Problem A: Functional ECO with Behavioral Change Guidance

Yen-Chun Fang, Shao-Lun Huang, Ching-Yi Huang, Chi-An (Rocky) Wu, Chung-Han Chou Chih-Jen (Jacky) Hsu, WoeiTzy (Wells) Jong, and Kei-Yong Khoo
Cadence Design Systems, Inc.

Q&A

Q1. Regarding the question about support of patch's output ports, could I have one wire of new function as a support of another new function? Or, we could only use original wires in G1.v as support?

Assume that I want to change wire x and y in G1.v. If I use changed x as
one support of y, is it a valid patch?

```
module top_eco(x,a,y);
output x, y;
input a;
wire a, x, y;
not eco_n1(x, a);
and eco_a1(y, x, a);
endmodule
```

A1. Yes, you can use any signal in the patch module to compose your functions. We only allow all-fanout changes in the patch.

For example,

```
  not eco_n1(x, a);
  and eco_a1(y, x, a);
```

Signal *a* is fanout to eco_n1 and eco_a1.
If you want to change the signal *a* by your patch
…

```
output a;
input b;
assign a=b;
```

…
All fanout of a will be changed to *b* in this case.
You may not change designated fanout of *a*.

Q2. Regarding the verifier to the problem, please help to advise if the verifier will be provided or need to be written by ourselves.

A2. The patch verifier need to be written by the participants themselves.

Q3. Could you please help to advise if possible to use python programming language for problem a?

A3. We didn't restrict any language/package. The program can only run with single threads and can be executed on evaluation machines.

Q4. According to the input format and output format given in the problem, the netlist should be composed of primitive gates or constant value. However, in the test case2, I noticed that "assign" statements also appear in the test case. Are these kind of statements correct in the input and output files?

A4. Yes, the assign statement is feasible in the netlist.

Q5. According to the input format and output format given in the problem, all inputs, outputs seem to be declared in wires, too. But there are not declarations repeated in test case1, but we can find it in test case2. Are both of them correct?

A5. Both are correct. Please follow the Verilog standard.

Q6. Does the space appear between the gate name and the following parameters? We can find it in test case2, but not in test case1.

A6. Both with and without the spaces are correct format. Please follow the Verilog standard.

Q7. Please help advise If there is "assign a=b" in g1.v, is the wire "a" allowed to be patched, or rewired to another net.

A7. Yes, wire a is allowed to be patched. Please remember that once you patched wire a to a new function, all the fanout of wire a will be patched to the new function.

Q8. In the released two test cases, we found that the wires are often forgotten to be declared in Verilog. Should we detect which wires are forgotten to be declared in our program?

Or this is a mistake in test cases and will not happen in the alpha test, the beta test, the final test and the final hidden cases.

In r1.v of the test case 1, the wire n1 is not declared.

```
module top(o, a, b, c);
output o;
input a, b, c;
and g2(o, a, n1);
and g1(n1, b, c);
endmodule
~
```

A8. r1.v of testcase1 is in a correct format. Declaring wires is not necessary. Please follow the Verilog standard for more detail.

Q9. Regarding the testcase, the description has stated that ”The original specification (R1) and the new specification (R2) differ in RTL with a few code lines. The implementation netlist (G1) is synthesized and optimized from the original specification (R1), and they are functionally equivalent.”
However, in test case 2, we find out that it is G1 and R2 differ in RTL with a few code lines, rather than R1 and R2. Furthermore, in test case 2, we figure out that R1 is more likely to be the optimized form, and G1 the original. We wonder that maybe these two files are given opposite. If not, we would like to ask whether the patch will be added on, R1 or G1?
A9. We have confirmed that R1 and G1 are correct. The patch needs to be applied in G1. Thank you

Q10. If I want to patch one wire which belongs to one bit of a variable, the output in the patch file should be this variable or this wire?

For example, there is a 8 bit variable y and I want to patch y[1].

Does the I/O format like the following patch file?

module top_eco (y[1], support)
output y[1];
input support;
wire support, y[1];
not eco_g1 (y[1], support)
endmodule
A10. Please note that if you want to patch on a single wire of a bus, please use \bus[0] instead of bus[0] because bus[0] is not a valid Verilog name. A valid Verilog name contained special character should start with an escape(\) and end with a space( ).
Ex:
module top_eco(\in_bus[0] ,\in_bus[1] ,\out_bus[0])
input \in_bus[0] ,\in_bus[1] ;
output \out_bus[0] ;
…
endmodule

Q11. As the previous reply "Please note that if you want to patch on a single wire of a bus, please use \bus[0] instead of bus[0] because bus[0] is not a valid Verilog name. A valid Verilog name contained special character should start with an escape(\) and end with a space( )." mentioned, I have a question: if my patch have an output "\y[0]", will the verifier patch"\y[0]" to "y[0]"?
A11. Yes. We will patch "\y[0] " to "y[0]”.

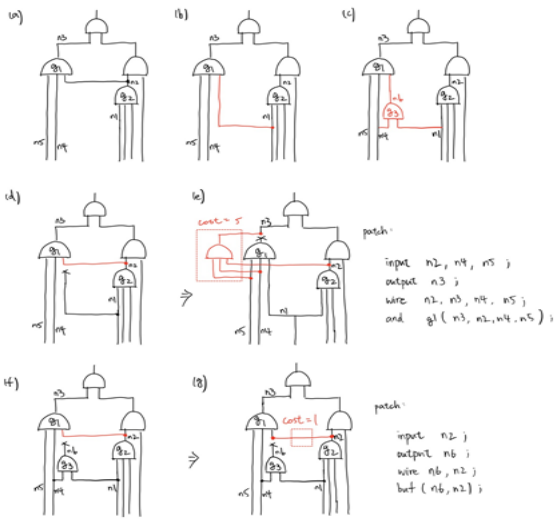
Q12. Could you help to advise if I declare a 8 bits bus in patch file but I only patch 2 bits of it, the number of wires of this bus in cost computation will be 2 or 8?
For example, the patch file is as follows. Does the cost of patch be 2 or 8?

```
module top_eco (o, in);
input [1:0] in;
output [7:0] o;
wire [1:0] in;
wire [7:0] o;
not n1 (o[0], in[0]);
not n2 (o[1], in[1]);
endmodule
```

A12. If you only want to patch 2 bits of a bus, you can patch on a single bit. For example, using "\o1[0] " and "\o1[1] ". If you patch the whole 8-bits bus, the cost will be 8.

Q13. About the patch format in Problem A, we have read that only "all-fanout changes" are allowed in Q&A.
But we found that, in some cases, the cost will be unnecessarily large with such restriction.

Here's an example of such cases.



(a) is the golden circuit.
(b) and (c) are the circuit with some error (marked in red).
More specifically, both (b) and (c) have the right most fanin of gate g1 changed.
We want to fix (b) and (c) so they are functionally equivalent to (a).

For (c), as shown in figure (f), we want to change the right most fanin of g1 to n2.
So, as shown in figure (g), we can directly change the net n6 to n2 to fix it.
The cost of the whole patch will be 1.

For (b), as shown in figure (d), we want to do the same thing to fix g1.
However, as shown in figure (e),
since both g1 and g2 have n1 as their fanin,
we have to re-declare the whole g1 (3-input AND) just to change one fanin of g1.
And the cost of the whole patch will be 5.

The only difference between the error fanin in (b) and (c) is that,
in (b), another gate (g2) is also using the fanin net "n1".
But the difference in cost is significant.

In conclusion, when a fanin net is used by some other gates,
we will have to re-declare the whole gate in our patch in order to change the fanin.
And the resulting extra cost can be significant.
Therefore, since the patch format is not in Verilog Standard and is formulated for this problem,
we suggest, with some extra rule, allowing the change of specific fanin in a gate.

We understand that there are a few reasons for the "all-fanout changes" restriction.
Since, in Verilog syntax, changing a net variable (a wire) will change all its fanout,
the restriction is very reasonable.
Also it might be easier to apply the patch with such restriction.
Please take our suggestion into consideration.
Since there might be some other reasons we didn't think of for the restriction,
we would also like to hear about it.
Thank you very much.
A13. This is a very good suggestion. We understand that all-fanout change will introduce extra cost in some cases. But just like what you say, change a net will change all its fanout. It is hard for us to define a rule of partial fix on a net. To simplify the problem, we only allowed to make all-fanout change on a net.

Q14. Regarding the question about support of patch's input ports.If we want to patch one wire which belongs to one bit of a variable(y[7]) and change overflow's fanin to y[7]. Could you help to advise which patch.v is correct?

```
//patch.v version 1
module top_eco(\y[7]_in, a, b, overflow);
  input \y[7]_in, a, b;
  output overflow, \y[7];
  nand g6133 (\y[7], a, b);
  assign overflow = \y[7]_in;
endmodule

//patch.v version 2
module top_eco(\y[7], a, b, overflow);
  input a, b;
  output overflow, \y[7];
  nand g6133 (\y[7], a, b);
  assign overflow = \y[7];
endmodule
```

A14. It depends on what is your expected fanin of the net, overflow? Before or after ECO?
If you want to connect overflow to y[7] before ECO, then version 1 is correct.
If you want to connect overflow to y[7] after ECO, then version 2 is correct.
BTW, please insert a space between \y[7] and comma.
"\y[7] ," is the correct syntax. For more detail , please refer to Verilog standard.

Q15. We have a question about the test cases. All test-cases for Problem A posted thus far are so small (less than 100 gates) and the runtime limit (3600 sec) so high that the problem may be solvable by simple enumeration. Do we expect to see larger test-cases to inspire development of more scalable methods? It would be helpful to know the approximate gate count of the largest test-case that our program has to work on. Thank you.
A15. All the testcases will be less than 1000 gates.

Q16. We have one more question about the patch signal name changes.
In Table2, the paper offers the following patch to add inverter at signal x:

```
module top_eco(x, x_in);
output x;
input x_in;
not eco1 (x, x_in);
endmodule
```

This works well when x is an internal signal, because we can rename the old signal in the original netlist

```
not n0(x, in0);
```

as follows (replacing "x" by "x_in"):

```
not n0(x_in, in0);
not eco1(x, x_in);
```

However, if x is a primary input, we cannot rename it into "x_in", because this will change the interface of the module, and to make a correct patch in this case, we would need to update several other lines in the original netlist.
Please help to provide guidance how to generate a patch, which adds an inverter at a primary input. Thank you.
A16. You can still handle the input net as other internal nets. If you want to insert an INV at the primary input 'a'. Please check the following sample patch.

```
module top_eco(a, a_in);
output a;
input a_in;
INV gate(a, a_in);
endmodule
```

To apply the patch, we will create a new wire like this.

```
module top(o, a, b, c);
output o;
input a, b, c;
wire a_1;
// AND g1(o, a, b, c);
INV gate(a_1,a);
and g1(o,a_1,b,c);
endmodule
```