly by giving a network connection, which specifies all logical port connections

2.1 3-D Circuit Abstraction

Many circuits in DSP are better viewed as 3-D structures rather than 2-D ones [3]. That is why we have extended the basic 2-D block abstraction in HIDE to handle an extra dimension as shown in Figure 3.

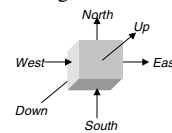


Figure 3. 3-D Block Abstraction in HIDE

In 3-D case, blocks can be combined either horizontally, vertically in a plane as in 2-D case or above each other in the third dimension using the new *above* and *a_seq* constructors.

Note that the 3-D block structures are not physical but rather conceptual at this stage. Since, the blocks are still 2-D structures on real hardware, the 3-D architectures are projected on the 2-D FPGA structure. Since there are many ways in which a 3-D structure is projected onto a 2-D one (e.g. vertically, horizontally, diagonally etc.), different projection modes can be specified by the user. These are a bit analogous to JAVA layout managers. Figure 4 illustrates a horizontal projection.

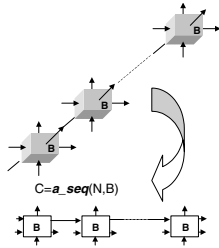


Figure 4. Horizontal projection of 3-D structures

3. A Matrix Multiplier Core Example

Figure 5 gives the 3-D systolic architecture for a 3x3 by a 3x3 matrix multiplication example.

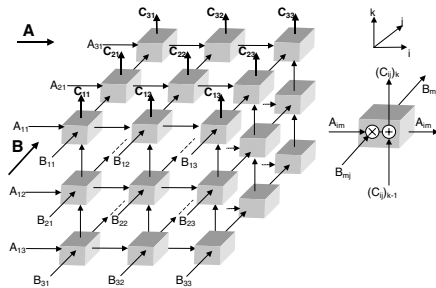


Figure 5. Systolic array architecture for a 3x3 by a 3x3 Matrix Multiplier ($C = A \times B$)

The basic building block in these architectures is a Multiply-Accumulate (MAC) block. An 8x8 multiplier can be assembled in HIDE based on 2-bit basic multipliers, two 10-bit adders (Adder10) and one 16-bit adder (Adder16) [4]. Parameterised adders can be described in HIDE using the **vertical** and **v_seq** block composition constructors. Using a 10-bit adder (Adder10) and a 16-bit one (Adder16) and using basic 2-bit multipliers (**mult2_lsb2**, **mult2_mid**, **mult2_msb**), an 8x8 multiplier can be described as follows (see Figure 6):

```
Multiplier8x8 = above([
  h_seq(4,
    vertical([mult2_lsb2, v_seq(3, mult2_mid),
              mult2_msb ]
    ),
  h_seq(2, Adder10),
  Adder16])
```

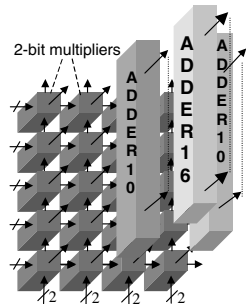


Figure 6. An 8x8 multiplier architecture in HIDE

A whole matrix multiplier can then be assembled in a similar way using a set of Prolog rules. The rules simply use the horizontal, vertical and above constructors. FPGA configurations in EDIF format with full placement

information are then generated automatically in less than 1 sec.

Figure 7 shows the physical layout and FPGA floorplan of a 3x3 matrix multiplier of 8-bit coefficients (using a horizontal projection of the 3-D structure shown in Figure 5), generated automatically using HIDE. The circuit consumes 2,448 slices on an XCV1000E-6 (out of 24,576 slices) and can run at 175 MHz, thus performing over 4.7 Billion MACs per second. This leads to 175 Million full 3x3 matrix result per second. The circuit is very compact and has a 70% hardware utilisation.

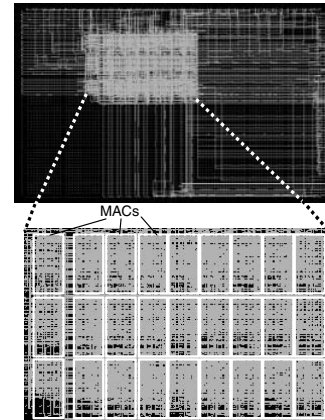


Figure 7. Physical layout of a 3x3 matrix multiplier

4. Conclusion

In this paper, we have presented a logic based approach to hardware abstraction and composition. Central to this approach is a Hardware Description Environment based on the logic programming language Prolog, called HIDE. The environment is based on a hardware description notation, which allows for very concise descriptions of scaleable and parameterisable architectures using a small set of 2-D and 3-D circuit constructors. FPGA configurations in EDIF/VHDL format with optional placement information can be generated automatically from HIDE descriptions. On top of the HIDE notation, Prolog based high-level generators that take application specific algorithm descriptions (e.g. for Matrix Multiplication, Convolutions etc.) and translate them into HIDE descriptions, can be built [2]. This allows for complete hardware generation from application abstractions.

5. References

- [1] Clocksin, W. F., and Melish, C. S., 'Programming in Prolog', Springer-Verlag, 1994.
- [2] Benkrid, K., 'Design and Implementation of a High Level FPGA Based Coprocessor for Image and Video Processing', PhD Thesis, Department of Computer Science, The Queen's University of Belfast, 2000.
- [3] Keshab, K., P., 'VLSI Digital Signal Processing Systems: Design and Implementation', John Wiley & Sons, 1999
- [4] Xilinx Application Notes, 'Design Tips for HDL Implementation of Arithmetic Functions', XAPP215, June 2002. <http://www.xilinx.com>