Considerations on the effectiveness of an RTL scan design

Seiji Hirota, Computer Design Laboratory, Hiroshima City University

April 22, 2009

【Overview】

This work considers the effectiveness of the RTL scan design [1], and implements a system to achieve it. The RTL scan design [1] inserts scan paths into a functional RTL specification of a design that can exploit existing functional paths between sequential elements in the original circuit. Building scan paths at the functional RTL is expected to reduce the total area overhead. In addition, it can reduce the time to market. We show the insertion method of the scan paths in the following figures.

```
entity sample is
port(reset, clock:in bit;
       input: in integer range 7 downto 0;
       output: out integer range 7 downto 0);
end sample;
architecture behav of sample is
  signal Num1: integer range 7 downto 0;
  signal Num2: integer range 7 downto 0;
begin
  process(clock, reset)
  begin
    if reset='1' then
      Num1<=0; Num2<=0; output<=0;
    elsif clock'event and clock='1' then
      Num1<=input; Num2<=Num1;
      output<=Num1*Num2;
    end if;
  end process;
end behav;
```

scan →

```
entity sample is
port(scan_in: in bit;
       reset, clock:in bit;
       input: in integer range 7 downto 0;
       output: out integer range 7 downto 0);
end sample;
architecture behav of sample is
  signal Num1: integer range 7 downto 0;
  signal Num2: integer range 7 downto 0;
begin
  process(clock, reset)
  begin
    if reset='1' then
      Num1<=0; Num2<=0; output<=0;
    elsif clock'event and clock='1' then
      if scan_en='1' then
        output<=Num2; Num2<=Num1; Num1<=input;
      else
        Num1<=input; Num2<=Num1; output<=Num1*Num2;
      end if;
    end if;
  end process;
end behav;
```

Fig. 1　Insert the scan path to the RTL description

In addition, the order to connect the registers is decided based on a data transfer graph. In this graph, nodes represent inputs and registers, edges represent data paths. With each edge, we associate a weight that defines the cost. The edge-cost is computed according to the following rules.

1. There is a direct connection between the two memories. Cost = 0.
2. There is a conditional transfer path between the two memories. Cost = 1
3. There is a functional block between the two memories. Cost = 3n. (n: bit widths)

We can obtain a set of scan paths by repeating greedy search from primary inputs to primary outputs, i.e., traversing nodes so that the sum of weights of edges on the paths becomes small.

For example, suppose a scan path insertion to the RTL description (VHDL) in Fig. 1. Fig. 2 shows the data flow graph of the RTL description. As shown in Fig. 2, we obtain the scan path passing through input, Num1, Num2 and output with the minimum cost 9. The scan insertion in the RTL description corresponds to the highlighted lines in Fig. 1.

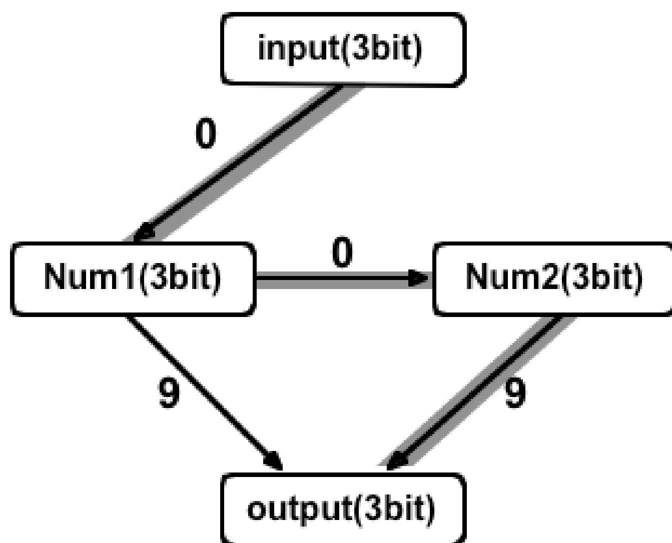

Fig. 2 Data flow graph of the RTL description in Fig. 1

[Experimental results]

Here, we call the traditional scan design at the gate level ”GL scan”. Note that we consider full scan design.

We made experiments with the benchmark circuits in ITC’99. The bit width of all the registers is assumed to be identical on the data path of every RTL circuit. We minimized either area or delay when we performed the logic synthesis, and compared the results by the RTL scan design with those by the GL scan design. For each benchmark, area, delay, the number of faults, the number of test patterns, fault coverage, fault efficiency and test application time are presented. We used Synopsys Design Vision for logic synthesis with library “class”, and Synopsys TetraMAX for test pattern generation. Experimental results are shown in the tables below.

Experimental results show that the RTL scan design resulted in a smaller area than the GL scan design in the case of area minimization, and resulted in a shorter delay time than the GL scan design in the case of delay time minimization. The area obtained by the RTL scan is smaller than that by the GL scan, while the delay by the RTL scan is also smaller than that by the GL scan, for all the circuits. This is because the scan insertion is applied to the RTL design prior to logic synthesis, and accordingly its overhead is absorbed by the logic synthesis. Another reason of area and delay reduction is the utilization of original data path for scan paths. That is, the RTL scan shares original data paths with scan ones as many as possible.

The test application time for RTL scan circuits is much smaller than that for GL scan circuits. This is chiefly due to parallel scan. The scan operations (shift in and shift out) are applied along the data paths in parallel for the RTL scan circuits, while those are applied along to a single scan path that is distinct from the original circuits. In general, the speeds of scan operation for the GL scan circuits are much slower than that of the normal operation, and thus the impact of the reduction is practically significant.

【Conclusion】

We considered the effectiveness of the RTL scan design proposed in [1]. Experimental results show that the RTL scan design can solve many issues of the traditional gate-level scan design. We shall consider other RTL DFT methods, and investigate more effective DFT.

【References】

[1] Y. Huang, C. C. Tsai, N. Mukherjee, O. Samman, D. Devries, W. T. Cheng, and S. M. Reddy, “On RTL scan Design,” Proc. ITC, pp.728-737, 2001.

[2] R. B. Norwood, and E. J. McCluskey, "ORTHGONAL SCAN: LOW OVERHEAD SCAN FOR DATA PATHS," Proc. ITC, pp.659-668, 1996.

[3] S. Bhattacharya, and S. Dey, "H-SCAN: A High Level Alternative to Full-Scan Testing With Reduced Area and Test Application Overheads," Proc. 14th VTC, pp.74-80, 1996.

[4] T. Asaka, S. Bhattacharya, S. Dey, and M. Yoshida, "H-SCAN+: A Practical Low-Overhead RTL Design-for-Testability Technique for Industrial Designs," Proc. ITC, pp.265-274, 1997.

[5] C. Aktouf, H. Fleury, and C. Robach, "Inserting Scan at the Behavioral Level," Proc. IEEE Design & Test of Computers, pp.34-42, July 2000.

B02

Bit width：3	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	106 (24.71)	103 (21.18)	115 (26.37)	128 (40.66)
Delay[ns] (overhead[%])	5.64 (4.83)	6.23 (15.30)	3.81 (10.76)	3.65 (6.10)
Number of faults	304	268	374	372
Number of test patterns	20	24	17	24
Fault coverage[%]	100	100	99.20	99.46
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	146	74	125	74

B03

Bit width：4	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	1021 (30.73)	815 (4.35)	1110 (27.00)	903 (3.32)
Delay[ns] (overhead[%])	16.09 (8.42)	13.08 (-11.86)	7.74 (8.71)	6.54 (-8.15)
Number of faults	2904	1624	3612	2156
Number of test patterns	106	87	103	96
Fault coverage[%]	99.66	100.00	99.66	99.72
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	6526	1407	6343	1551

B04

Bit width：8	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	1598 (21.98)	1359 (3.74)	1780 (19.22)	1565 (4.82)
Delay[ns] (overhead[%])	30.20 (1.27)	29.94 (0.40)	8.04 (6.63)	7.25 (-3.85)
Number of faults	4598	3284	5892	4696
Number of Test patterns	138	122	140	136
Fault coverage[%]	91.19	88.00	91.02	91.01
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	10146	1229	10292	1369

B06

Bit width：3	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	193 (29.53)	182 (22.15)	215 (25.73)	210 (22.81)
Delay[ns] (overhead[%])	5.73 (9.56)	8.30 (58.70)	4.22 (9.61)	4.53 (17.66)
Number of faults	568	446	704	570
Number of test patterns	25	24	24	27
Fault coverage[%]	94.72	100.00	95.45	100.00
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	311	74	299	83

B07

Bit width：8	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	1196 (23.05)	1055 (8.54)	1322 (11.84)	1271 (7.53)
Delay[ns] (overhead[%])	22.58 (4.88)	22.57 (4.83)	7.83 (24.29)	6.58 (4.44)
Number of faults	3640	2774	4482	4058
Number of test patterns	132	128	149	165
Fault coverage[%]	99.78	100.00	99.89	99.66
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	7580	1031	8549	1327

B09

Bit width：9	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	762 (30.93)	643 (10.48)	836 (25.15)	744 (11.38)
Delay[ns] (overhead[%])	13.63 (2.79)	12.62 (-4.83)	6.12 (8.15)	6.15 (9.04)
Number of faults	2190	1394	2622	1936
Number of test patterns	88	82	79	72
Fault coverage[%]	99.27	100.00	99.31	99.54
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	4093	497	3679	437

B10

Bit width：4	Area minimization		Delay minimization	
	GL scan	RTL scan	GL scan	RTL scan
Area (overhead[%])	362 (21.48)	326 (9.40)	428 (17.58)	457 (25.55)
Delay[ns] (overhead[%])	9.93 (3.87)	9.56 (0.00)	6.42 (6.12)	5.74 (-5.12)
Number of faults	1166	912	1506	1554
Number of test patterns	57	68	62	61
Fault coverage[%]	99.91	100.00	98.74	98.65
Fault efficiency[%]	100.00	100.00	100.00	100.00
Test application time (cycles)	985	275	1070	247