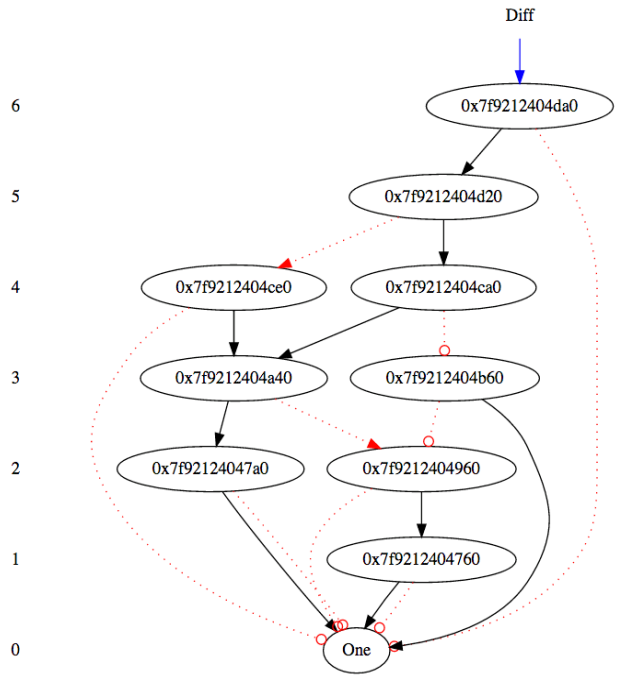
P6d

電機系　　李友岐

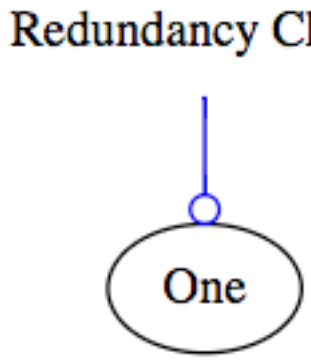
(1)



Above is the BDD for Diff(g9|g5), and it is not constant zero obviously.
The main reason is that the definition of redundancy doesn't include Diff()=0 .
The definition of redundancy is that there exists an undetectable fault.
After testing, I found that g5-stuck-at-1 fault is undetectable, which means
~g5& Diff(g9|g5) =0 . Thus, I revised the code like below.

```
BddNode Diff = (g8 & (f & BddNode::_zero))^(g8 & (f & BddNode::_one));
BddNode Redundancy=Diff & ~g5;
cout << "Redundancy Check:" << endl;
cout << Redundancy << endl;
```

And the result BDD became constant zero.

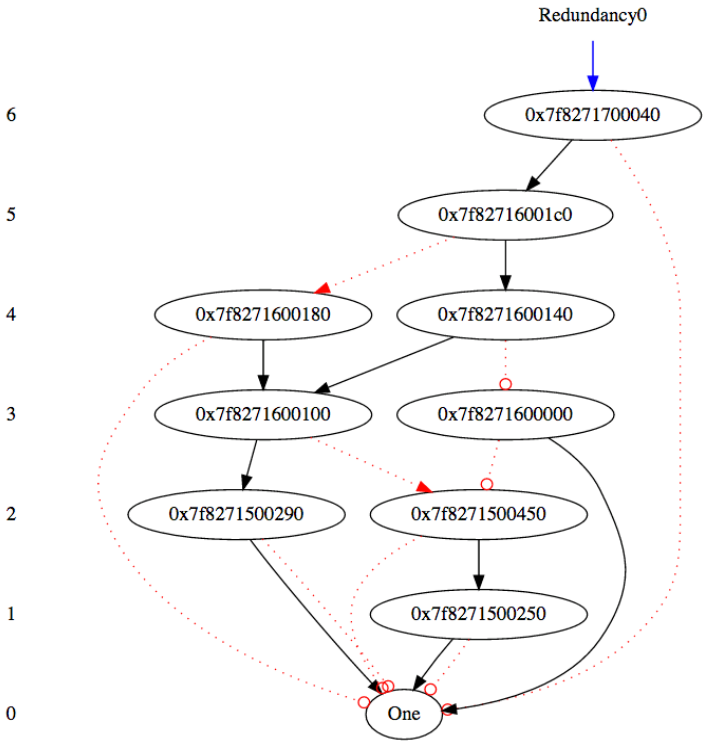


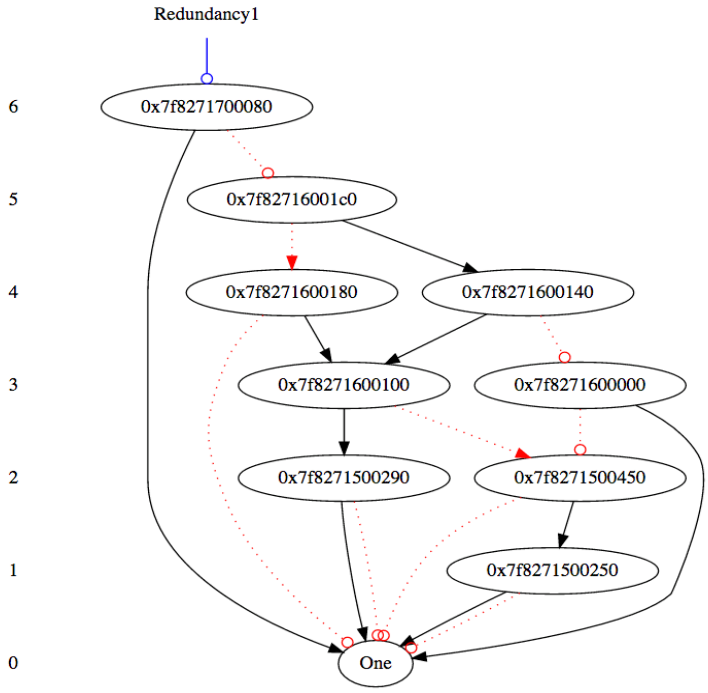
Therefore, g5 -> g9 is redundant.

(2)

If I replace g5 by f, then the result BDD is not constant zero

```
BddNode Diff = (g8 & (g5 & BddNode::_zero))^(g8 & (g5 & BddNode::_one));
BddNode Redundancy1=Diff & ~f;
BddNode Redundancy0=Diff & f;
cout << "Redundancy Check:" << endl;
cout << Redundancy1 << endl;
cout << Redundancy0 << endl;
```





Both f-stuck-at-0 fault and f-stuck-at-1 fault are detectable, so f -> g9 is irredundant.