Tutorial of baseline algorithm

The baseline algorithm utilizes pattern simulation and observes the simulation signatures to generate patch function. We first use the example in Section V to demonstrate the algorithm.

a. First, introduce input patterns into both *F.v* and *G.v*, run simulation, and build up simulation tables of *F.v* with t_0 = 0, *F.v* with t_0 = 1, and *G.v*, respectively. Here we exhaustively simulate all pattern combinations and record simulation values of all nodes.

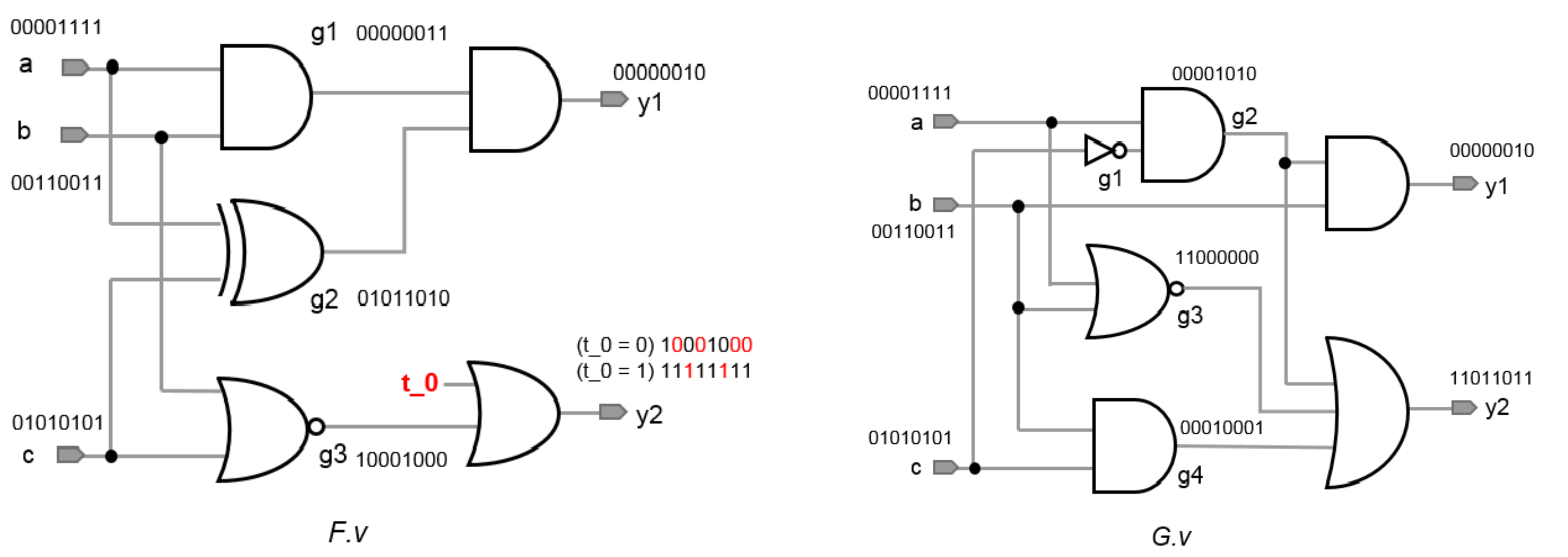

Fig. 3. Exhaustive simulation results of F.v and G.v.

Simulation table of *F.v* (t_0 = 0)

a	b	c	g1	g2	g3	t_0	$y1_F$	$y2_F$	
0	0	0	0	0	1	0	0	1	
0	0	1	0	1	0	0	0	0	NEQ
0	1	0	0	0	0	0	0	0	
0	1	1	0	1	0	0	0	0	NEQ
1	0	0	0	1	1	0	0	1	
1	0	1	0	0	0	0	0	0	
1	1	0	1	1	0	0	1	0	NEQ
1	1	1	1	0	0	0	0	0	NEQ

Simulation table of *F.v* (t_0 = 1)

a	b	c	g1	g2	g3	t_0	$y1_F$	$y2_F$	
0	0	0	0	0	1	1	0	1	
0	0	1	0	1	0	1	0	1	
0	1	0	0	0	0	1	0	1	NEQ
0	1	1	0	1	0	1	0	1	
1	0	0	0	1	1	1	0	1	
1	0	1	0	0	0	1	0	1	NEQ
1	1	0	1	1	0	1	1	1	
1	1	1	1	0	0	1	0	1	

Simulation table of *G.v*

a	b	c	$y1_G$	$y2_G$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	1

Fig. 4. Simulation tables for F.v with t_0 = 0, F.v with t_0 = 1, and G.v, respectively.

b. Observe the simulation values at primary outputs. As shown in Fig. 4, we can find that y1 between *F.v* and *G.v* are equivalent but y2 are not equivalent under some patterns.

c. Observe the values at y2 and t_0 in tables. As shown in Fig. 5, when observing the table of *F.v* with t_0 = 0, we can find that the values of t_0 should be changed to 1 under th patterns S_{on} = {(*a*, *b*, *c*) = (0, 0, 1), (0, 1, 1), (1, 1, 0), (1, 1, 1)} to fix the output value of y2 for equivalence (you can use simulation again to verify this). On the other hand when observing the table with t_0 = 1, we can find that the values of t_0 should be changed to 0 under the patterns S_{off} = {(0, 1, 0), (1, 0, 1)} for equivalence. Note that we must check that S_{on} and S_{off} should not have intersection, i.e., conflicts. If the intersection is not empty, it means that you cannot generate patch at the target point with these patterns. Finally, t_0 is don't care under patterns S_{dc} = {(0, 0, 0), (1, 0, 0)} since y2 of *F.v* is equivalent to y2 of *G.v* no matter what value t_0 is under these patterns.

Simulation table of *F.v* (t_0 = 0)

a	b	c	g1	g2	g3	t_0	$y2_F$
0	0	0	0	0	1	0	1
0	0	1	0	1	0	0→1	0→1
0	1	0	0	0	0	0	0
0	1	1	0	1	0	0→1	0→1
1	0	0	0	1	1	0	1
1	0	1	0	0	0	0	0
1	1	0	1	1	0	0→1	0→1
1	1	1	1	0	0	0→1	0→1

Simulation table of *F.v* (t_0 = 1)

a	b	c	g1	g2	g3	t_0	$y2_F$
0	0	0	0	0	1	1	1
0	0	1	0	1	0	1	1
0	1	0	0	0	0	1→0	1→0
0	1	1	0	1	0	1	1
1	0	0	0	1	1	1	1
1	0	1	0	0	0	1→0	1→0
1	1	0	1	1	0	1	1
1	1	1	1	0	0	1	1

Fig. 5. Figure out Son, Soff, and Sdc for t_0.

d. Integrate the information acquired from step (c) and build up a new simulation table Table I for new behavior of t_0, as shown in Fig. 6. Now we can synthesize a patch circu according to Table I.

Table I

	a	b	c	g1	g2	g3	t_0	
S_{dc}	0	0	0	0	0	1	X	
	0	0	1	0	1	0	1	S_{on}
S_{off}	0	1	0	0	0	0	0	
	0	1	1	0	1	0	1	S_{on}
S_{dc}	1	0	0	0	1	1	X	
S_{off}	1	0	1	0	0	0	0	
	1	1	0	1	1	0	1	S_{on}
	1	1	1	1	0	0	1	

Fig. 6. Table I.

e. Here we choose g1 and g2 as the base nodes, so the table can be transformed to Table II, as shown in Fig. 7. Note that we must check whether there are conflicts on the values of t_0. If conflict exists, you should try other base node combination.

Table I

a	b	c	g1	g2	g3	t_0
0	0	0	0	0	1	X
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	1	0	1
1	0	0	0	1	1	X
1	0	1	0	0	0	0
1	1	0	1	1	0	1
1	1	1	1	0	0	1

Table II (a)

g1	g2	t_0
0	0	X
0	1	1
0	0	0
0	1	1
0	1	X
0	0	0
1	1	1
1	0	1

Table II (b)

g1	g2	t_0
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 7. Table II.

f. Finally, we synthesize Table II and obtain the patch t_0 = g1 OR g2, as shown in Fig. 8.

Table II (a)

g1	g2	t_0
0	0	X
0	1	1
0	0	0
0	1	1
0	1	X
0	0	0
1	1	1
1	0	1

Table II (b)

g1	g2	t_0
0	0	0
0	1	1
1	0	1
1	1	1

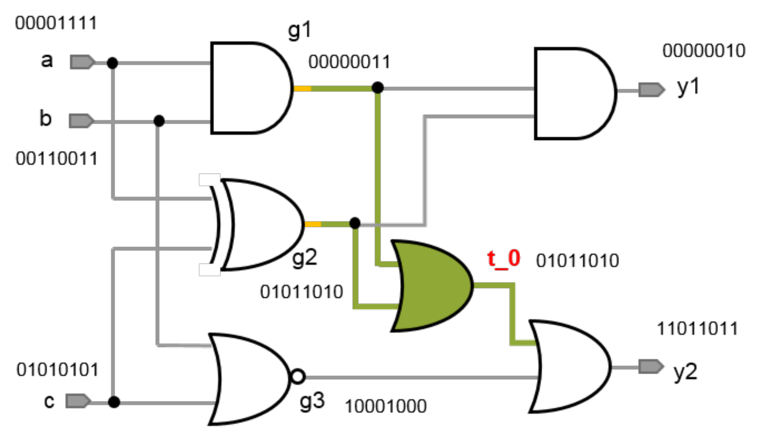

Fig. 8. Synthesize the patch t_0 = g1 OR g2 according to Table II.

g. Considering practicability, we can use random patterns instead of exhaustive patterns to generate patch. After simulation, in the synthesis phase, we only consider existing pattern combination, and view the others as don't cares. For the example shown in Fig. 9, we can still obtain the same patch as that generated at step (f), i.e., t_0 = g1 OR g2. Since the random patterns may not be complete enough for a right patch, you must check the equivalence after patching. If not equivalent, consider more patterns.

a	b	c	g1	g2	g3	t_0
0	0	0	0	0	1	X
0	0	1	0	1	0	1
0	1	0	0	0	0	0
1	1	1	1	0	0	1

g1	g2	t_0
0	0	0
0	1	1
1	0	1
1	1	X

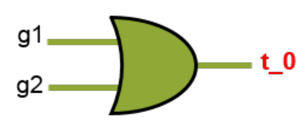

Fig. 9. The example with random simulation.

Algorithm

Algorithm: *Patch_Generation_Single_Fix(F, G)*

```
    //Assume there is only one target point t_0
Simulate input patterns for circuits {F(t_0 = 0), F(t_0 = 1), G};// exhaustive or random; step (a)
Build up the simulation tables {T(F, t_0 = 0), T(F, t_0 = 1), TG} for {F(t_0 = 0), F(t_0 = 1), G}, respectively;// step (a)
Observe output values in {T(F, t_0 = 0), T(F, t_0 = 1), TG} and mark the NEQ bits in {T(F, t_0 = 0), T(F, t_0 = 1)}; // step (b)
Generate {Son, Soff, Sdc} for t_0;// step (c)
    // by flipping t_0 values at NEQ bits in {T(F, t_0 = 0), T(F, t_0 = 1)} and validating through re-simulation
    // Son: the pattern set where t_0 should be 1 for EQ
    // Soff: the pattern set where t_0 should be 0 for EQ
    // Sdc: the pattern set where t_0 can be either 1 or 0
if Son ∩ Soff  is ∅ do
  Build up simulation table Tnew_t_0 for new behavior of t_0 based on {Son, Soff, Sdc};// step (d)
  repeat
    Choose base nodes B;// your method
    Transform table Tnew_t_0 to Tbase_node based on B;// step (e)
    if Tbase_node has conflict do
      continue;
    end if
    Synthesize patch R based on Tbase_node;// your method; step (f)
    break;
  end repeat
  return R;
end if
return Fail.
```