**DG2VHDL: A Tool to facilitate the Synthesis of Parallel Array Architectures**

Andrew Stone (NU), Prof. Elias S. Manolakos (NU)

**Goals**
1. **Simple Design Entry:** Bring hardware design for signal processing to a very high level of abstraction where algorithms are described graphically using Dependence Graphs (DGs).
2. **Synthesis Power:** Overcome non-scalability of existing synthesis tools by intelligently partitioning and mapping algorithms onto parallel processing arrays.
3. **Quality of Results:** Silicon area should scale at the slowest possible rate with increases in problem size.

**Significance**
1. Current hardware synthesis tools require expert users to write HDL in a quality, synthesizable manner. **DG2VHDL** will allow non-experts to reap the benefits of these powerful industrial tools by simplifying the entry point.
2. Poorly written, unpartitioned HDL which describes a complex algorithm will synthesize very slowly (if at all) and produce very poor area results. Certain partitioning schemes demonstrably work well. Rather than force a designer to decompose an algorithm into multiple, concurrent partitions, and write HDL for each, the design process can be greatly improved by automating this process.
3. The ultimate goal of synthesis tools is to meet the problem requirements while utilizing the minimal possible amount of hardware resources.

**Technical Approach**

### Design Flow

1. In Figure 1, section 1: A text parser has been created which reads a text based DG description and linear mapping operator information and creates an internal database representation.

2. **DG2VHDL** applies the space-time mapping operators (linear or non-linear) to the DG to create an internal representation of the Signal Flow Graph (SFG). The SFG consists of processing elements (PEs) and FIFOs (see Figure 2).

3. Output is generated. High quality VHDL models of the PEs are generated, using an optimal loop rolling approach (described in [1]). FIFO VHDL and top-level SFG wiring VHDL are created, along with synthesis scripts. Optionally, a test bench may be generated to facilitate array testing.

4. Industrial tools take over. Synthesis is performed targeting a plethora of ASIC and FPGA manufacturers.

---

**Figure 1:** Design Flow with DG2VHDL.
Internal Processing

1. With theoretical approach, DG is translated to SFG (another graph).
2. With DG2VHDL, DG is mapped to VHDL model of the SFG, with FIFOs for delays, and VHDL architectures for PEs.
3. This VHDL model may then be synthesized, with industrial tools, to the RTL level, which may then be further translated to gate level (and silicon).

Automated Testbench Generation

1. VHDL testbench timing is specific to particular mapping operators used.
2. Designer is freed from writing new VHDL testbench for each mapping attempted.
3. DG2VHDL will parse input file and generate mapping specific testbench, using user supplied test vectors.
4. Cycle by cycle IO behavior is reported, as well as normed error analysis, for evaluation of correctness and datapath rounding/truncation schemes.

Results:

6. Figure 4 shows the area results from the synthesis of two VHDL models for higher order moments estimation (see [1] and [2]), where M represents the length of the input sequence (i.e. the problem size).
7. The poorly scaling area (V2.02) represents the naïve approach of no loop partitioning, only entity partitioning.
8. Optimal loop rolling (an algorithm embedded into DG2VHDL) provides additional loop partitioning, and allows for excellent area scaling with problem size (V2.11 in Figure 4).
9. To date, we have synthesized architectures for many other significant algorithms: Motion Estimation [2], Discrete Wavelet Transform [3], DCT [5], and Lempel-Ziv Compression [6].

Figure 2: Model Translation, Theory and Practice.

Figure 3: Software flow with DG2VHDL testbench generator.

Figure 4: Demonstration of DG2VHDL scalability improvement from version 2.02 (1998) to version 2.11 (1999).
Relation to ERC

DG2VHDL is a vital component of research thrust R3. The large linear algebra and signal processing algorithms used in the solution of inverse problems are precisely what DG2VHDL is being developed for. For real time processing to be achieved, scalable parallel architectures are a necessity, and the DG2VHDL approach greatly facilitates their rapid prototyping.

Current Status

1. DG2VHDL is currently fully functional, allowing for DGs for a wide variety of Signal and Image Processing algorithms to be quickly entered and synthesized for mapping onto functioning FPGA implementations.

Plans and Project Evolution

1. Within the next year DG2VHDL will add additional features, e.g. it will allow for the automated selection of linear mapping operators based on desired design characteristics (Speed, Area, Etc). This will allow for a more rapid exploration of the design space by engineers.

2. Within three years, DG2VHDL will be a stable environment for the rapid prototyping and simulation for scaleable, parallel architectures for signal processing and other algorithms.

References


PI Contact Information

Prof. Elias S. Manolakos
ECE Dept., 442 Dana Build.
Northeastern University
Boston, MA 02115
elias@ece.neu.edu