Extended Abstract: A Race-free Hardware Modeling Language

Patrick Schaumont (UCLA) schaum@ee.ucla.edu
Sandeep Shukla (Virginia Tech) shukla@vt.edu
Ingrid Verbauwhede (UCLA/KUL) ingrid@ee.ucla.edu

Abstract

We describe race-free properties of a hardware description language called GEZEL. The language describes networks of cycle-true finite-state-machines with datapaths (FSMDs). We derive a set of four rules under which a network of such FSMDs satisfies the Kahn principle. When applying those rules, GEZEL programs will be determinate and a designer will thus obtain race-free hardware. We define extended FSMD networks as FSMD networks for which some components are user-defined and not specified as FSMDs. An important result is that the determinate properties of the FSMD network are also valid for the extended FSMD network provided that the user-defined components are determinate. Most hardware description languages do not have this determinacy. Their simulation semantics are dependent on simulator implementation, and on a run-time race resolution mechanism. We therefore position GEZEL as a model of computation that RTL designers should have in mind while creating RTL models. In fact, we can generate SystemC and other HDL code from GEZEL models, thereby guaranteeing the determinacy in the generated HDL code.

1 Motivation

Simply stated, the Kahn principle tells that under certain process semantics a determinate set of components can be composed into a determinate system [1]. There are several process semantics for which this principle is valid, as shown in Figure 1. They include Kahn Process Networks, Input-Output Automata, FSMD, and others. Kahn Process Networks are well-known forms of dataflow-like networks. Input-Output Automata are a semantic model for concurrent, distributed discrete-event systems [2]. In this paper, we present an FSMD modeling language called GEZEL that satisfies the same Kahn principle.

We use GEZEL for hardware design. By creating components with determinate behavior, we are able to assemble them into a determinate system. The most obvious causes of non-determinism in hardware design are races. In traditional HDLs (VHDL, Verilog or SystemC), race conditions occur when a single global variable is concurrently assigned by different processes. The value stored in the global variable is indeterminate. This will show up as 'X' in a good simulator, but often the result is simply simulator dependent. Non-determinism by itself can be useful as a specification mechanism at higher abstraction level [3]. But in HDL-based design, non-determinism sneaks in as a side effect: a designer can create race conditions without being aware of it [4]. This poses a challenge to the implementation, verification and comprehension of the RTL code. Therefore, we feel that irrespective of the language of choice for RTL coding (SystemC, Verilog or VHDL), the desired model of computation for such coding should be networks of processes which satisfy the Kahn principle.

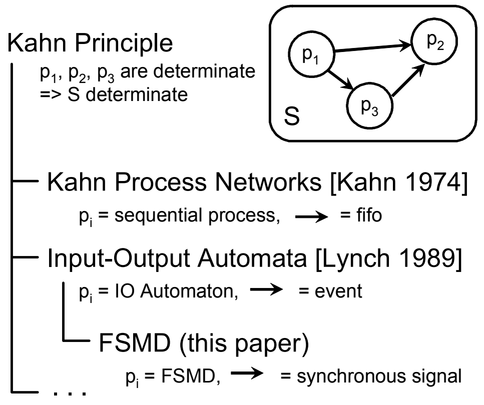
Kahn Principle
p_1, p_2, p_3 are determinate => S determinate
p_1
p_2
p_3
S
Kahn Process Networks [Kahn 1974]
p_i = sequential process, → = fifo
Input-Output Automata [Lynch 1989]
p_i = IO Automaton, → = event
FSMD (this paper)
p_i = FSMD, → = synchronous signal
. . .

Figure 1: The Kahn principle is applicable to multiple process semantics, including FSMD.

2 Hardware Modeling in GEZEL

The GEZEL language implements instances of an FSMD network. A small example of the GEZEL language, using two FSMD, is given in Listing 1.

One FSMD is called `counter`, and is an accumulating counter that can either increment, decrement or remain constant. The other FSMD is called `updown`, and controls the counter by observing the counter value and commanding it to increment or decrement. Each FSMD module consists of a datapath and a controller. A datapath is defined by means of a number of datapath instructions, which execute when selected by the controller. The complete model is synchronous and works under control of a single, unspecified clock. When the example in Listing 1 executes, the behavior for the first 7 clock cycles is as shown in Table 1. A comparison of Listing 1 with SystemC, as well as a detailed description of the GEZEL language (GEZEL Language Reference Manual) are available online [5].

LISTING 1. Two communicating FSMD in GEZEL

```
 dp counter(in ud : ns(2); out a : ns(3)) {
    reg c  : ns(7);
    sig nc : ns(7);
    reg u  : ns(2);
    sfg io { u = ud; a  = nc; }
    sfg up { c = nc; nc = c + 1; }
    sfg dn { c = nc; nc = c - 1; }
}
fsm fsm_updown(counter) {
    initial s0;
    state   s1;
    @s0 if (u[0]) then (dn,io) -> s1;
        else           (up,io) -> s0;
    @s1 if (u[1]) then (up,io) -> s0;
        else           (dn,io) -> s1;
}
dp updown(out ud : ns(2); in a : ns(3)) {
  sfg run { ud = (a==3) ? 1 : ((a==0) ? 2:0);}
}
fsm fsm_updown(updown) {
    initial s0;
    @s0 (run) -> s0;
}
system S {
    counter(u, a);
    updown (u, a);
}
```

3 Rules for Determinate Behavior

An FSMD in GEZEL can be formulated as an Input-Output Automaton. One particular property of IOAs that is of interest for our FSMD is the proof of Stark on determinate IOAs [2]. Stark's proof involves showing that the Kahn principle for IOA implies single-valued port histories. That is, for each possible history of input port values, only a single history of output port values is possible. For a GEZEL FSMD, single-valued port histories can be obtained by veryfying four properties in the GEZEL description. The four properties can be enumerated as follows.

1. **Single Assignment**: Each signal or register in a GEZEL description may be assigned only once per clock cycle. For example, instructions up and dn in Listing 1 are not allowed to execute simultaneously.
2. **No Undefined Operands**: All expressions that execute during a clock cycle must be determined. 'Dangling' signals and inputs are not allowed.
3. **No Combinatorial Loops**: Circular dependencies between signals are illegal. Such dependencies can introduce artifical state, thus resulting in multiple possible port histories.
4. **No Undefined Outputs**: All outputs of an FSMD must be defined at each clock cycle.

Given a description made by a designer in GEZEL, we can evaluate the four properties above and conclude wether the FSMD will be determinate. This test is implemented in the GEZEL tools to guide a designer in writing race-free code.

Table 1: Behavior of counter in Listing 1.

clock edge	FSM	sfg executing	register	
			c	u
1	s0->s0	`up,io`	0->1	0
2	s0->s0	`up,io`	1->2	0
3	s0->s0	`up,io`	2->3	0->1
4	s0->s1	`dn,io`	3->2	1->0
5	s1->s1	`dn,io`	2->1	0
6	s1->s1	`dn,io`	1->0	0->2
7	s1->s0	`up,io`	0->1	2->0
8	(again like edge 2)			

4 Related Work

Bluespec organizes state variables inside of modules and then employs a rule-based coding style for the behavior of the module. Concurrent execution of rules is allowed provided that the result is equivalent to sequential execution of single rules [6]. An alternate approach to create race-free hardware is to start from a higher-level formalism that guarantees determinism, and create a compiler to synthesize hardware. This has been shown for example for Esterel [7].

5 Conclusions

We have presented a race-free hardware modeling language. This language creates networks of FSMDs. Building on the Kahn Principle and the semantics of Input-Output Automata, we have defined four rules that will help the designer in the modeling of determinate hardware using FSMD. GEZEL is developed with an open-source policy, and the webpage in [5] may be consulted to download a simulator for GEZEL designs, several cosimulators for GEZEL-based hardware-software codesign, and a GEZEL-to-VHDL code generator.

6 Acknowledgements

The authors acknowledge the support of NSF Grant CCR 0310527 (PS,IV), NSF Grant CCR 0237947 (SS), and SRC Grant 2003-HJ-1116 (PS,IV).

7 References

[1] G. Kahn, "The semantics of a simple language for parallel programming," Information Processing 74, 1974.

[2] N. Lynch, E. Stark, "A proof of the Kahn principle for input/output automata," Information and Computation archive, 82(1):81—92, 1989.

[3] S. Edwards, L. Lavagno, E. Lee, A. Sangiovanni-Vincentelli, "Design of Embedded Systems: Formal Models, Validation, and Synthesis," Proceedings of the IEEE, 85(3):366—390, 1997.

[4] OSCI, "Non-determinism in SystemC", Ch. 5.6 in "Functional Specification for SystemC 2.0", v. 2.0-Q, http://www.systemc.org

[5] GEZEL Homepage, http://www.ee.ucla.edu/~schaum/gezel.

[6] Arvind, R. Nikhil, D. Rosenband, N. Dave, "High-level Synthesis: An Essential Ingredient for Designing Complex ASICs," Proc. ICCAD, November 2004.

[7] S. Edwards, "High-level Synthesis from the Synchronous Language Esterel," Proc. of the International Workshop of Logic and Synthesis (IWLS), New Orleans, Louisiana, June, 2002.