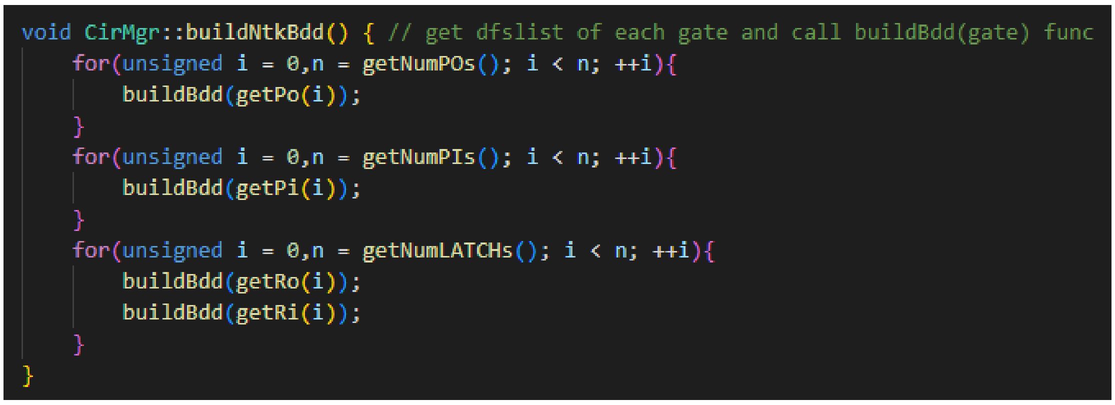
SoCV HW3

B10901085 李昀儒

1)Implementation of the functions:

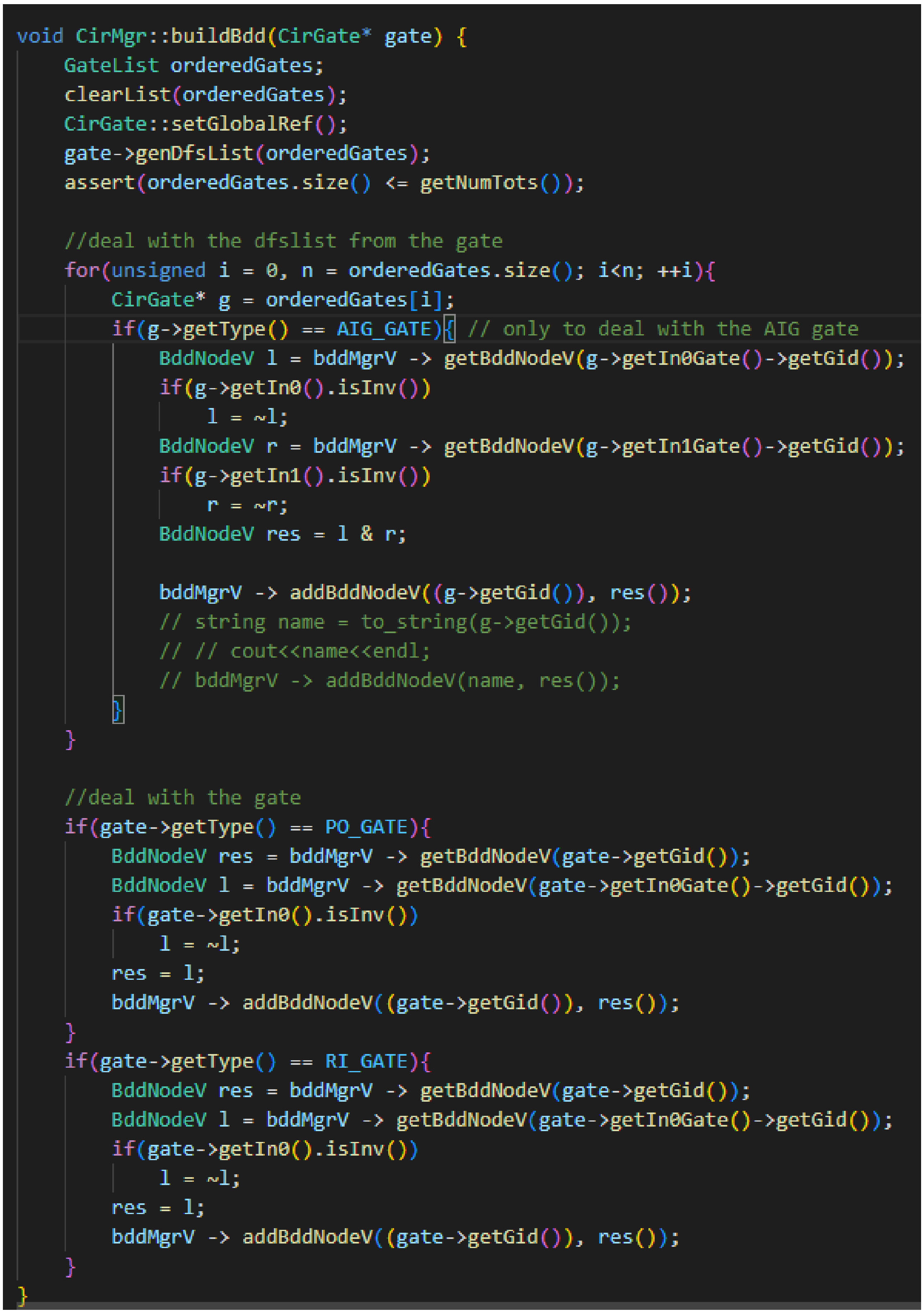
cirBdd.cpp:

(i)



```
void CirMgr::buildNtkBdd() { // get dfslist of each gate and call buildBdd(gate) func
    for(unsigned i = 0,n = getNumPOs(); i < n; ++i){
        buildBdd(getPo(i));
    }
    for(unsigned i = 0,n = getNumPIs(); i < n; ++i){
        buildBdd(getPi(i));
    }
    for(unsigned i = 0,n = getNumLATCHs(); i < n; ++i){
        buildBdd(getRo(i));
        buildBdd(getRi(i));
    }
}
```

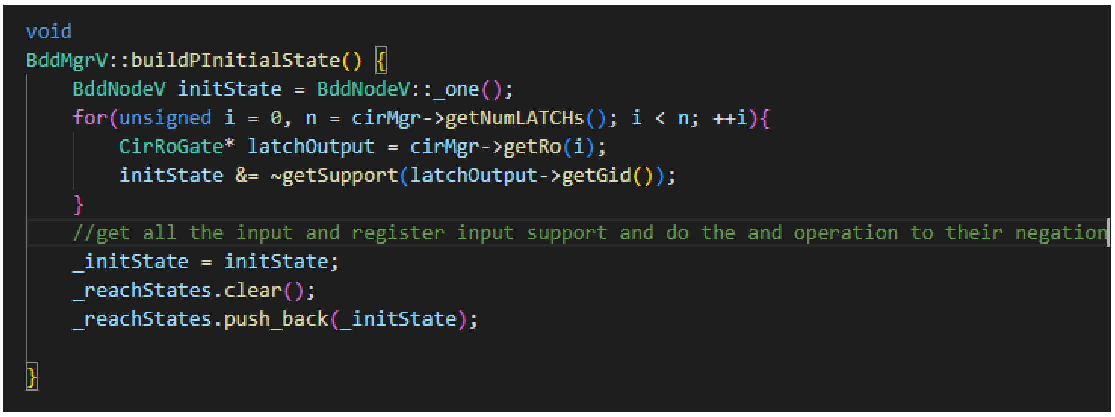
(ii)



```
void CirMgr::buildBdd(CirGate* gate) {
    GateList orderedGates;
    clearList(orderedGates);
    CirGate::setGlobalRef();
    gate->genDfsList(orderedGates);
    assert(orderedGates.size() <= getNumTots());

    //deal with the dfslist from the gate
    for(unsigned i = 0, n = orderedGates.size(); i<n; ++i){
        CirGate* g = orderedGates[i];
        if(g->getType() == AIG_GATE){ // only to deal with the AIG gate
            BddNodeV l = bddMgrV -> getBddNodeV(g->getIn0Gate()->getGid());
            if(g->getIn0().isInv())
                l = ~l;
            BddNodeV r = bddMgrV -> getBddNodeV(g->getIn1Gate()->getGid());
            if(g->getIn1().isInv())
                r = ~r;
            BddNodeV res = l & r;

            bddMgrV -> addBddNodeV((g->getGid()), res());
            // string name = to_string(g->getGid());
            // // cout<<name<<endl;
            // bddMgrV -> addBddNodeV(name, res());
        }
    }

    //deal with the gate
    if(gate->getType() == PO_GATE){
        BddNodeV res = bddMgrV -> getBddNodeV(gate->getGid());
        BddNodeV l = bddMgrV -> getBddNodeV(gate->getIn0Gate()->getGid());
        if(gate->getIn0().isInv())
            l = ~l;
        res = l;
        bddMgrV -> addBddNodeV((gate->getGid()), res());
    }
    if(gate->getType() == RI_GATE){
        BddNodeV res = bddMgrV -> getBddNodeV(gate->getGid());
        BddNodeV l = bddMgrV -> getBddNodeV(gate->getIn0Gate()->getGid());
        if(gate->getIn0().isInv())
            l = ~l;
        res = l;
        bddMgrV -> addBddNodeV((gate->getGid()), res());
    }
}
```

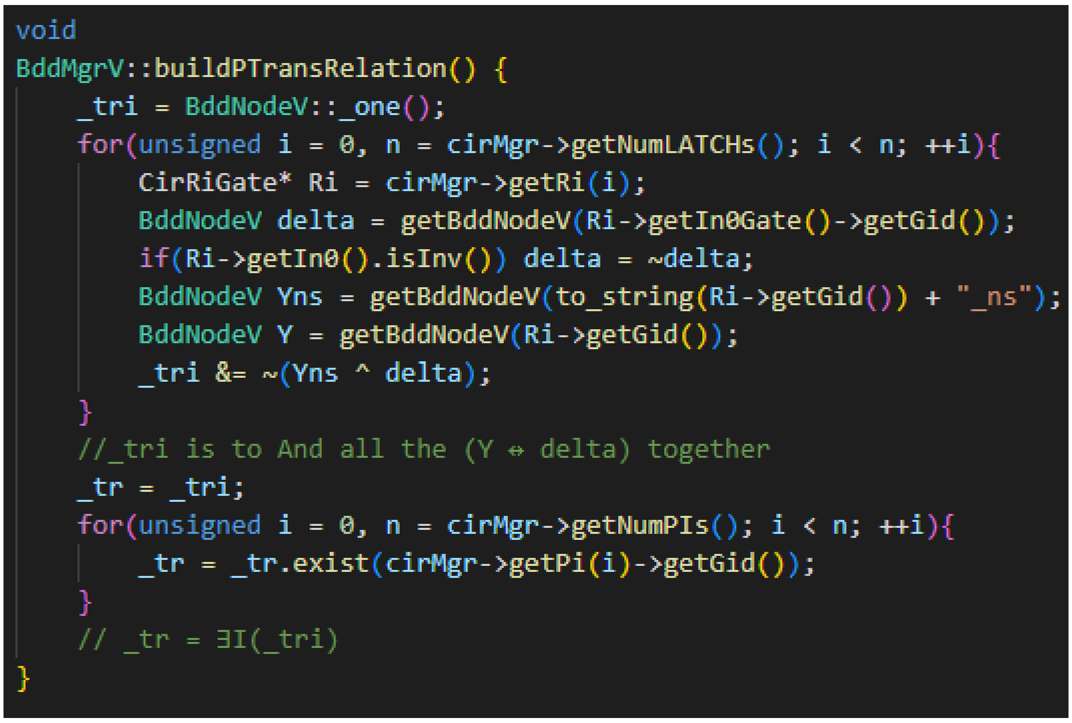
proveBdd.cpp

(i)



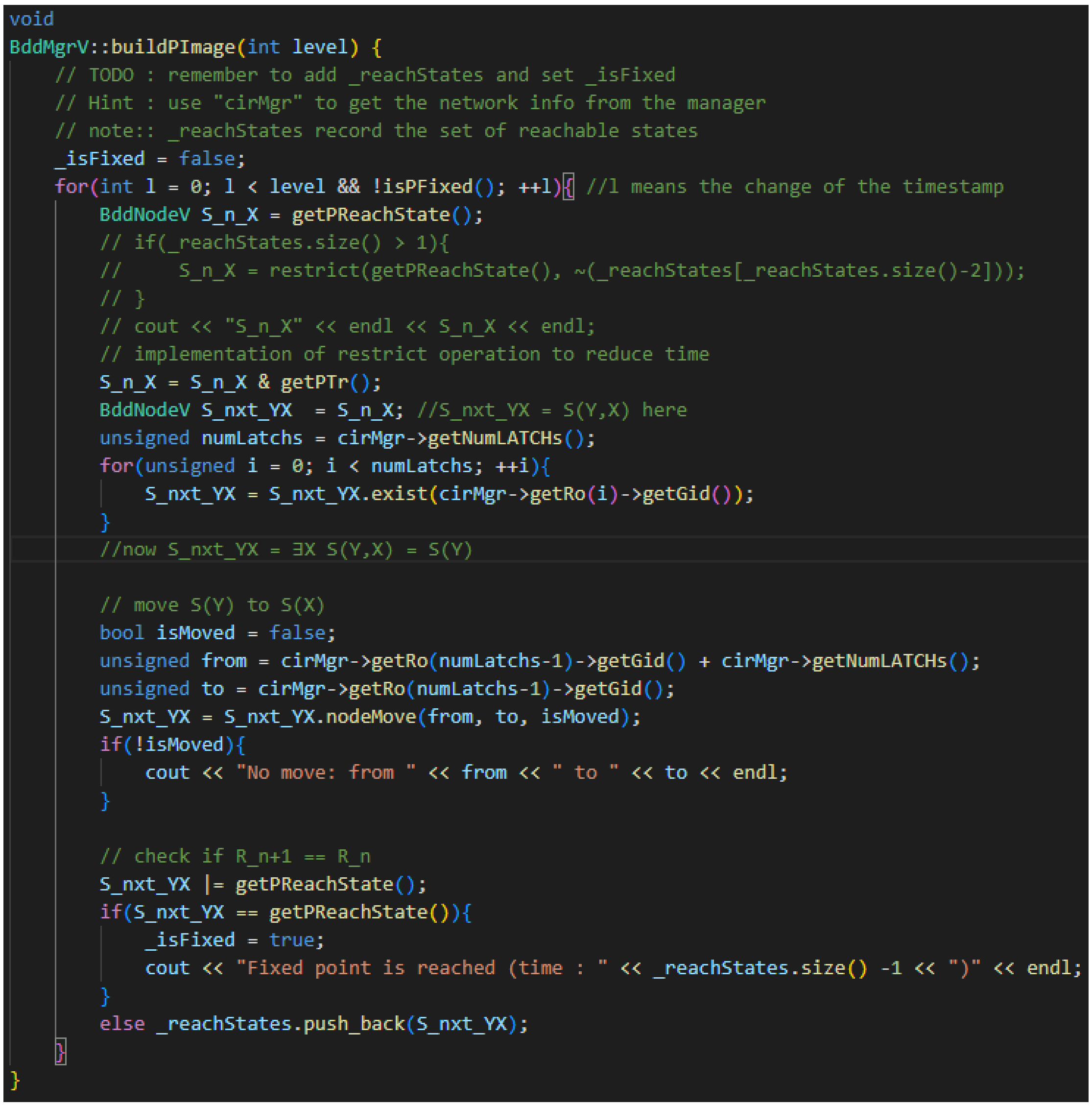
```cpp
void
BddMgrV::buildPInitialState() {
    BddNodeV initState = BddNodeV::_one();
    for(unsigned i = 0, n = cirMgr->getNumLATCHs(); i < n; ++i){
        CirRoGate* latchOutput = cirMgr->getRo(i);
        initState &= ~getSupport(latchOutput->getGid());
    }
    //get all the input and register input support and do the and operation to their negation
    _initState = initState;
    _reachStates.clear();
    _reachStates.push_back(_initState);

}
```

(ii)



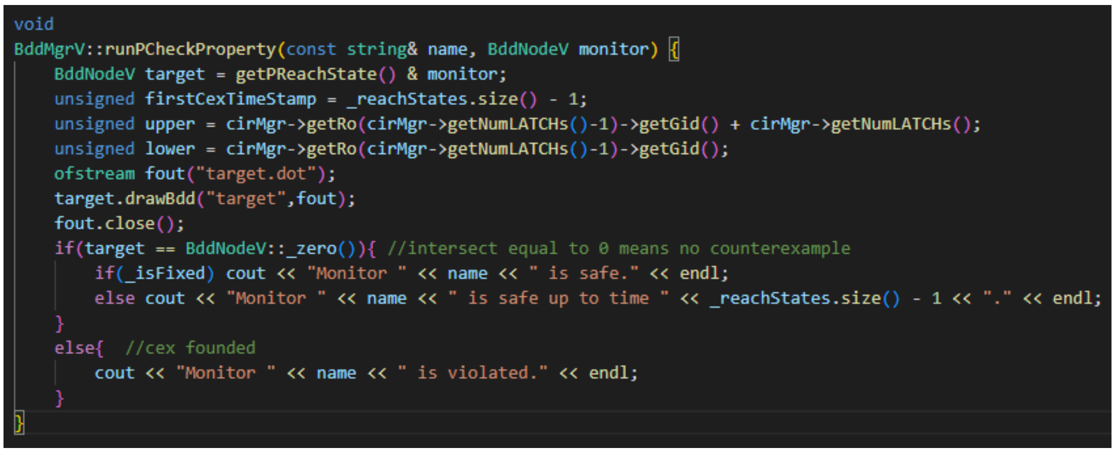
```cpp
void
BddMgrV::buildPTransRelation() {
    _tri = BddNodeV::_one();
    for(unsigned i = 0, n = cirMgr->getNumLATCHs(); i < n; ++i){
        CirRiGate* Ri = cirMgr->getRi(i);
        BddNodeV delta = getBddNodeV(Ri->getIn0Gate()->getGid());
        if(Ri->getIn0().isInv()) delta = ~delta;
        BddNodeV Yns = getBddNodeV(to_string(Ri->getGid()) + "_ns");
        BddNodeV Y = getBddNodeV(Ri->getGid());
        _tri &= ~(Yns ^ delta);
    }
    //_tri is to And all the (Y ↔ delta) together
    _tr = _tri;
    for(unsigned i = 0, n = cirMgr->getNumPIs(); i < n; ++i){
        _tr = _tr.exist(cirMgr->getPi(i)->getGid());
    }
    // _tr = ∃I(_tri)
}
```

(iii)



```cpp
void
BddMgrV::buildPImage(int level) {
    // TODO : remember to add _reachStates and set _isFixed
    // Hint : use "cirMgr" to get the network info from the manager
    // note:: _reachStates record the set of reachable states
    _isFixed = false;
    for(int l = 0; l < level && !isPFixed(); ++l){ //l means the change of the timestamp
        BddNodeV S_n_X = getPReachState();
        // if(_reachStates.size() > 1){
        //     S_n_X = restrict(getPReachState(), ~(_reachStates[_reachStates.size()-2]));
        // }
        // cout << "S_n_X" << endl << S_n_X << endl;
        // implementation of restrict operation to reduce time
        S_n_X = S_n_X & getPTr();
        BddNodeV S_nxt_YX  = S_n_X; //S_nxt_YX = S(Y,X) here
        unsigned numLatchs = cirMgr->getNumLATCHs();
        for(unsigned i = 0; i < numLatchs; ++i){
            S_nxt_YX = S_nxt_YX.exist(cirMgr->getRo(i)->getGid());
        }
        //now S_nxt_YX = ∃X S(Y,X) = S(Y)

        // move S(Y) to S(X)
        bool isMoved = false;
        unsigned from = cirMgr->getRo(numLatchs-1)->getGid() + cirMgr->getNumLATCHs();
        unsigned to = cirMgr->getRo(numLatchs-1)->getGid();
        S_nxt_YX = S_nxt_YX.nodeMove(from, to, isMoved);
        if(!isMoved){
            cout << "No move: from " << from << " to " << to << endl;
        }

        // check if R_n+1 == R_n
        S_nxt_YX |= getPReachState();
        if(S_nxt_YX == getPReachState()){
            _isFixed = true;
            cout << "Fixed point is reached (time : " << _reachStates.size() -1 << ")" << endl;
        }
        else _reachStates.push_back(S_nxt_YX);
    }
}
```

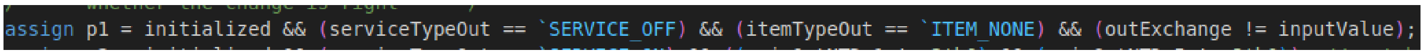
(iv)



```cpp
void
BddMgrV::runPCheckProperty(const string& name, BddNodeV monitor) {
    BddNodeV target = getPReachState() & monitor;
    unsigned firstCexTimeStamp = _reachStates.size() - 1;
    unsigned upper = cirMgr->getRo(cirMgr->getNumLATCHs()-1)->getGid() + cirMgr->getNumLATCHs();
    unsigned lower = cirMgr->getRo(cirMgr->getNumLATCHs()-1)->getGid();
    ofstream fout("target.dot");
    target.drawBdd("target",fout);
    fout.close();
    if(target == BddNodeV::_zero()){ //intersect equal to 0 means no counterexample
        if(_isFixed) cout << "Monitor " << name << " is safe." << endl;
        else cout << "Monitor " << name << " is safe up to time " << _reachStates.size() - 1 << "." << endl;
    }
    else{  //cex founded
        cout << "Monitor " << name << " is violated." << endl;
    }
}
```

2) The assertions and their meaning

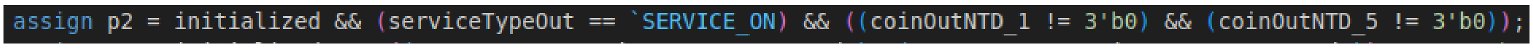
(i)



```
assign p1 = initialized && (serviceTypeOut == `SERVICE_OFF) && (itemTypeOut == `ITEM_NONE) && (outExchange != inputValue);
```

Meaning: The exchange coming from the machine should always equal to the inputValue when no item is bought.

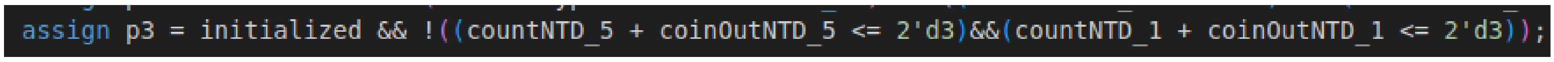
(ii)



```
assign p2 = initialized && (serviceTypeOut == `SERVICE_ON) && ((coinOutNTD_1 != 3'b0) && (coinOutNTD_5 != 3'b0));
```

Meaning: When the machine is still calculating the exchange, no coins should comes out from the machine.

(iii)



```
assign p3 = initialized && !((countNTD_5 + coinOutNTD_5 <= 2'd3)&&(countNTD_1 + coinOutNTD_1 <= 2'd3));
```

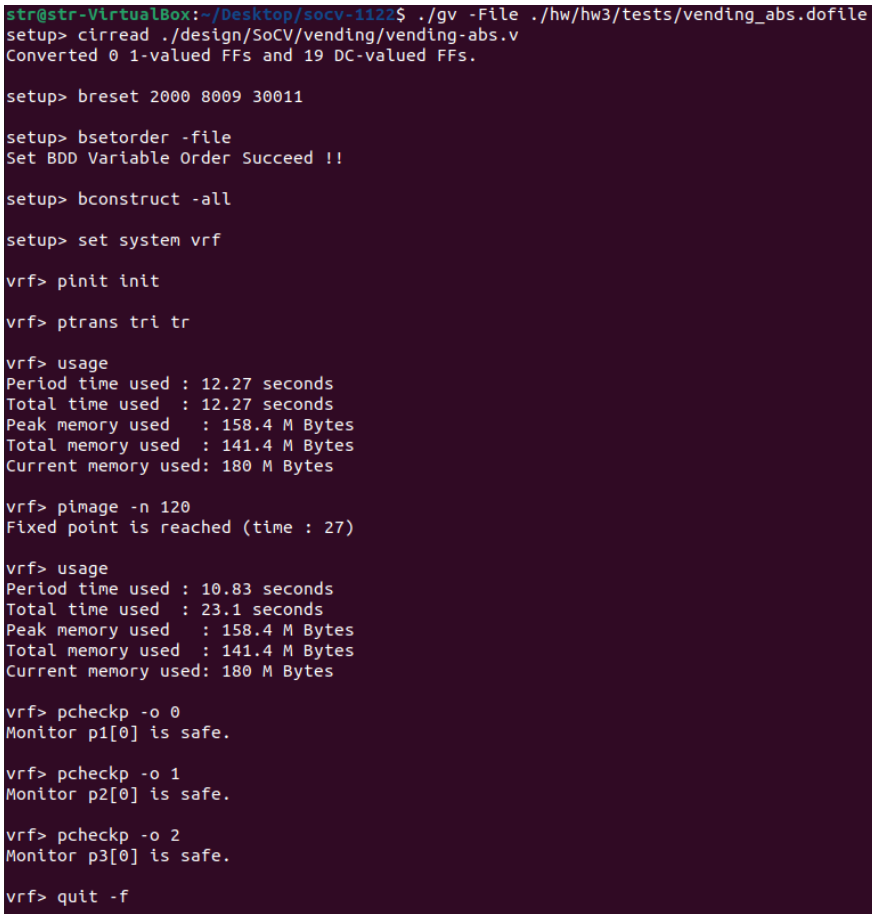
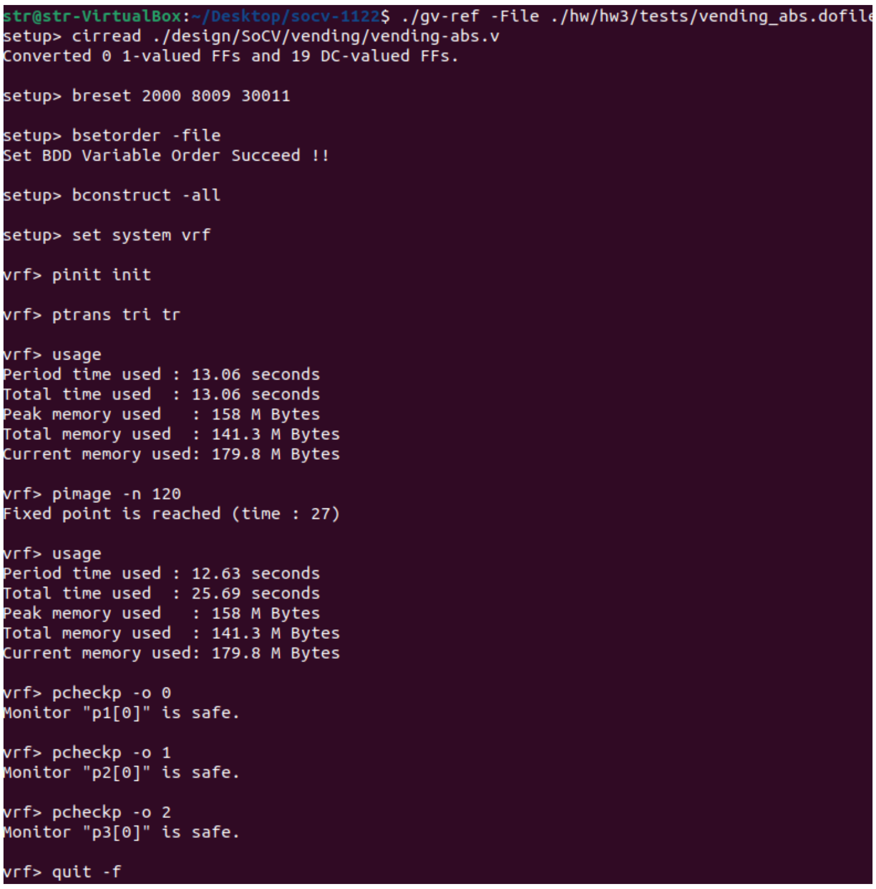
Meaning: For each type of the coins, the sum of coins in the count and coins coming out from machine should be less than 3(capacity of the count).

(3) (4) Verification results and the comparison with the ref program.

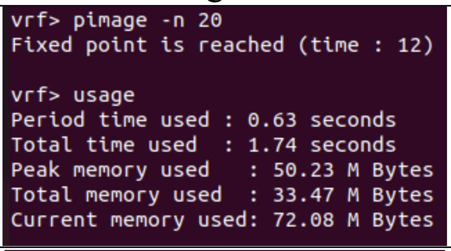
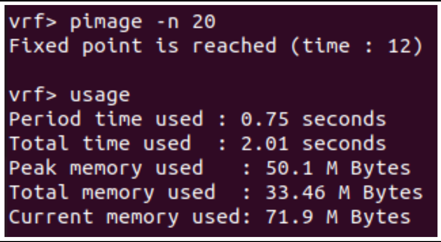
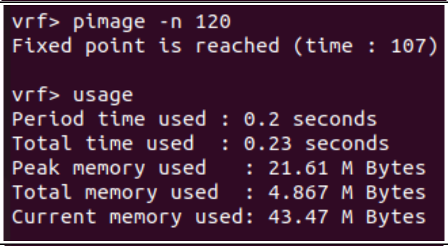
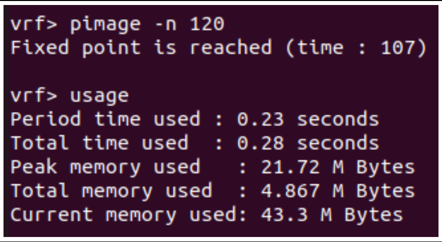
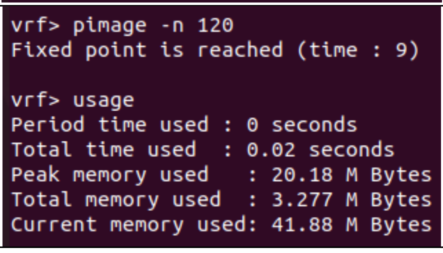
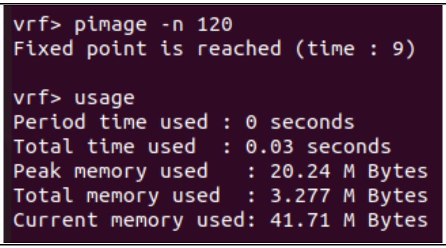
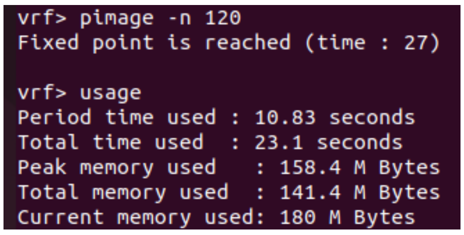
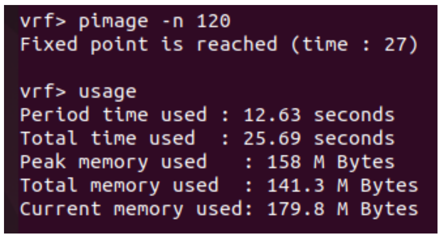
For the original design of vending.v, neither the implemented program(gv) nor ref program(gv-ref) can deal with the file properly. So we simplify some of the design:

i) Types of coins reduced to only $1 and $5.
ii) Types of the item reduced to only ITEM_A and change it price.
iii) Capacity of count reduced from 7 to 3.
iv) Reduce the maximum numbers of the input coins

In the end, the number of variables decreased and the number of the register also reduced to less than 20, making both gv and gv-ref be able to run and test on the file. And the two programs both verify that the above three assertions are safe.


```
str@str-VirtualBox:~/Desktop/socv-1122$ ./gv -File ./hw/hw3/tests/vending_abs.dofile
setup> cirread ./design/SoCV/vending/vending-abs.v
Converted 0 1-valued FFs and 19 DC-valued FFs.

setup> breset 2000 8009 30011

setup> bsetorder -file
Set BDD Variable Order Succeed !!

setup> bconstruct -all

setup> set system vrf

vrf> pinit init

vrf> ptrans tri tr

vrf> usage
Period time used : 12.27 seconds
Total time used  : 12.27 seconds
Peak memory used   : 158.4 M Bytes
Total memory used  : 141.4 M Bytes
Current memory used: 180 M Bytes

vrf> pimage -n 120
Fixed point is reached (time : 27)

vrf> usage
Period time used : 10.83 seconds
Total time used  : 23.1 seconds
Peak memory used   : 158.4 M Bytes
Total memory used  : 141.4 M Bytes
Current memory used: 180 M Bytes

vrf> pcheckp -o 0
Monitor p1[0] is safe.

vrf> pcheckp -o 1
Monitor p2[0] is safe.

vrf> pcheckp -o 2
Monitor p3[0] is safe.

vrf> quit -f
```

```
str@str-VirtualBox:~/Desktop/socv-1122$ ./gv-ref -File ./hw/hw3/tests/vending_abs.dofile
setup> cirread ./design/SoCV/vending/vending-abs.v
Converted 0 1-valued FFs and 19 DC-valued FFs.

setup> breset 2000 8009 30011

setup> bsetorder -file
Set BDD Variable Order Succeed !!

setup> bconstruct -all

setup> set system vrf

vrf> pinit init

vrf> ptrans tri tr

vrf> usage
Period time used : 13.06 seconds
Total time used  : 13.06 seconds
Peak memory used   : 158 M Bytes
Total memory used  : 141.3 M Bytes
Current memory used: 179.8 M Bytes

vrf> pimage -n 120
Fixed point is reached (time : 27)

vrf> usage
Period time used : 12.63 seconds
Total time used  : 25.69 seconds
Peak memory used   : 158 M Bytes
Total memory used  : 141.3 M Bytes
Current memory used: 179.8 M Bytes

vrf> pcheckp -o 0
Monitor "p1[0]" is safe.

vrf> pcheckp -o 1
Monitor "p2[0]" is safe.

vrf> pcheckp -o 2
Monitor "p3[0]" is safe.

vrf> quit -f
```

Below are the time both programs takes to perform pimage on different designs:

	gv	gv-ref
a.v		
b.v		
c.v		
vending -abs.v		

a.v, gv:
```
vrf> pimage -n 20
Fixed point is reached (time : 12)

vrf> usage
Period time used : 0.63 seconds
Total time used  : 1.74 seconds
Peak memory used   : 50.23 M Bytes
Total memory used  : 33.47 M Bytes
Current memory used: 72.08 M Bytes
```

a.v, gv-ref:
```
vrf> pimage -n 20
Fixed point is reached (time : 12)

vrf> usage
Period time used : 0.75 seconds
Total time used  : 2.01 seconds
Peak memory used   : 50.1 M Bytes
Total memory used  : 33.46 M Bytes
Current memory used: 71.9 M Bytes
```

b.v, gv:
```
vrf> pimage -n 120
Fixed point is reached (time : 107)

vrf> usage
Period time used : 0.2 seconds
Total time used  : 0.23 seconds
Peak memory used   : 21.61 M Bytes
Total memory used  : 4.867 M Bytes
Current memory used: 43.47 M Bytes
```

b.v, gv-ref:
```
vrf> pimage -n 120
Fixed point is reached (time : 107)

vrf> usage
Period time used : 0.23 seconds
Total time used  : 0.28 seconds
Peak memory used   : 21.72 M Bytes
Total memory used  : 4.867 M Bytes
Current memory used: 43.3 M Bytes
```

c.v, gv:
```
vrf> pimage -n 120
Fixed point is reached (time : 9)

vrf> usage
Period time used : 0 seconds
Total time used  : 0.02 seconds
Peak memory used   : 20.18 M Bytes
Total memory used  : 3.277 M Bytes
Current memory used: 41.88 M Bytes
```

c.v, gv-ref:
```
vrf> pimage -n 120
Fixed point is reached (time : 9)

vrf> usage
Period time used : 0 seconds
Total time used  : 0.03 seconds
Peak memory used   : 20.24 M Bytes
Total memory used  : 3.277 M Bytes
Current memory used: 41.71 M Bytes
```

vending-abs.v, gv:
```
vrf> pimage -n 120
Fixed point is reached (time : 27)

vrf> usage
Period time used : 10.83 seconds
Total time used  : 23.1 seconds
Peak memory used   : 158.4 M Bytes
Total memory used  : 141.4 M Bytes
Current memory used: 180 M Bytes
```

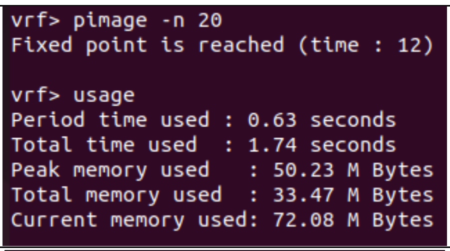
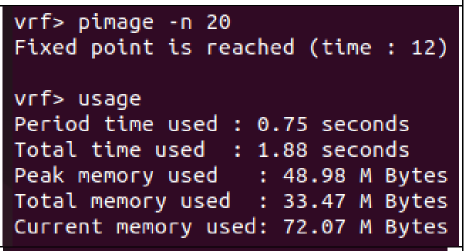
vending-abs.v, gv-ref:
```
vrf> pimage -n 120
Fixed point is reached (time : 27)

vrf> usage
Period time used : 12.63 seconds
Total time used  : 25.69 seconds
Peak memory used   : 158 M Bytes
Total memory used  : 141.3 M Bytes
Current memory used: 179.8 M Bytes
```

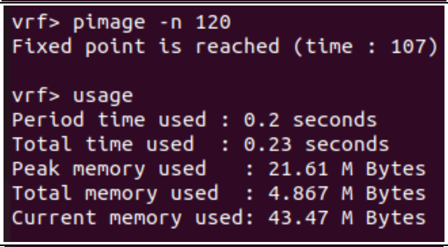
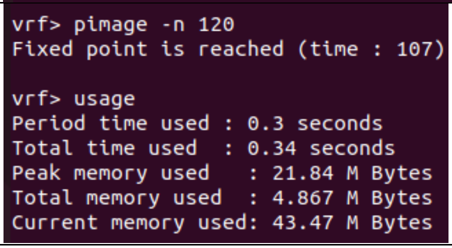
We can see that in four case, our implemented programs outperforms the ref program on the time used.

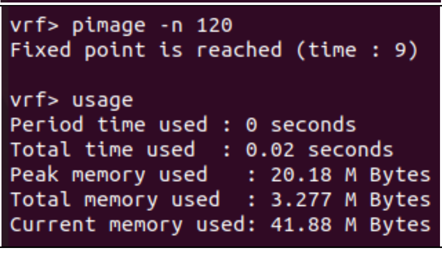
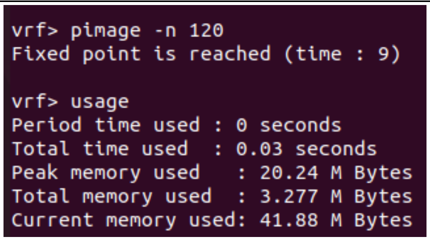
(5) Advanced technique

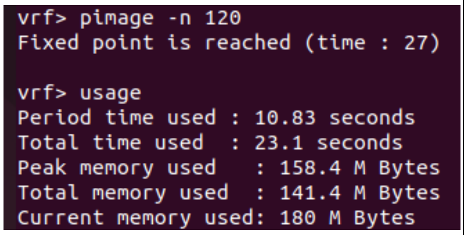
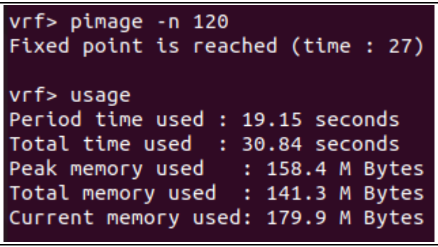
In the program, we implemented the restrict() function taught in lecture and utilized this in pimage command to achieve better efficiency. Below are the comparison in time for the program with and without restrict() implemented.

	with restrict	Without restrict

a.v	 `vrf> pimage -n 20`
	`Fixed point is reached (time : 12)`
	
	`vrf> usage`
	`Period time used : 0.63 seconds`
	`Total time used  : 1.74 seconds`
	`Peak memory used   : 50.23 M Bytes`
	`Total memory used  : 33.47 M Bytes`
	`Current memory used: 72.08 M Bytes`
		 `vrf> pimage -n 20`
		`Fixed point is reached (time : 12)`
		
		`vrf> usage`
		`Period time used : 0.75 seconds`
		`Total time used  : 1.88 seconds`
		`Peak memory used   : 48.98 M Bytes`
		`Total memory used  : 33.47 M Bytes`
		`Current memory used: 72.07 M Bytes`

b.v	 `vrf> pimage -n 120`
	`Fixed point is reached (time : 107)`
	
	`vrf> usage`
	`Period time used : 0.2 seconds`
	`Total time used  : 0.23 seconds`
	`Peak memory used   : 21.61 M Bytes`
	`Total memory used  : 4.867 M Bytes`
	`Current memory used: 43.47 M Bytes`
		 `vrf> pimage -n 120`
		`Fixed point is reached (time : 107)`
		
		`vrf> usage`
		`Period time used : 0.3 seconds`
		`Total time used  : 0.34 seconds`
		`Peak memory used   : 21.84 M Bytes`
		`Total memory used  : 4.867 M Bytes`
		`Current memory used: 43.47 M Bytes`

c.v	 `vrf> pimage -n 120`
	`Fixed point is reached (time : 9)`
	
	`vrf> usage`
	`Period time used : 0 seconds`
	`Total time used  : 0.02 seconds`
	`Peak memory used   : 20.18 M Bytes`
	`Total memory used  : 3.277 M Bytes`
	`Current memory used: 41.88 M Bytes`
		 `vrf> pimage -n 120`
		`Fixed point is reached (time : 9)`
		
		`vrf> usage`
		`Period time used : 0 seconds`
		`Total time used  : 0.03 seconds`
		`Peak memory used   : 20.24 M Bytes`
		`Total memory used  : 3.277 M Bytes`
		`Current memory used: 41.88 M Bytes`

vending -abs.v	 `vrf> pimage -n 120`
	`Fixed point is reached (time : 27)`
	
	`vrf> usage`
	`Period time used : 10.83 seconds`
	`Total time used  : 23.1 seconds`
	`Peak memory used   : 158.4 M Bytes`
	`Total memory used  : 141.4 M Bytes`
	`Current memory used: 180 M Bytes`
		 `vrf> pimage -n 120`
		`Fixed point is reached (time : 27)`
		
		`vrf> usage`
		`Period time used : 19.15 seconds`
		`Total time used  : 30.84 seconds`
		`Peak memory used   : 158.4 M Bytes`
		`Total memory used  : 141.3 M Bytes`
		`Current memory used: 179.9 M Bytes`

We can see that in four designs, the time used all decreased, especially in the large design like vending-abs.v. But it’s still not enough to make the program test vending.v properly.