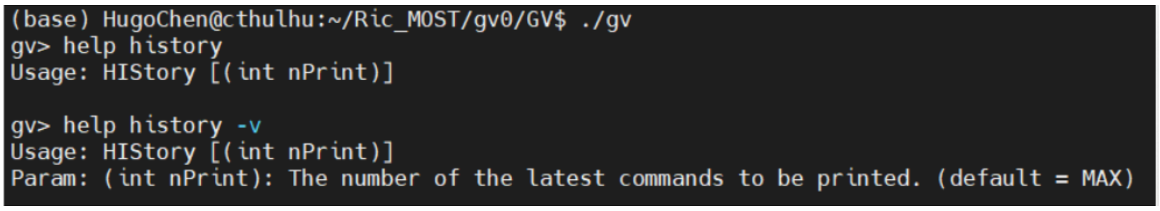
—---

GV Help

File: GV/src/cmd/gvCmdComm.cpp (.h)
Command: HELp [<(string cmd) [-Verbose]> | -Revealed]

Example:



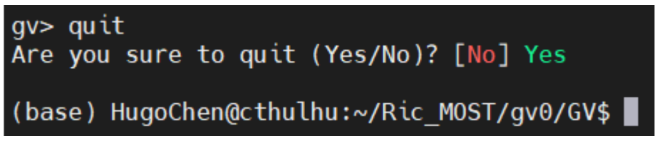
```
(base) HugoChen@cthulhu:~/Ric_MOST/gv0/GV$ ./gv
gv> help history
Usage: HIStory [(int nPrint)]

gv> help history -v
Usage: HIStory [(int nPrint)]
Param: (int nPrint): The number of the latest commands to be printed. (default = MAX)
```

—---

GV Quit

File: GV/src/cmd/gvCmdComm.cpp (.h)
Command: Quit [-Force]

Example:



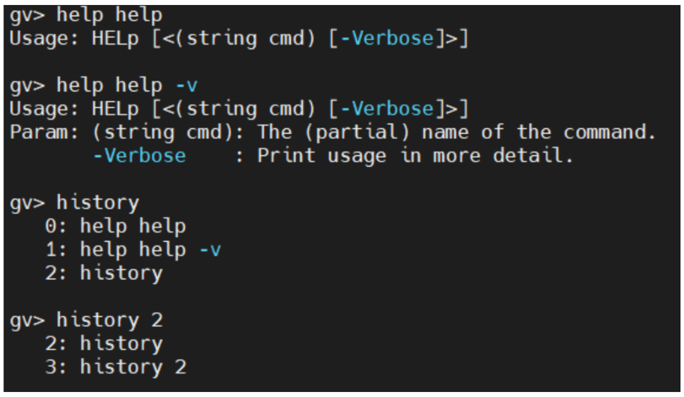
```
gv> quit
Are you sure to quit (Yes/No)? [No] Yes

(base) HugoChen@cthulhu:~/Ric_MOST/gv0/GV$
```

—---

GV History

File: GV/src/cmd/gvCmdComm.cpp (.h)
Command: HIStory [(int nPrint)]

Example:



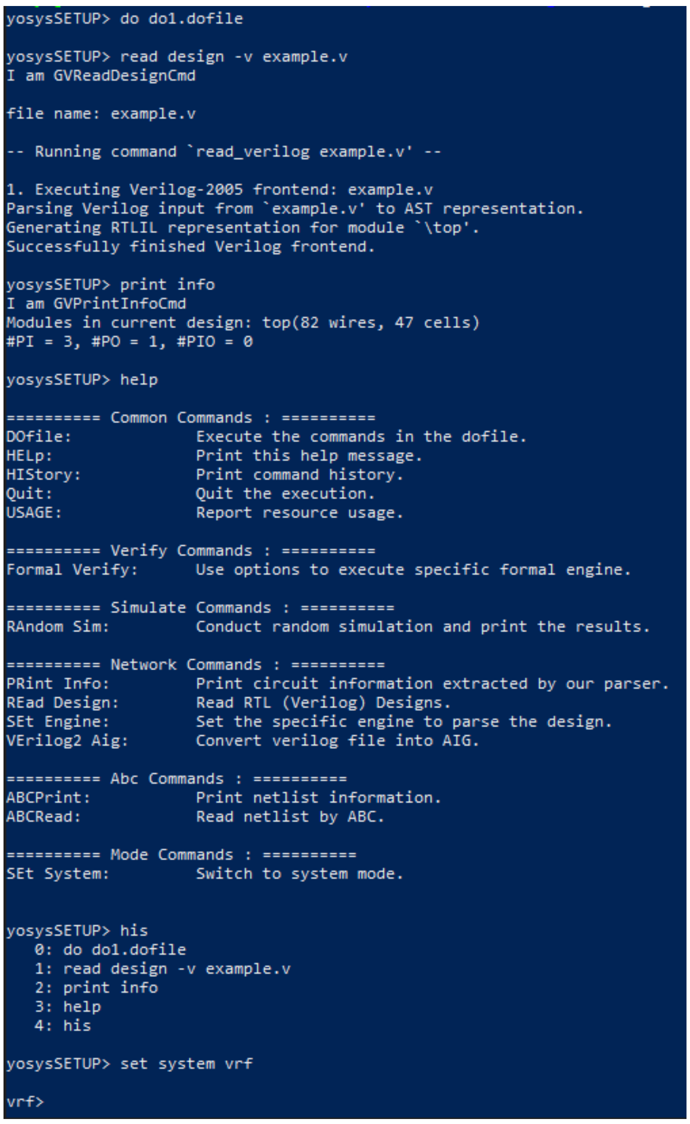
```
gv> help help
Usage: HELp [<(string cmd) [-Verbose]>]

gv> help help -v
Usage: HELp [<(string cmd) [-Verbose]>]
Param: (string cmd): The (partial) name of the command.
       -Verbose     : Print usage in more detail.

gv> history
   0: help help
   1: help help -v
   2: history

gv> history 2
   2: history
   3: history 2
```

—---

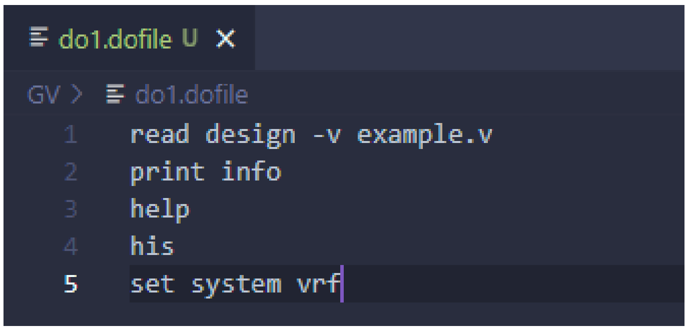
GV Do file

File: GV/src/cmd/gvCmdComm.cpp (.h)
Command: DOfile <(string fileName)>

Example:



```
yosysSETUP> do do1.dofile

yosysSETUP> read design -v example.v
I am GVReadDesignCmd

file name: example.v

-- Running command `read_verilog example.v' --

1. Executing Verilog-2005 frontend: example.v
Parsing Verilog input from `example.v' to AST representation.
Generating RTLIL representation for module `\top'.
Successfully finished Verilog frontend.

yosysSETUP> print info
I am GVPrintInfoCmd
Modules in current design: top(82 wires, 47 cells)
#PI = 3, #PO = 1, #PIO = 0

yosysSETUP> help

========== Common Commands : ==========
DOfile:             Execute the commands in the dofile.
HELp:               Print this help message.
HIStory:            Print command history.
Quit:               Quit the execution.
USAGE:              Report resource usage.

========== Verify Commands : ==========
Formal Verify:      Use options to execute specific formal engine.

========== Simulate Commands : ==========
RAndom Sim:         Conduct random simulation and print the results.

========== Network Commands : ==========
PRint Info:         Print circuit information extracted by our parser.
REad Design:        Read RTL (Verilog) Designs.
SEt Engine:         Set the specific engine to parse the design.
VErilog2 Aig:       Convert verilog file into AIG.

========== Abc Commands : ==========
ABCPrint:           Print netlist information.
ABCRead:            Read netlist by ABC.

========== Mode Commands : ==========
SEt System:         Switch to system mode.


yosysSETUP> his
   0: do do1.dofile
   1: read design -v example.v
   2: print info
   3: help
   4: his

yosysSETUP> set system vrf

vrf>
```

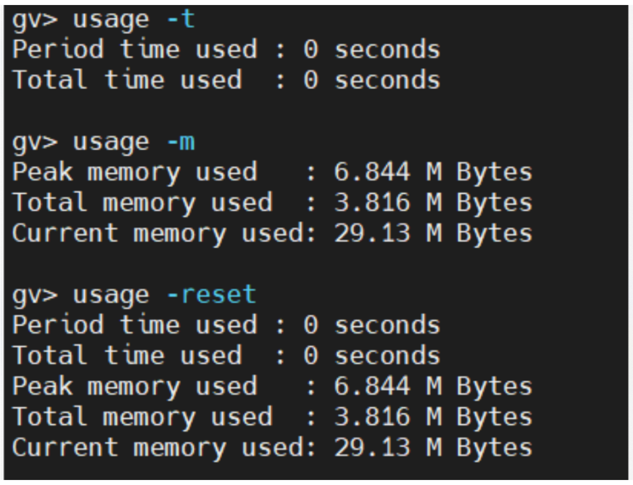


```
read design -v example.v
print info
help
his
set system vrf
```

GV Usage

File: GV/src/cmd/gvCmdComm.cpp (.h)
Command: USAGE [-Time-only | -Memory-only] [-RESET]

Example:



```
gv> usage -t
Period time used : 0 seconds
Total time used  : 0 seconds

gv> usage -m
Peak memory used    : 6.844 M Bytes
Total memory used   : 3.816 M Bytes
Current memory used: 29.13 M Bytes

gv> usage -reset
Period time used : 0 seconds
Total time used  : 0 seconds
Peak memory used    : 6.844 M Bytes
Total memory used   : 3.816 M Bytes
Current memory used: 29.13 M Bytes
```

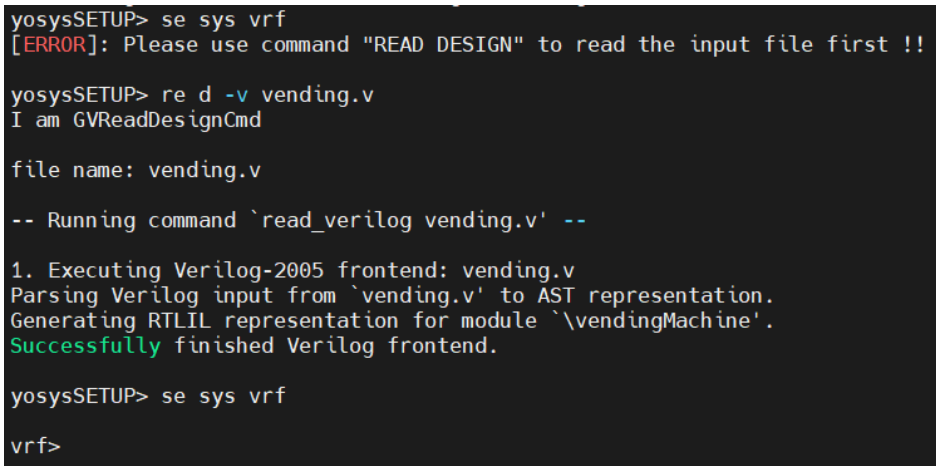
—---

GV SEt System

File: GV/src/mod/gvModCmd.cpp (.h)
Command: SEt SYStem <(string modeName)>

- note: <(string modeName)> should be “vrf" or "setup".
- note: You should run command “REad Design” to read the input file first!
- note: You can only run commands “Formal Verify” & “RAndom Simulation” in vrf mode.

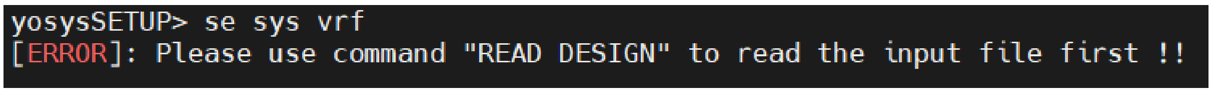
Example:



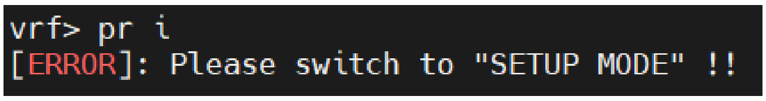
```
yosysSETUP> se sys vrf
[ERROR]: Please use command "READ DESIGN" to read the input file first !!

yosysSETUP> re d -v vending.v
I am GVReadDesignCmd

file name: vending.v

-- Running command `read_verilog vending.v' --

1. Executing Verilog-2005 frontend: vending.v
Parsing Verilog input from `vending.v' to AST representation.
Generating RTLIL representation for module `\vendingMachine'.
Successfully finished Verilog frontend.

yosysSETUP> se sys vrf

vrf>
```

Error Handling:

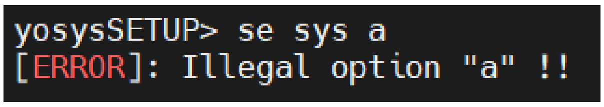
1. You should read the design first.


```
yosysSETUP> se sys vrf
[ERROR]: Please use command "READ DESIGN" to read the input file first !!
```

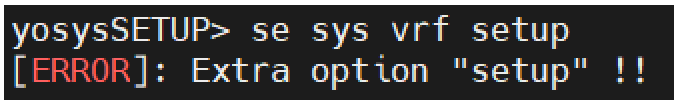
2. You need to switch to setup mode.


```
vrf> pr i
[ERROR]: Please switch to "SETUP MODE" !!
```

3. Illegal <(string modeName)>


```
yosysSETUP> se sys a
[ERROR]: Illegal option "a" !!
```

4. Extra <(string modeName)>.


```
yosysSETUP> se sys vrf setup
[ERROR]: Extra option "setup" !!
```

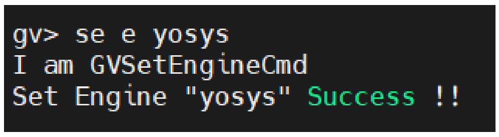
—--

GV Set Engine

File: GV/src/ntk/gvNtkCmd.cpp (.h)
Command: SEt Engine <(string engineName)>

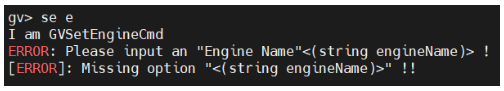
- note: <(string engineName)> should be yosys, abc or V3.

Example:

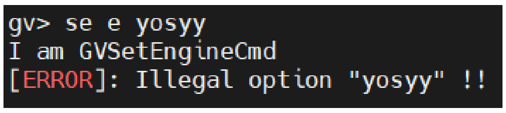

```
gv> se e yosys
I am GVSetEngineCmd
Set Engine "yosys" Success !!
```

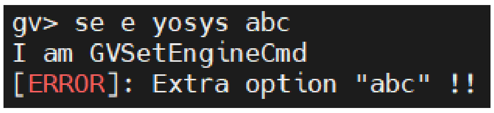
Error Handling:

1. <(string engineName)> needs to exist.


```
gv> se e
I am GVSetEngineCmd
ERROR: Please input an "Engine Name"<(string engineName)> !
[ERROR]: Missing option "<(string engineName)>" !!
```

2. <(string engineName)> should only be one of the yosys, abc or V3.


```
gv> se e yosyy
I am GVSetEngineCmd
[ERROR]: Illegal option "yosyy" !!
```


```
gv> se e yosys abc
I am GVSetEngineCmd
[ERROR]: Extra option "abc" !!
```

—--

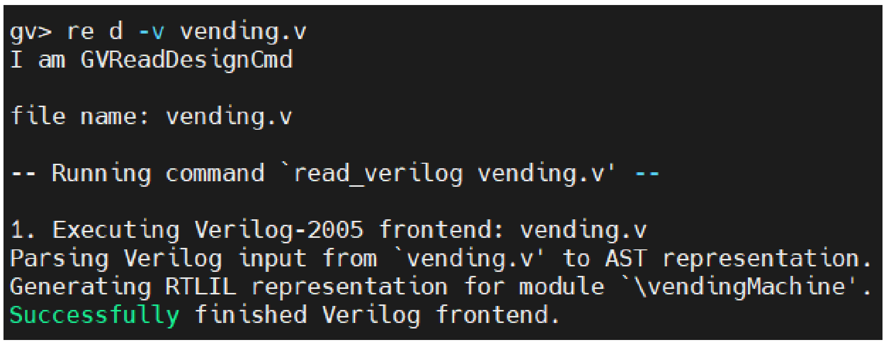
GV Read Design

File: GV/src/ntk/gvNtkCmd.cpp (.h)
Command: REad Design <-Verilog | -Blif | -Aig> <(string fileName)>

- note: For now, we use yosys as our default engine.
- note: engine → supported format currently

- yosysSETUP → verilog, blif
- abcSETUP → verilog, blif, aig
- v3SETUP → verilog, aig

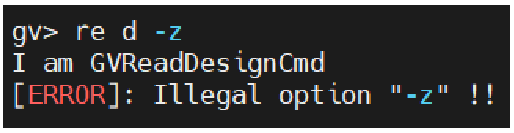
Example:



```
gv> re d -v vending.v
I am GVReadDesignCmd

file name: vending.v

-- Running command `read_verilog vending.v' --

1. Executing Verilog-2005 frontend: vending.v
Parsing Verilog input from `vending.v' to AST representation.
Generating RTLIL representation for module `\vendingMachine'.
Successfully finished Verilog frontend.
```

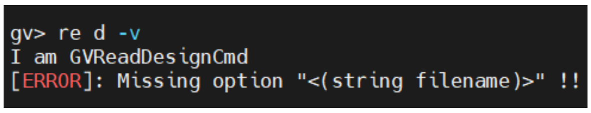
Error Handling:

3. Illegal option.



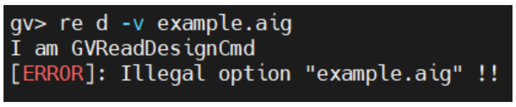
```
gv> re d -z
I am GVReadDesignCmd
[ERROR]: Illegal option "-z" !!
```

4. <(string fileName)> needs to exist.



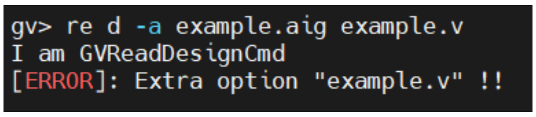
```
gv> re d -v
I am GVReadDesignCmd
[ERROR]: Missing option "<(string filename)>" !!
```

5. The User specifies input file format should match the input file extension.



```
gv> re d -v example.aig
I am GVReadDesignCmd
[ERROR]: Illegal option "example.aig" !!
```

6. Extra input file name.



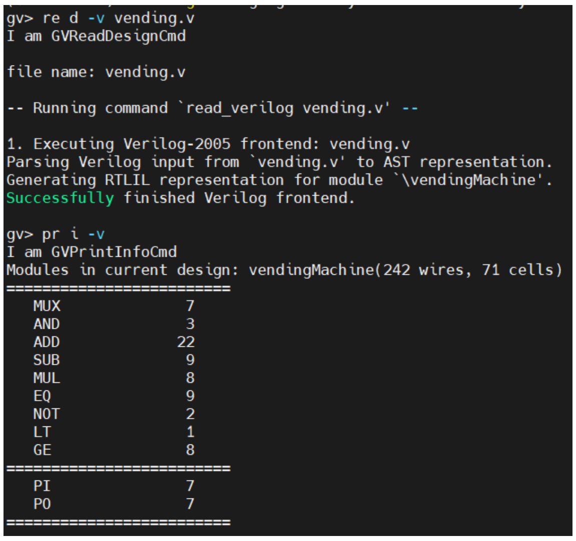
```
gv> re d -a example.aig example.v
I am GVReadDesignCmd
[ERROR]: Extra option "example.v" !!
```

—--

GV Print Info

File: GV/src/ntk/gvNtkCmd.cpp (.h)
Command: PRint Info [-Verbose]

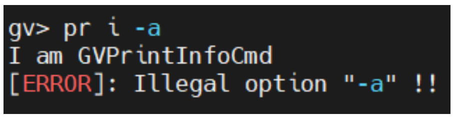
Example:



```
gv> re d -v vending.v
I am GVReadDesignCmd

file name: vending.v

-- Running command `read_verilog vending.v' --

1. Executing Verilog-2005 frontend: vending.v
Parsing Verilog input from `vending.v' to AST representation.
Generating RTLIL representation for module `\vendingMachine'.
Successfully finished Verilog frontend.

gv> pr i -v
I am GVPrintInfoCmd
Modules in current design: vendingMachine(242 wires, 71 cells)
=========================
   MUX              7
   AND              3
   ADD             22
   SUB              9
   MUL              8
   EQ               9
   NOT              2
   LT               1
   GE               8
=========================
   PI               7
   PO               7
=========================
```

Error Handling:

1. Illegal option.



```
gv> pr i -a
I am GVPrintInfoCmd
[ERROR]: Illegal option "-a" !!
```

—---

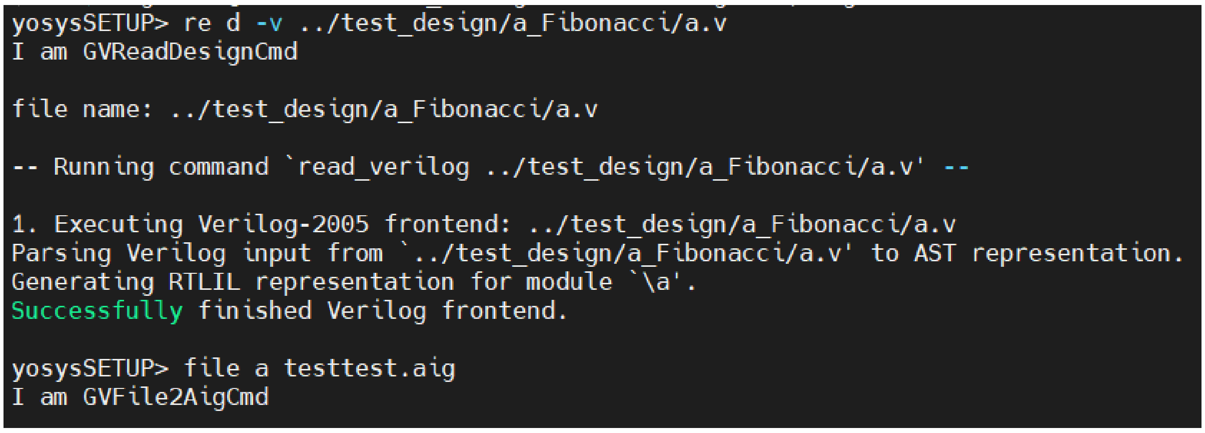
GV File 2 Aig

File: GV/src/ntk/gvNtkCmd.cpp (.h)

Command: File2 Aig <output_filename.aig>

- note: a user needs to "read design" first to get the <input_filename>
- note: <input_file> should be Verilog file now (but further we can let use specify the option like "-v", "-sv", etc. to read in different file format)
- note: <output_file> should be AIG

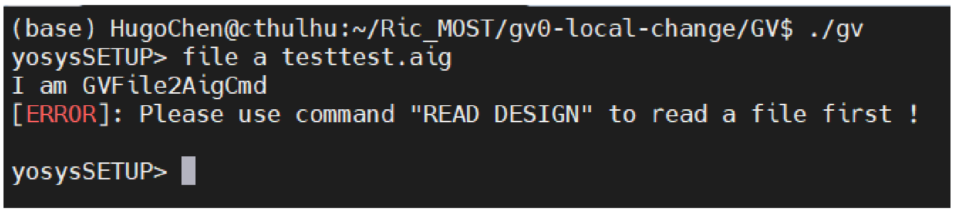
Example:



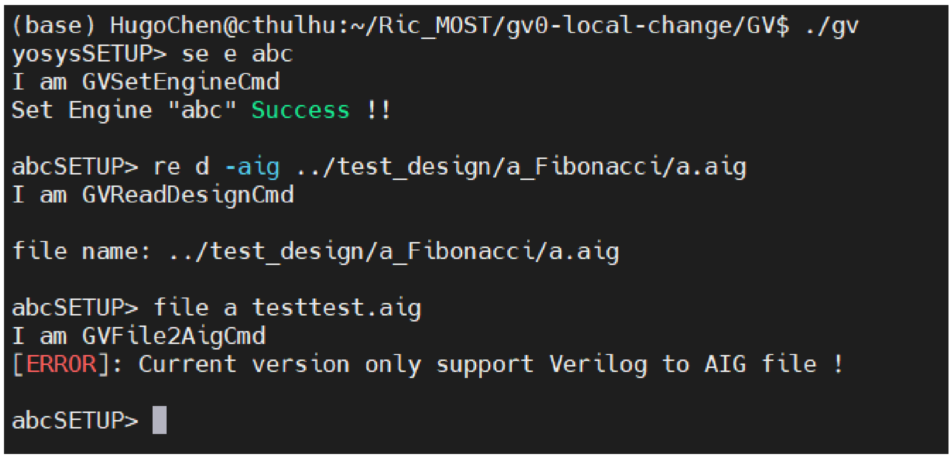
```
yosysSETUP> re d -v ../test_design/a_Fibonacci/a.v
I am GVReadDesignCmd

file name: ../test_design/a_Fibonacci/a.v

-- Running command `read_verilog ../test_design/a_Fibonacci/a.v' --

1. Executing Verilog-2005 frontend: ../test_design/a_Fibonacci/a.v
Parsing Verilog input from `../test_design/a_Fibonacci/a.v' to AST representation.
Generating RTLIL representation for module `\a'.
Successfully finished Verilog frontend.

yosysSETUP> file a testtest.aig
I am GVFile2AigCmd
```

Error Handling:

1. The input file requires to exist, and we only support Verilog2Aig now

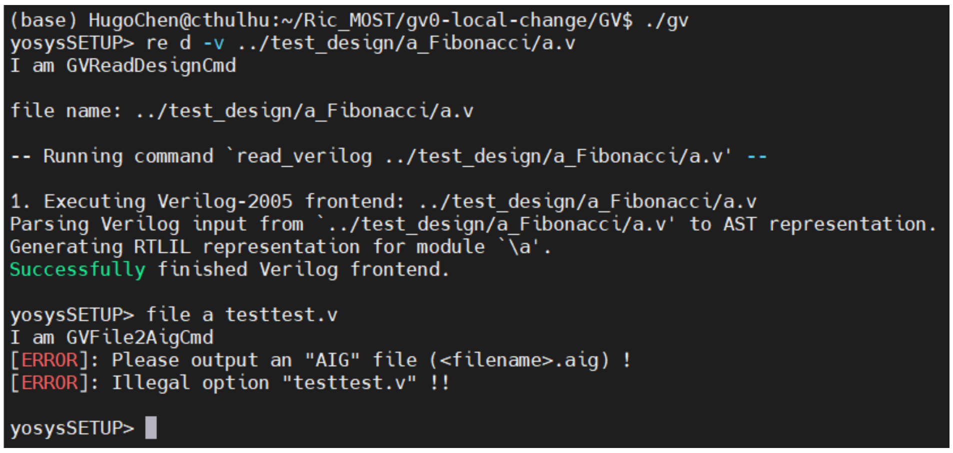


```
(base) HugoChen@cthulhu:~/Ric_MOST/gv0-local-change/GV$ ./gv
yosysSETUP> file a testtest.aig
I am GVFile2AigCmd
[ERROR]: Please use command "READ DESIGN" to read a file first !

yosysSETUP>
```



```
(base) HugoChen@cthulhu:~/Ric_MOST/gv0-local-change/GV$ ./gv
yosysSETUP> se e abc
I am GVSetEngineCmd
Set Engine "abc" Success !!

abcSETUP> re d -aig ../test_design/a_Fibonacci/a.aig
I am GVReadDesignCmd

file name: ../test_design/a_Fibonacci/a.aig

abcSETUP> file a testtest.aig
I am GVFile2AigCmd
[ERROR]: Current version only support Verilog to AIG file !

abcSETUP>
```

2. The output file needs to be AIG



```
(base) HugoChen@cthulhu:~/Ric_MOST/gv0-local-change/GV$ ./gv
yosysSETUP> re d -v ../test_design/a_Fibonacci/a.v
I am GVReadDesignCmd

file name: ../test_design/a_Fibonacci/a.v

-- Running command `read_verilog ../test_design/a_Fibonacci/a.v' --

1. Executing Verilog-2005 frontend: ../test_design/a_Fibonacci/a.v
Parsing Verilog input from `../test_design/a_Fibonacci/a.v' to AST representation.
Generating RTLIL representation for module `\a'.
Successfully finished Verilog frontend.

yosysSETUP> file a testtest.v
I am GVFile2AigCmd
[ERROR]: Please output an "AIG" file (<filename>.aig) !
[ERROR]: Illegal option "testtest.v" !!

yosysSETUP>
```

—---

GV Formal Verify

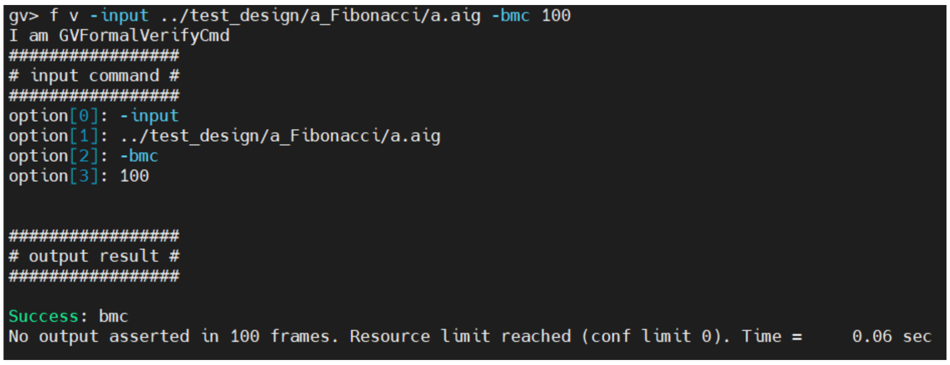
File: GV/src/vrf/gvVrfCmd.cpp (.h)

Command: Formal Verify [-bmc <int_depth> | -pdr | -itp]

- note: File should be AIG (gvVerilog2Aig first / read_design -aig)
- note: <int_depth> should be an integer (i.e. time frame)

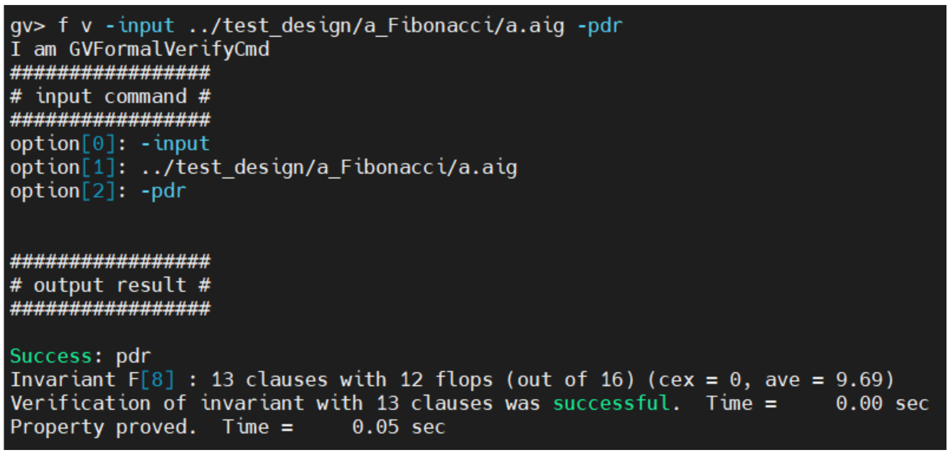
Example:

bmc



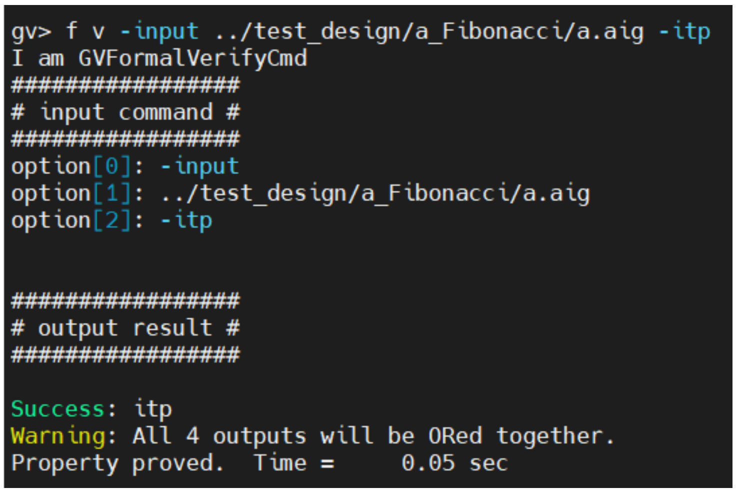
```
gv> f v -input ../test_design/a_Fibonacci/a.aig -bmc 100
I am GVFormalVerifyCmd
#################
# input command #
#################
option[0]: -input
option[1]: ../test_design/a_Fibonacci/a.aig
option[2]: -bmc
option[3]: 100


#################
# output result #
#################

Success: bmc
No output asserted in 100 frames. Resource limit reached (conf limit 0). Time =     0.06 sec
```

pdr



```
gv> f v -input ../test_design/a_Fibonacci/a.aig -pdr
I am GVFormalVerifyCmd
#################
# input command #
#################
option[0]: -input
option[1]: ../test_design/a_Fibonacci/a.aig
option[2]: -pdr


#################
# output result #
#################

Success: pdr
Invariant F[8] : 13 clauses with 12 flops (out of 16) (cex = 0, ave = 9.69)
Verification of invariant with 13 clauses was successful.  Time =     0.00 sec
Property proved.  Time =     0.05 sec
```

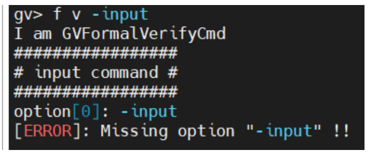
itp



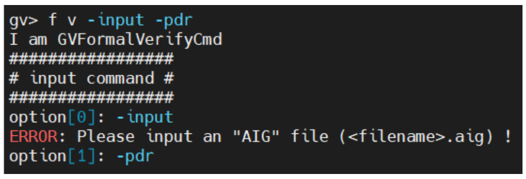
```
gv> f v -input ../test_design/a_Fibonacci/a.aig -itp
I am GVFormalVerifyCmd
#################
# input command #
#################
option[0]: -input
option[1]: ../test_design/a_Fibonacci/a.aig
option[2]: -itp


#################
# output result #
#################

Success: itp
Warning: All 4 outputs will be ORed together.
Property proved.  Time =     0.05 sec
```

Error Handling:

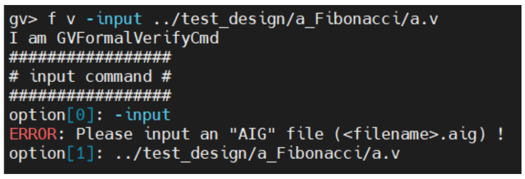
1. <filename> needs to exist, and should be an AIG file (now)



```
gv> f v -input
I am GVFormalVerifyCmd
#################
# input command #
#################
option[0]: -input
[ERROR]: Missing option "-input" !!
```

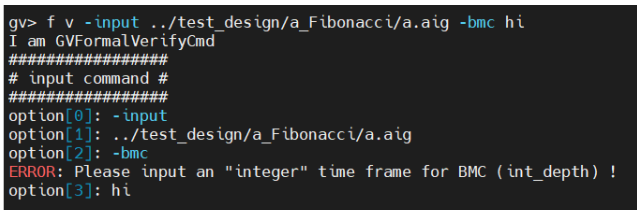


```
gv> f v -input -pdr
I am GVFormalVerifyCmd
################
# input command #
################
option[0]: -input
ERROR: Please input an "AIG" file (<filename>.aig) !
option[1]: -pdr
```



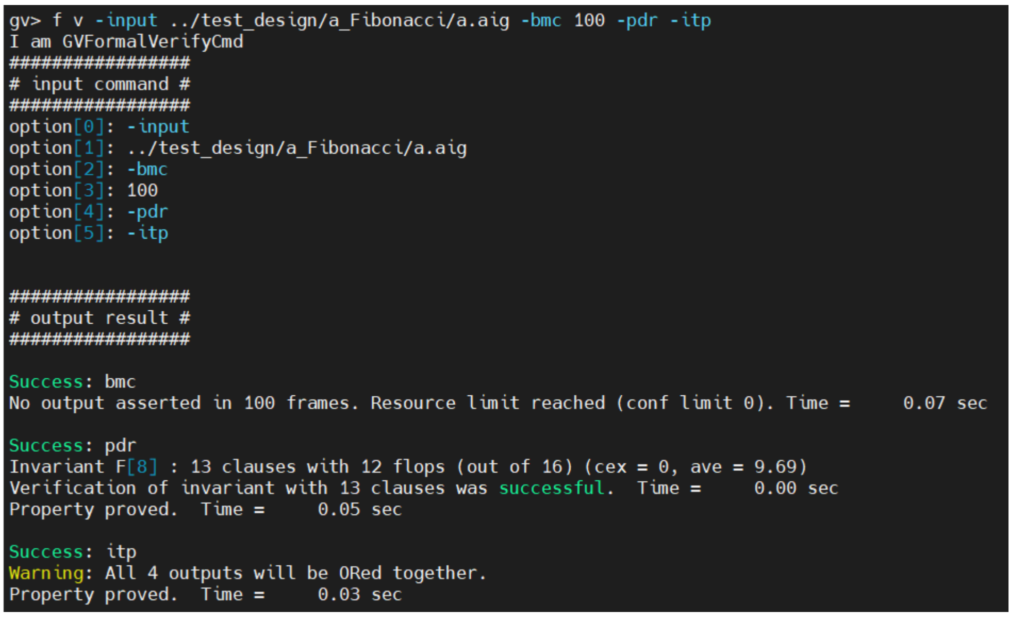
```
gv> f v -input ../test_design/a_Fibonacci/a.v
I am GVFormalVerifyCmd
################
# input command #
################
option[0]: -input
ERROR: Please input an "AIG" file (<filename>.aig) !
option[1]: ../test_design/a_Fibonacci/a.v
```

2. <int_depth> should be an integer



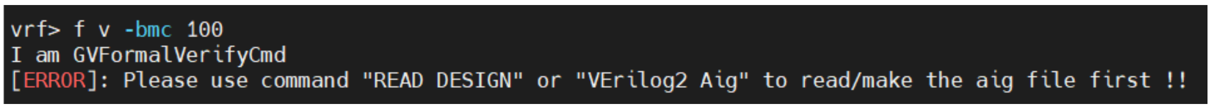
```
gv> f v -input ../test_design/a_Fibonacci/a.aig -bmc hi
I am GVFormalVerifyCmd
################
# input command #
################
option[0]: -input
option[1]: ../test_design/a_Fibonacci/a.aig
option[2]: -bmc
ERROR: Please input an "integer" time frame for BMC (int_depth) !
option[3]: hi
```

3. if specify multiple formal engine, then we will execute all



```
gv> f v -input ../test_design/a_Fibonacci/a.aig -bmc 100 -pdr -itp
I am GVFormalVerifyCmd
################
# input command #
################
option[0]: -input
option[1]: ../test_design/a_Fibonacci/a.aig
option[2]: -bmc
option[3]: 100
option[4]: -pdr
option[5]: -itp


################
# output result #
################

Success: bmc
No output asserted in 100 frames. Resource limit reached (conf limit 0). Time =     0.07 sec

Success: pdr
Invariant F[8] : 13 clauses with 12 flops (out of 16) (cex = 0, ave = 9.69)
Verification of invariant with 13 clauses was successful.  Time =     0.00 sec
Property proved.  Time =     0.05 sec

Success: itp
Warning: All 4 outputs will be ORed together.
Property proved.  Time =     0.03 sec
```

4. A user needs to read in **(GV Read Design)** or convert **(GV File2Aig)** the file to AIG format before doing formal verification



```
vrf> f v -bmc 100
I am GVFormalVerifyCmd
[ERROR]: Please use command "READ DESIGN" or "VErilog2 Aig" to read/make the aig file first !!
```

—---

GV randomSim

File: yosys/ext-sim/sim.cc
Command : randomSim [options]

note: you should enter vrf mode first
note: random simulation can run on yosys-supported files (Verilog, …)
note: the parser engine must be “**yosys**”

<-input>
The name of the verilog file you want to simulate

[-clk]
The clock name of the verilog file you want to simulate
Default to be "clk"

[-rst]
The reset name of the verilog file you want to simulate
Default to be "reset"
reset when reset = 1

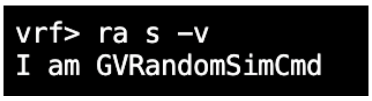
[-rst_n]
The reset_n name of the verilog file you want to simulate
Default to be "reset_n"
reset when reset_n = 0
Only one of reset and reset_n should be set

[-output]
The name of output file which contains the simulation result

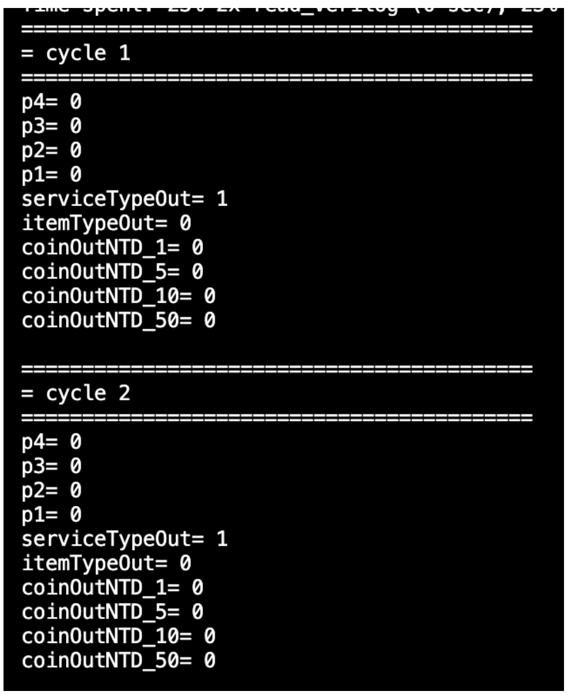
[-v]
verbose print the result of simulation on the command line

[-file]
The stimulus file name.
The format of the stimulus file should be unsigned int separated by blank for each row. Each row stands for a cycle, so each int corresponds to an input signal. The order of inputs is reversed compared to the declaration order in the verilog file. Please see the following example for reference.
Please also note that clk and rst cannot be assigned.

Example:



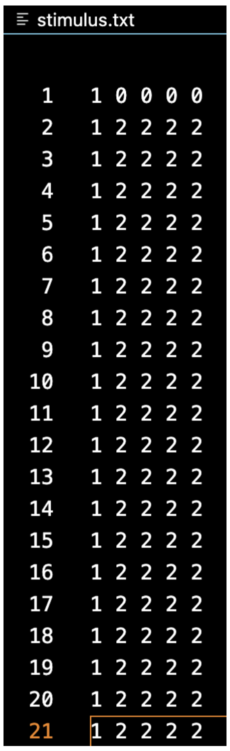
```
vrf> ra s -v
I am GVRandomSimCmd
```



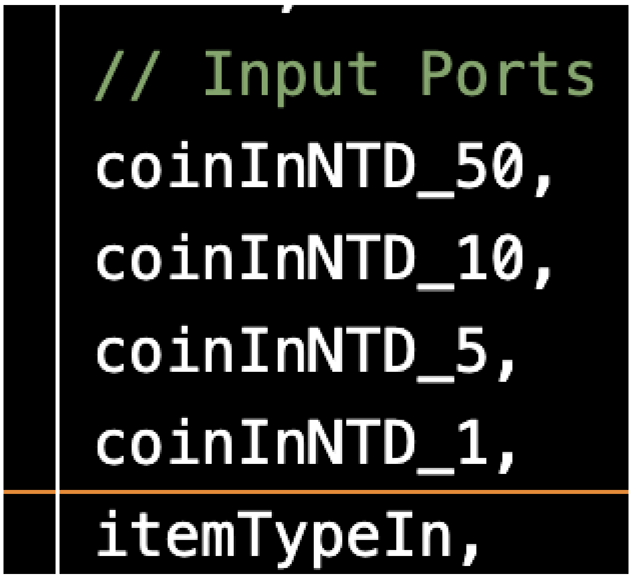
```
Time spent: 230 2x read_verilog (0 sec), 230
=========================================
= cycle 1
=========================================
p4= 0
p3= 0
p2= 0
p1= 0
serviceTypeOut= 1
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0

=========================================
= cycle 2
=========================================
p4= 0
p3= 0
p2= 0
p1= 0
serviceTypeOut= 1
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0
```

With stimulus :

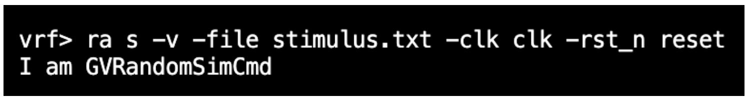
The ‘1’ in the first row corresponds to itemTypeIn, the first ‘0’ in row 1 corresponds to coinInNTD_1, and so on.



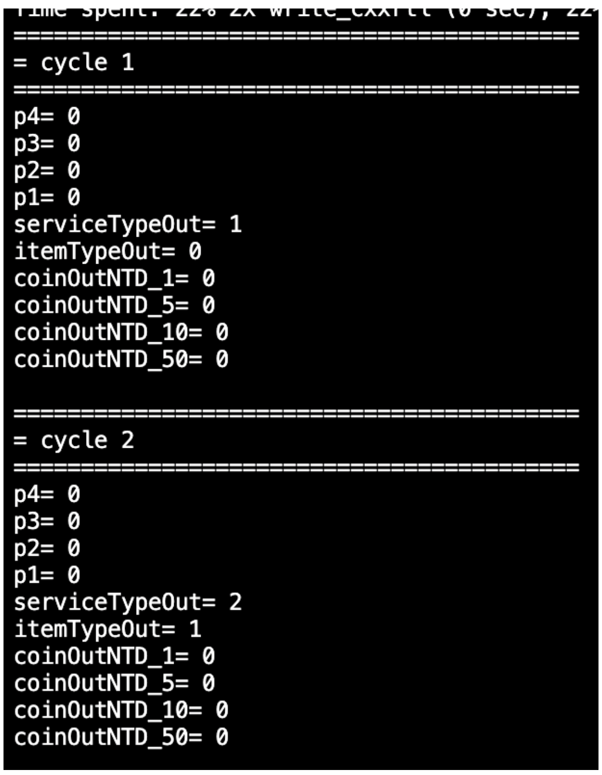
```
stimulus.txt

 1   1 0 0 0 0
 2   1 2 2 2 2
 3   1 2 2 2 2
 4   1 2 2 2 2
 5   1 2 2 2 2
 6   1 2 2 2 2
 7   1 2 2 2 2
 8   1 2 2 2 2
 9   1 2 2 2 2
10   1 2 2 2 2
11   1 2 2 2 2
12   1 2 2 2 2
13   1 2 2 2 2
14   1 2 2 2 2
15   1 2 2 2 2
16   1 2 2 2 2
17   1 2 2 2 2
18   1 2 2 2 2
19   1 2 2 2 2
20   1 2 2 2 2
21   1 2 2 2 2
```



```
// Input Ports
coinInNTD_50,
coinInNTD_10,
coinInNTD_5,
coinInNTD_1,
itemTypeIn,
```



```
vrf> ra s -v -file stimulus.txt -clk clk -rst_n reset
I am GVRandomSimCmd
```



```
==========================================
= cycle 1
==========================================
p4= 0
p3= 0
p2= 0
p1= 0
serviceTypeOut= 1
itemTypeOut= 0
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0

==========================================
= cycle 2
==========================================
p4= 0
p3= 0
p2= 0
p1= 0
serviceTypeOut= 2
itemTypeOut= 1
coinOutNTD_1= 0
coinOutNTD_5= 0
coinOutNTD_10= 0
coinOutNTD_50= 0
```