Design/Practical Experience (CSN2020)

Department of Electrical Engineering
Academic Year 2021-22 — Summer Term

July 8, 2022

Abstract

The effectiveness of the patched circuit is crucial when dealing with a functional Engineering Change Order (ECO) problem. Participants in this competition must create patch functions that will equate two circuits while reducing the resource cost of the created patches. The total physical cost of all the patches is known as the resource cost, and lowering the resource cost entails raising patch quality (timing, power, routing, or area). With respect to a number of physical characteristics of nodes used for patches, the resource cost of patches can be described as a weighting function. We have given every internal node in the ICCAD 2017 CAD Contest a fair constant weight to represent the relevant physical cost if the node is used to generate patches. Additionally, the weighted total of the support nodes for the patches is used to compute the resource cost of the patches. With this formulation, desired methods for the resource-aware patch creation problem can be elegantly identified.

1 Project : Resource Aware Patch Generation

- **Name of the Mentor : Dr. Binod Kumar**
- **Name of the Students**
  - Tushar Kumar : *B20AI048*
  - Tammireddi Sai Unnathi : *B20AI045*
- **Title of the Project:** *Resource-constrained Hardware Patch Generation*
- **Project Category:** Summer/Winter/Semester projects with Institute faculty within or outside the department.
- **Targeted Deliverables**
  - Technique for generation of hardware patch under resource constraints.
  - Documentation and test-case evaluation

2 Introduction

In a modern design flow, if some functionality has to be changed or functional bugs are found at late stages, restarting the whole synthesis and verification flow is impractical. To save time and cost, automating Engineering Change Orders (ECOs) is more practical. As illustrated in Fig. 1, an automated ECO process can identify the differences between the old circuit F and the new circuit G, and generate a corresponding patch function such that F' (the resultant circuit after applying the patch to the old circuit F) and G become equivalent. In addition, patch generation plays an important role in the ECO problem because the quality of patch directly influences the performance of the patched circuit.

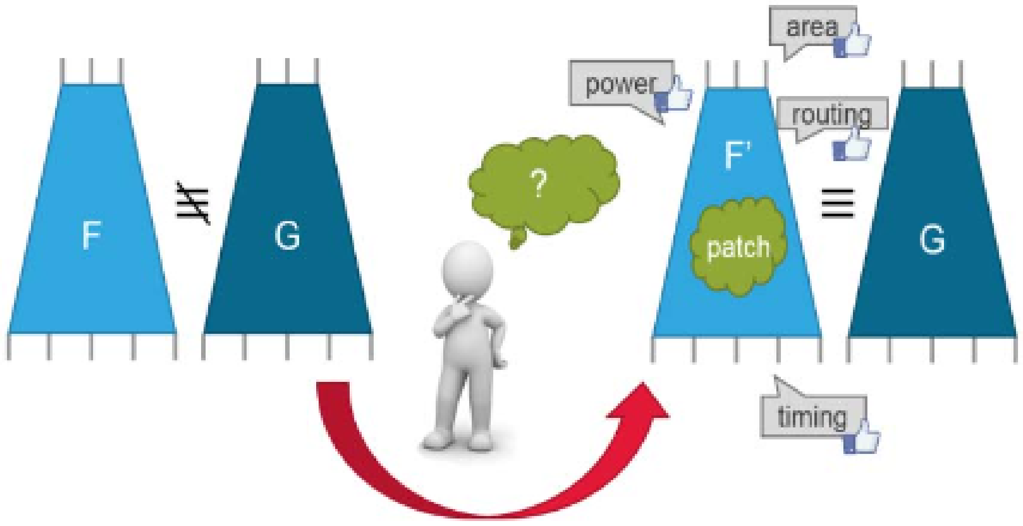


Figure 1: The resource-aware patch generation problem.

3 Problem

3.1 Objective

The objective of this project is to develop a flexible, scalable, and practical resource-aware patch generation algorithm that can be utilized in industry tools. In this project, the benchmarks are representatives of industrial problems to facilitate the academic research. Although existing work may not provide feasible solutions for these benchmarks, we look forward to inspiring new research into the functional ECO.

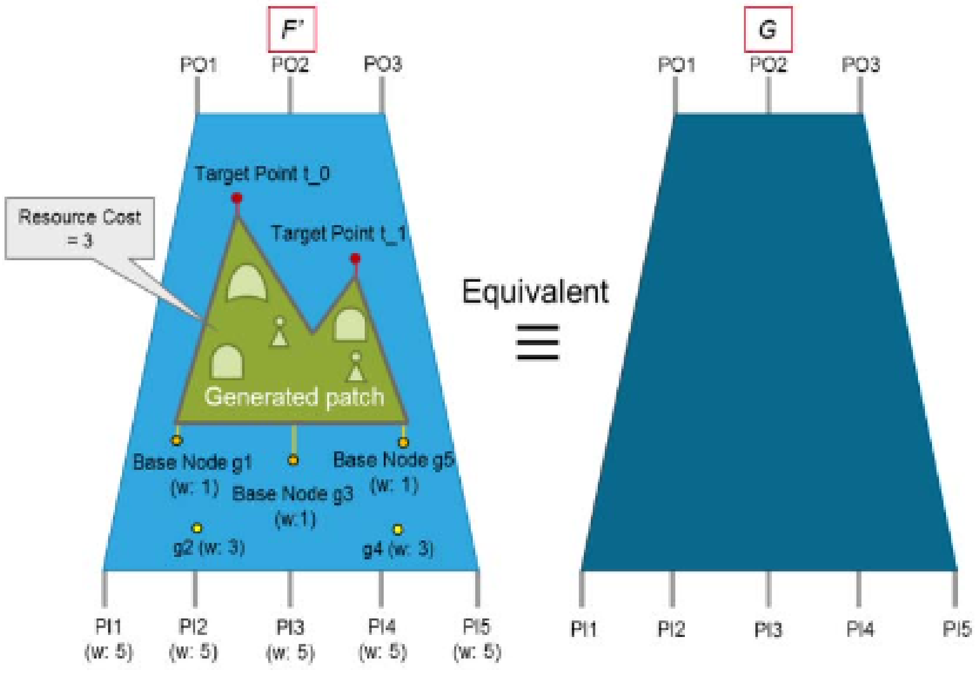


Figure 2: Problem Formulation.

3.2 Problem Statement

Given two circuits F and G, and the weight information of internal nodes in F, contestants need to utilize internal nodes in F as supports, called base nodes, to generate the patch functions with minimum resource cost at a specific set of target points in F such that F', the patched circuit, and G are equivalent, as Fig. 2 shows. The resource cost is calculated by the weight summation of the used based nodes.

3.3 Industrial Relevance

In an industrial design flow, the functional ECO tool, e.g., Cadence Encounter Conformal ECO Designer [1], has been widely used for industrial cases for years. Although the patch size is an important metric for patch quality, patch generation must consider other physical issues, including timing and power closure, to solve the ECO problem practically.

4 Work done

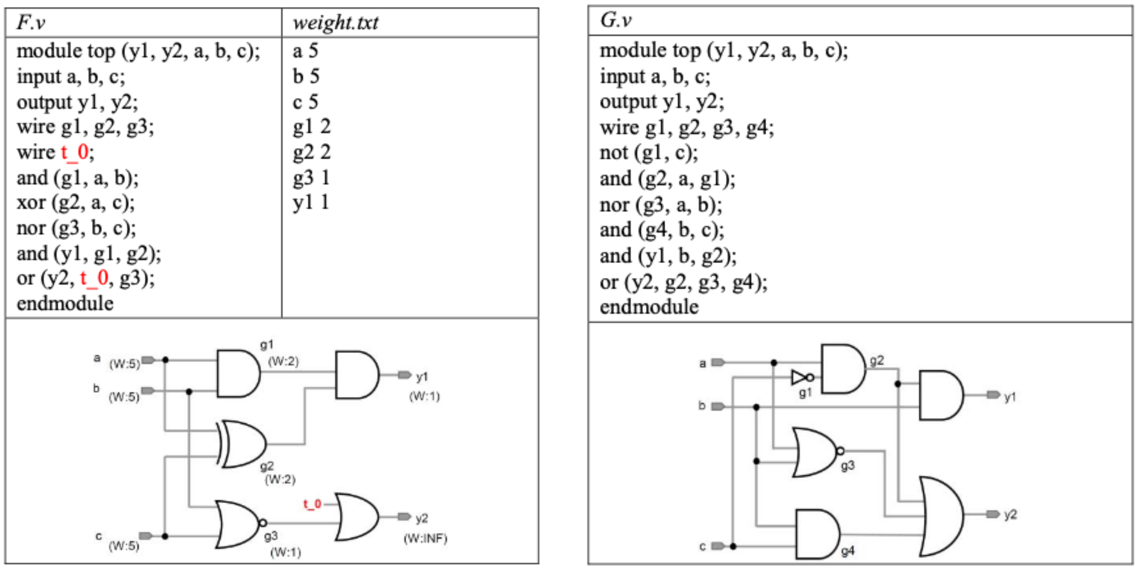
Basically, we equated the two circuits in this project by using a patch. ECOs are now employed in the modern world to expedite and reduce the time and expense associated with this activity. The difference between the old circuit F and the new circuit G can be identified by an automated ECO process, as indicated in the figure above, and a corresponding patch function can then be generated to make the old circuit F and the new circuit G equivalent.

4.1 Different Available Approaches

There were several approaches available to tackle the given problem. Let's analyse them with the help of an example.

4.1.1 Example

Here we use an example to illustrate the problem description, different approaches available and how to calculate the resource cost. In this example, there are two designs F.v and G.v and one weight information file weight.txt for F.v, as shown in figure.



F.v

```
module top (y1, y2, a, b, c);
input a, b, c;
output y1, y2;
wire g1, g2, g3;
wire t_0;
and (g1, a, b);
xor (g2, a, c);
nor (g3, b, c);
and (y1, g1, g2);
or (y2, t_0, g3);
endmodule
```

weight.txt

```
a 5
b 5
c 5
g1 2
g2 2
g3 1
y1 1
```

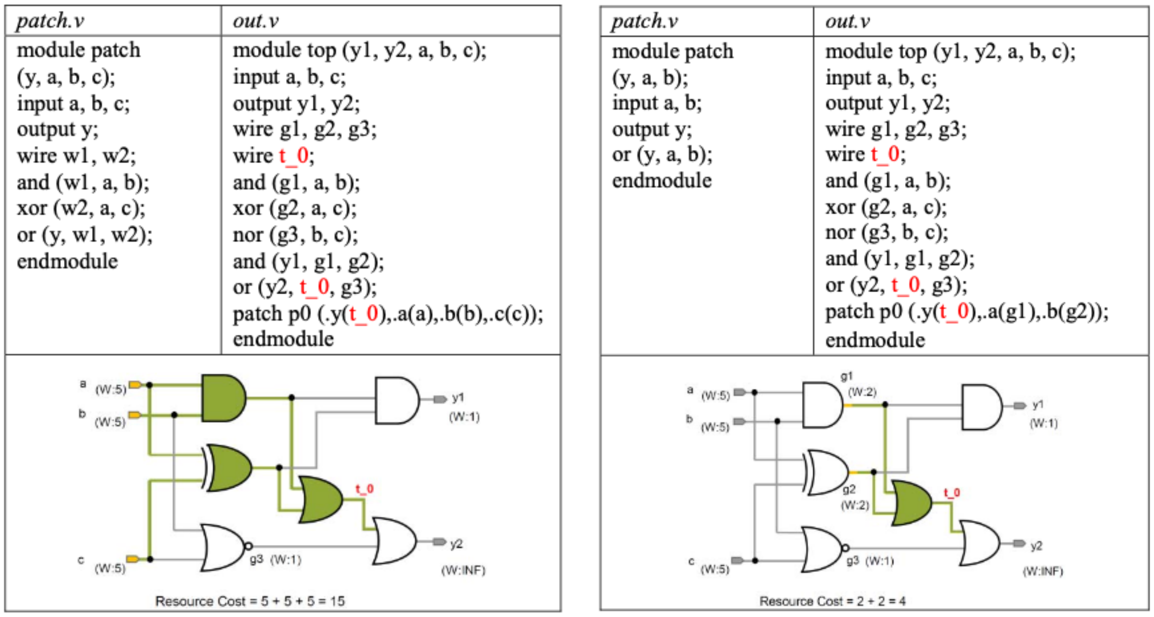
G.v

```
module top (y1, y2, a, b, c);
input a, b, c;
output y1, y2;
wire g1, g2, g3, g4;
not (g1, c);
and (g2, a, g1);
nor (g3, a, b);
and (g4, b, c);
and (y1, b, g2);
or (y2, g2, g3, g4);
endmodule
```

Figure 3: Sample F.v and G.v Circuits

4.1.2 Approach 1 : Generating patch from inputs

One approach would have been as shown in Fig. 4, uses a, b, c as the base nodes to generate the patch at t_0. According to the file weight.txt, the weights of nodes a, b, c are 5, 5, 5, respectively. Thus, the corresponding resource cost of the patch is 5 + 5 + 5 = 15. Besides, since there are 3 primitive gates in ¡patch.v¿, the corresponding patch size is 3.



patch.v
```
module patch
(y, a, b, c);
input a, b, c;
output y;
wire w1, w2;
and (w1, a, b);
xor (w2, a, c);
or (y, w1, w2);
endmodule
```

out.v
```
module top (y1, y2, a, b, c);
input a, b, c;
output y1, y2;
wire g1, g2, g3;
wire t_0;
and (g1, a, b);
xor (g2, a, c);
nor (g3, b, c);
and (y1, g1, g2);
or (y2, t_0, g3);
patch p0 (.y(t_0),.a(a),.b(b),.c(c));
endmodule
```

patch.v
```
module patch
(y, a, b);
input a, b;
output y;
or (y, a, b);
endmodule
```

out.v
```
module top (y1, y2, a, b, c);
input a, b, c;
output y1, y2;
wire g1, g2, g3;
wire t_0;
and (g1, a, b);
xor (g2, a, c);
nor (g3, b, c);
and (y1, g1, g2);
or (y2, t_0, g3);
patch p0 (.y(t_0),.a(g1),.b(g2));
endmodule
```

Figure 4: Approaches available to generate patch

4.1.3 Approach 2 — *Our approach* : Using the very last level wires

Our solution first levelize the circuit and then for optimal patch generation use last level available wires output to generate patch. As shown in Fig. 7, uses g1, g2 as the base nodes to generate the patch at t_0. According to the file weight.txt, the weights of nodes g1, g2 are 2, 2, respectively. Thus, the corresponding resource cost of the patch is 2 + 2 = 4. Besides, the corresponding patch size is 1 because there is only one gate in ¡patch.v¿.

5 Methodology

We approached the issue methodically. The most crucial actions were...

5.1 Simulator Development

To mimic the circuit run, we first created a circuit simulator. Since we chose to generate the patch using previous level wires for better results, levelizing the circuit was necessary while constructing the simulator and was helpful for achieving the best results. We used a greedy search method to find this fix.

5.2 Greedy Search

Given the resource constraint hardware patch generation problem , we were given wires and gates to use. In order to cope with the issue and find the best answer, Greedy Search was one obvious option.So, what we did was take the wires from the circuit's very previous level that had known outputs. To create the patch, from weight.txt we attempted using the cheapest gates and wires. In one iteration, we first finish the F.v circuit at the lowest cost, compare it to the G.v circuit, and then, if there are any conflicts, go back. After several iteration the optimal patch was created as a result..

5.3 Comparing Circuits

After the patch was created, we used the random test generator to test its equivalency by comparing all of the output values. They were all on an equal footing. We have used ABC library that is a system for Sequential Synthesis and Verification

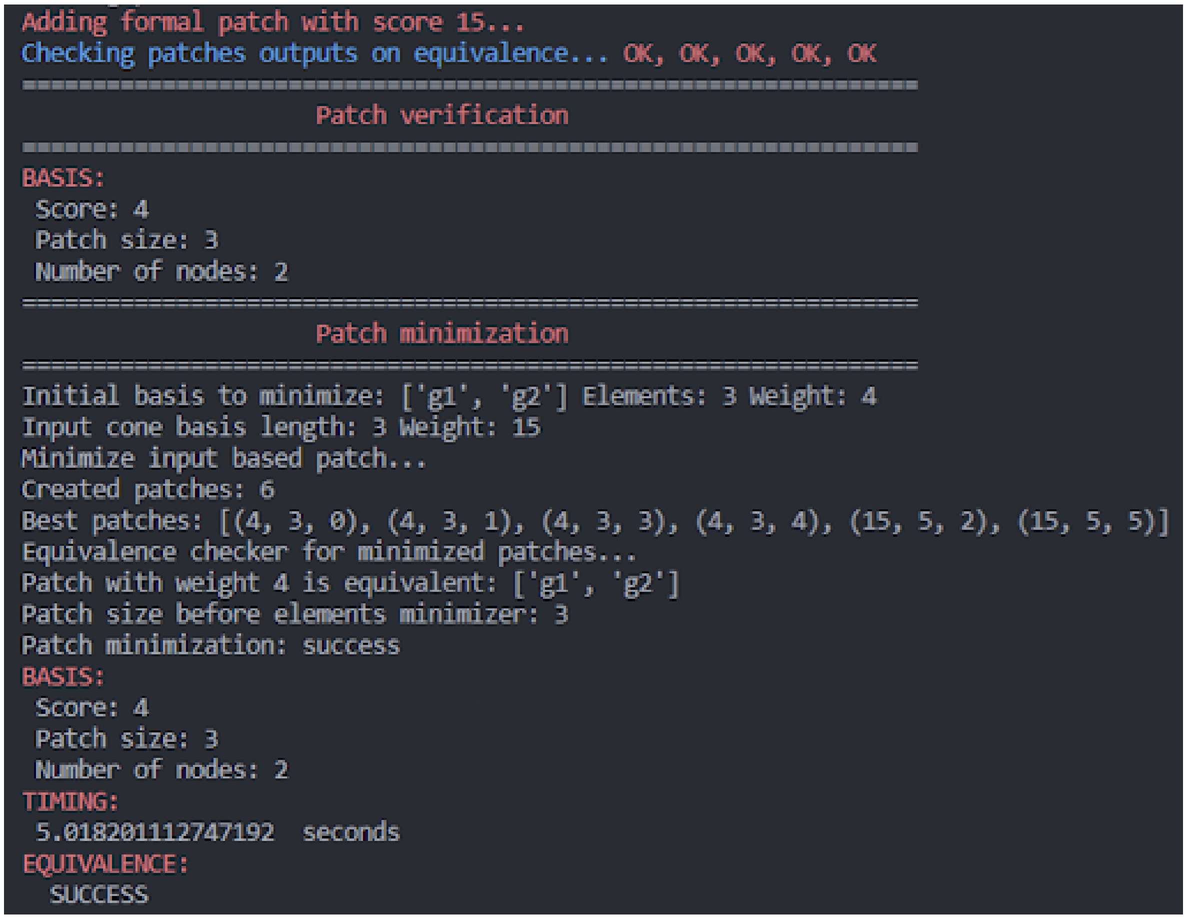
6 Results

We were able to generate the patch for all test cases provided by the contest authorities and that to a very good cost.

6.1 Example Solution

For the example in *section 4.1* in the aforementioned figures, as can be seen, are of test case 1. The original circuit is the G.V circuit, followed by the F.V circuit, which has a break point at t o. At this point, we need to add a patch to make the G.V. and F.V. circuits equivalent, meaning they both produce the same outputs from the same set of inputs. Thus, after running the code, the necessary code is obtained in verilog code format in the results folder (together with the verilog code of the patch), and after conversion to a logical circuit, it resembles the output.v circuit shown above. So, for this test scenario, this is the solution that we must provide.

As can be seen from this terminal snippet we were able to generate the patch using previous level wires at a very good cost.



```
Adding formal patch with score 15...
Checking patches outputs on equivalence... OK, OK, OK, OK, OK
=================================================================
                     Patch verification
=================================================================
BASIS:
 Score: 4
 Patch size: 3
 Number of nodes: 2
=================================================================
                     Patch minimization
=================================================================
Initial basis to minimize: ['g1', 'g2'] Elements: 3 Weight: 4
Input cone basis length: 3 Weight: 15
Minimize input based patch...
Created patches: 6
Best patches: [(4, 3, 0), (4, 3, 1), (4, 3, 3), (4, 3, 4), (15, 5, 2), (15, 5, 5)]
Equivalence checker for minimized patches...
Patch with weight 4 is equivalent: ['g1', 'g2']
Patch size before elements minimizer: 3
Patch minimization: success
BASIS:
 Score: 4
 Patch size: 3
 Number of nodes: 2
TIMING:
 5.018201112747192  seconds
EQUIVALENCE:
  SUCCESS
```

Figure 5: Terminal output for example problem

6.1.1 Patch Generated

Our algorithm generated the following patch for the F.v and G.v given in figure 3.

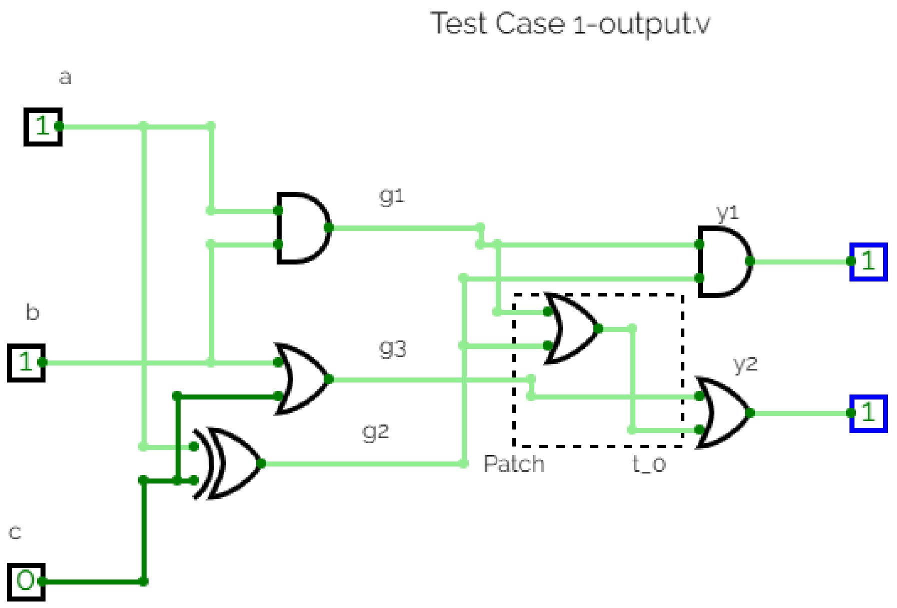


Figure 6: Output for example problem

7 Conclusion

The effectiveness of the patched circuit in the functional ECO problem is significantly influenced by the quality of the patches. Infact that was the main reason they formulated a patch generation issue for the ICCAD 2017 CAD Contest while considering resource costs.
While talking about our solution we were able to generate the patch at very reasonable cost that is a good step towards the future progress. Although existing work may not provide feasible solutions for these benchmarks, we look forward to inspiring new research into the functional ECO.

8 References

We have used many resources available for this projects. Some of the most important refrences are..

- Contest Website link
- Functional ECO Paper link
- Contest Paper link
- Github Resource Aware Patch Generation