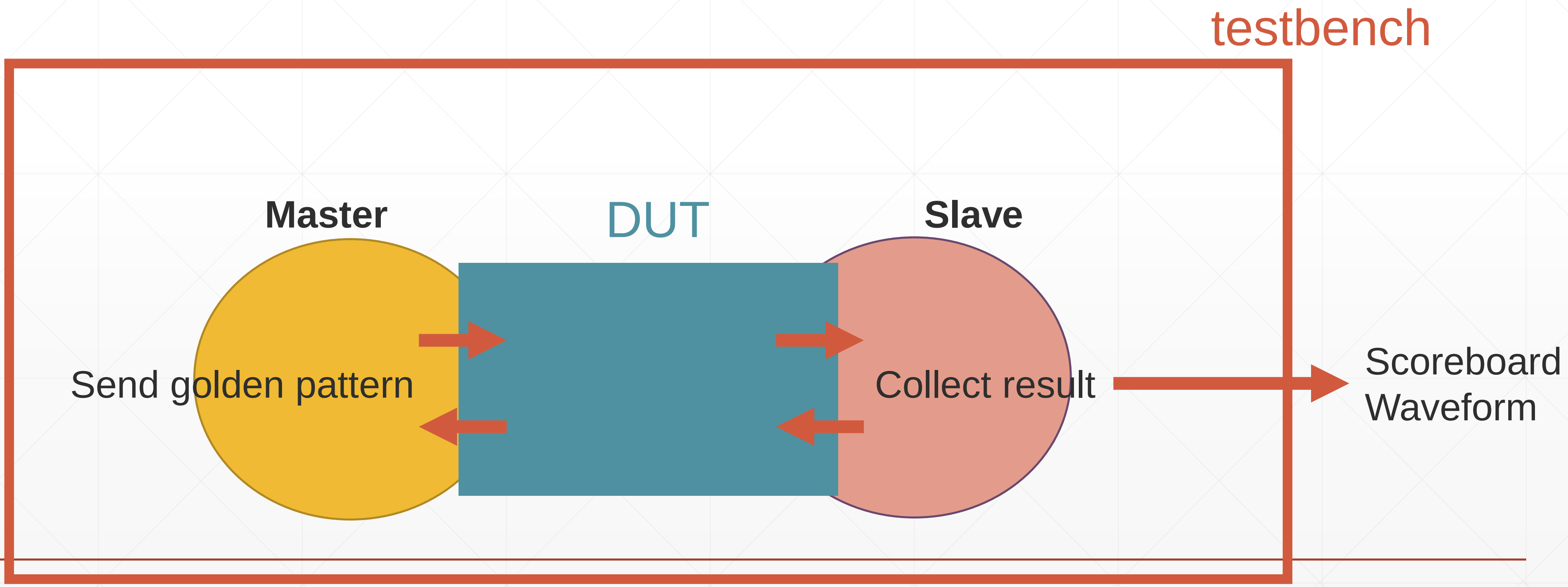
Nicotb Tutorial

謝汶璁
2021-08-03

Introduction

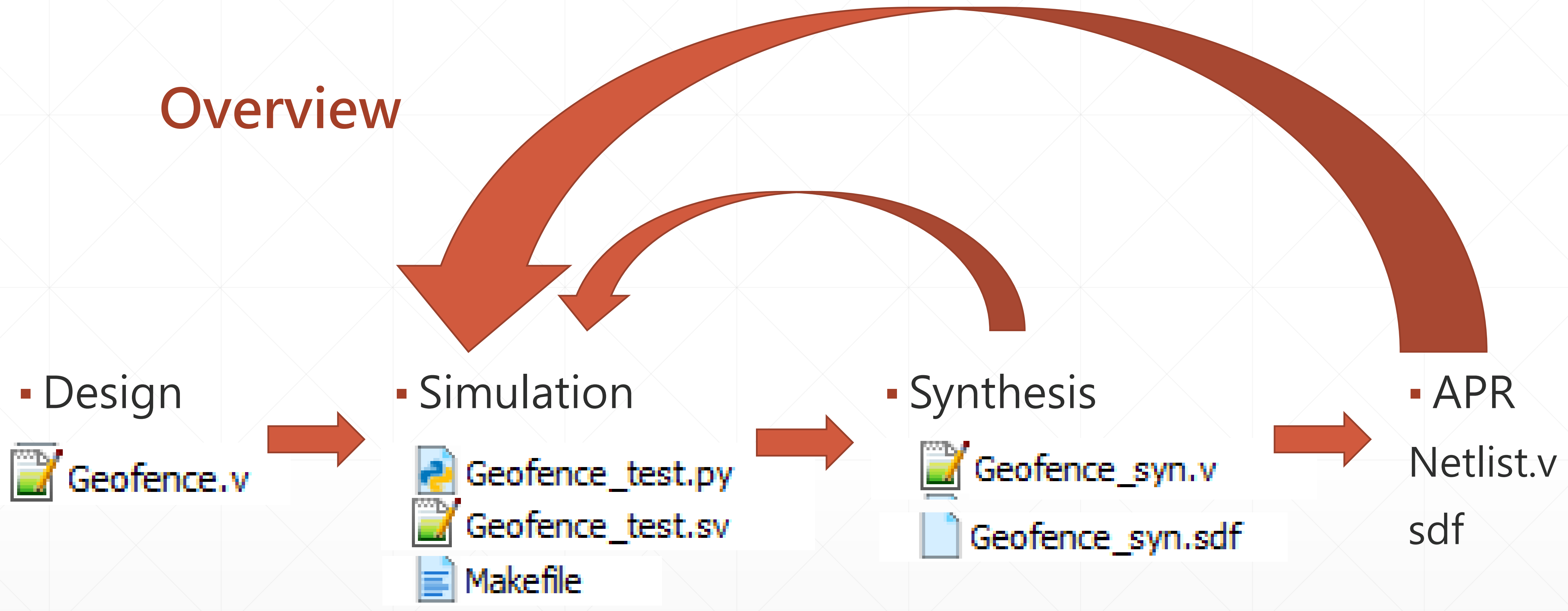
- Developed by **johnjohnlin.**
- Provide **Python-Verilog (ncverilog) co-simulation**



Advantages

- **Dynamic** golden pattern
- Have access to useful packages in Python (e.g. Numpy, Torch, matplotlib…)
- Used to verify **MERIT**

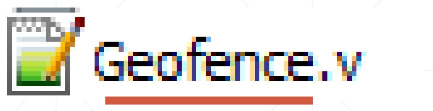
Overview



- Design
  - Geofence.v
- Simulation
  - Geofence_test.py
  - Geofence_test.sv
  - Makefile
- Synthesis
  - Geofence_syn.v
  - Geofence_syn.sdf
- APR
  - Netlist.v
  - sdf

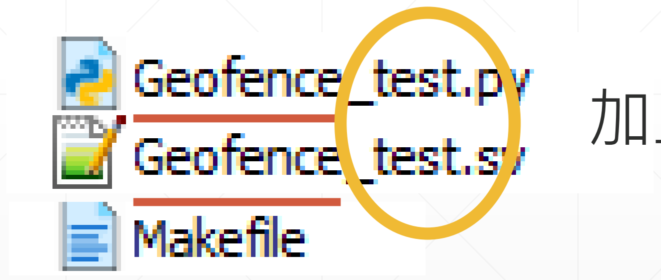
模擬前準備

- Design


Geofence.v

檔名必須一致

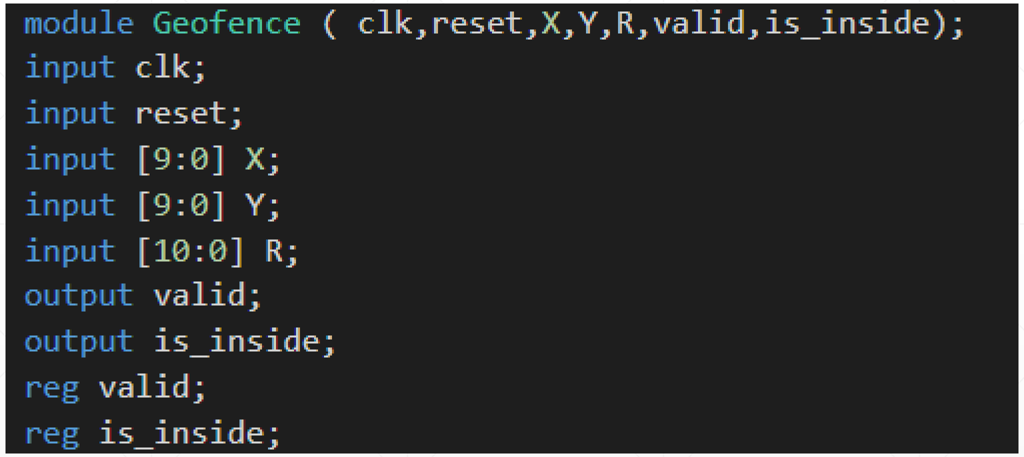
- Simulation


Geofence_test.py
Geofence_test.sv
Makefile

加上_test

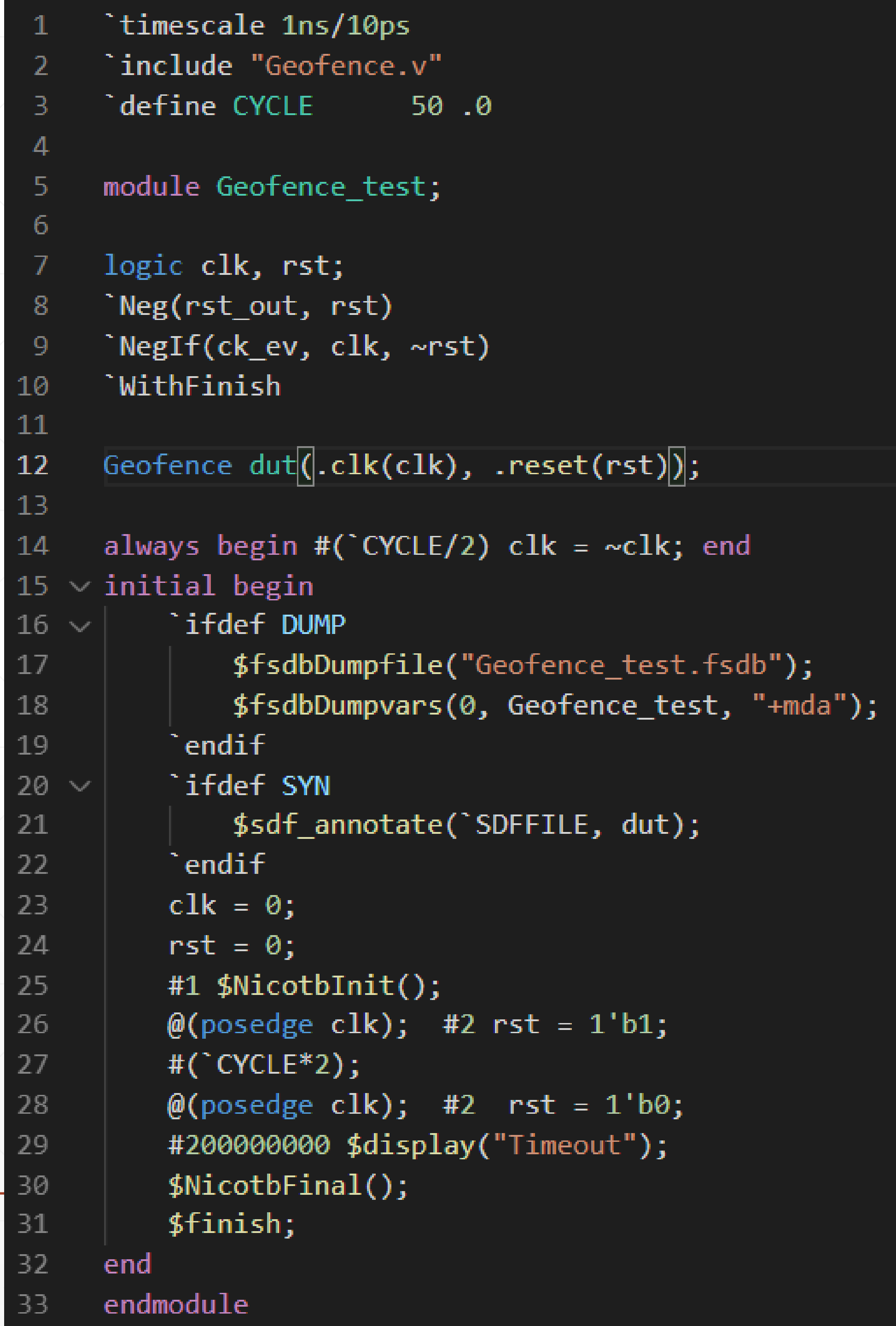
以上規則寫在makefile內

範例design


```
module Geofence ( clk,reset,X,Y,R,valid,is_inside);
input clk;
input reset;
input [9:0] X;
input [9:0] Y;
input [10:0] R;
output valid;
output is_inside;
reg valid;
reg is_inside;
```

_test.sv overview

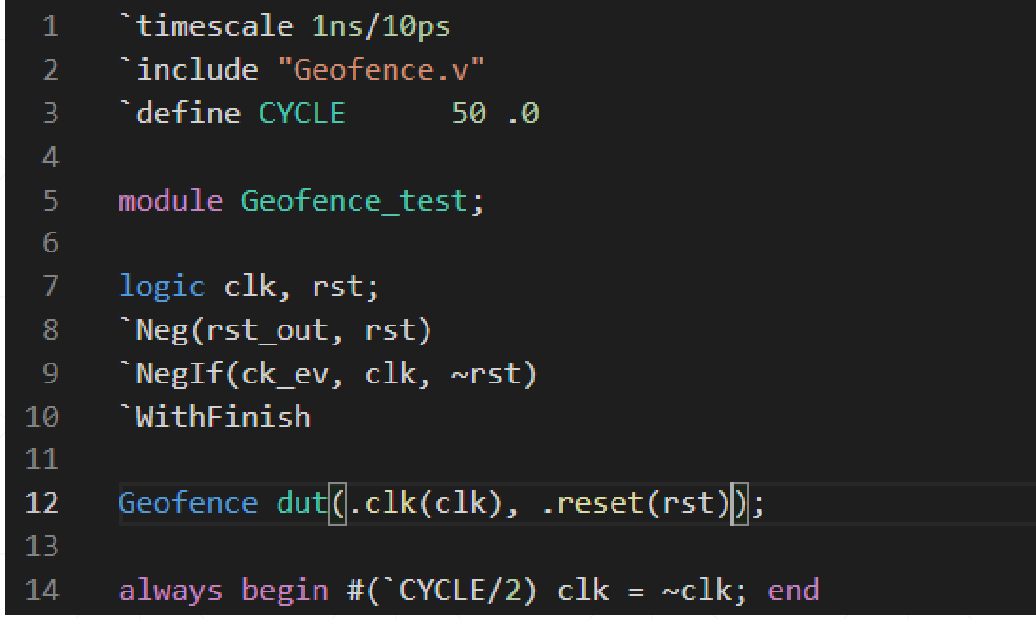
- 如一般的testbench
- 只需控制clock, reset
- dut的I/O在_test.py中控制



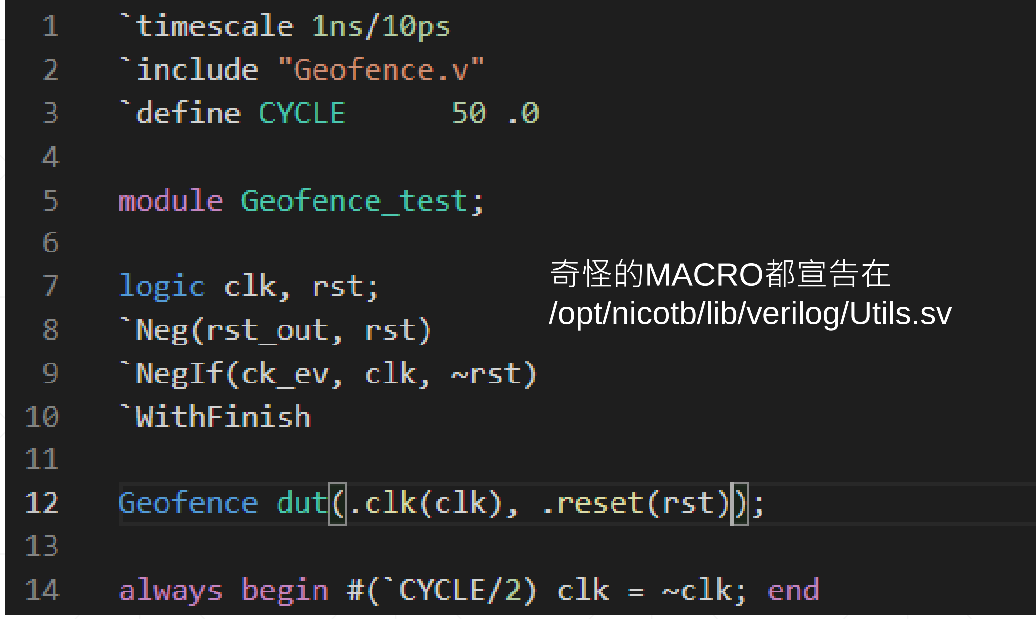
```
`timescale 1ns/10ps
`include "Geofence.v"
`define CYCLE      50 .0

module Geofence_test;

logic clk, rst;
`Neg(rst_out, rst)
`NegIf(ck_ev, clk, ~rst)
`WithFinish

Geofence dut(.clk(clk), .reset(rst));

always begin #(`CYCLE/2) clk = ~clk; end
initial begin
    `ifdef DUMP
        $fsdbDumpfile("Geofence_test.fsdb");
        $fsdbDumpvars(0, Geofence_test, "+mda");
    `endif
    `ifdef SYN
        $sdf_annotate(`SDFFILE, dut);
    `endif
    clk = 0;
    rst = 0;
    #1 $NicotbInit();
    @(posedge clk);  #2 rst = 1'b1;
    #(`CYCLE*2);
    @(posedge clk);  #2  rst = 1'b0;
    #200000000 $display("Timeout");
    $NicotbFinal();
    $finish;
end
endmodule
```

_test.sv-宣告

- Timescale
- Include design
- Clock cycle
- 宣告clock, reset訊號 (**其他IO在 _test.py**)
- 宣告dut
- **宣告Clock, Reset的Event**



```
`timescale 1ns/10ps
`include "Geofence.v"
`define CYCLE      50 .0

module Geofence_test;

logic clk, rst;
`Neg(rst_out, rst)
`NegIf(ck_ev, clk, ~rst)
`WithFinish

Geofence dut(.clk(clk), .reset(rst));

always begin #(`CYCLE/2) clk = ~clk; end
```



```
`timescale 1ns/10ps
`include "Geofence.v"
`define CYCLE      50 .0

module Geofence_test;

logic clk, rst;
`Neg(rst_out, rst)
`NegIf(ck_ev, clk, ~rst)
`WithFinish

Geofence dut(.clk(clk), .reset(rst));

always begin #(`CYCLE/2) clk = ~clk; end
```

奇怪的MACRO都宣告在
/opt/nicotb/lib/verilog/Utils.sv

_test.sv-宣告Event

在_test.py中會**等待event觸發**
需要是這一層的訊號才能宣告event

- `Pos / `Neg(event名稱, verilog訊號)

通常用來宣告**reset event**
在verilog訊號正/負緣時**觸發_test.py中的event**

- `PosIf / `NegIf(event名稱, verilog訊號, 條件)

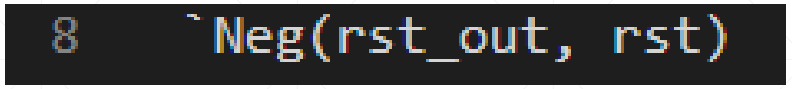
在verilog訊號正/負緣時檢查有無滿足條件再觸發event

- `WithFinish

照打

_test.sv-reset & clock event

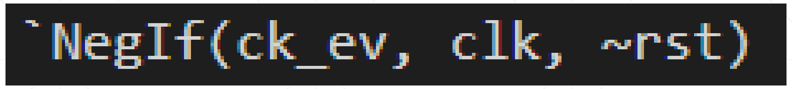
- reset event



```
`Neg(rst_out, rst)
```

Active high用`Pos
Active low用`Neg

- Clock event



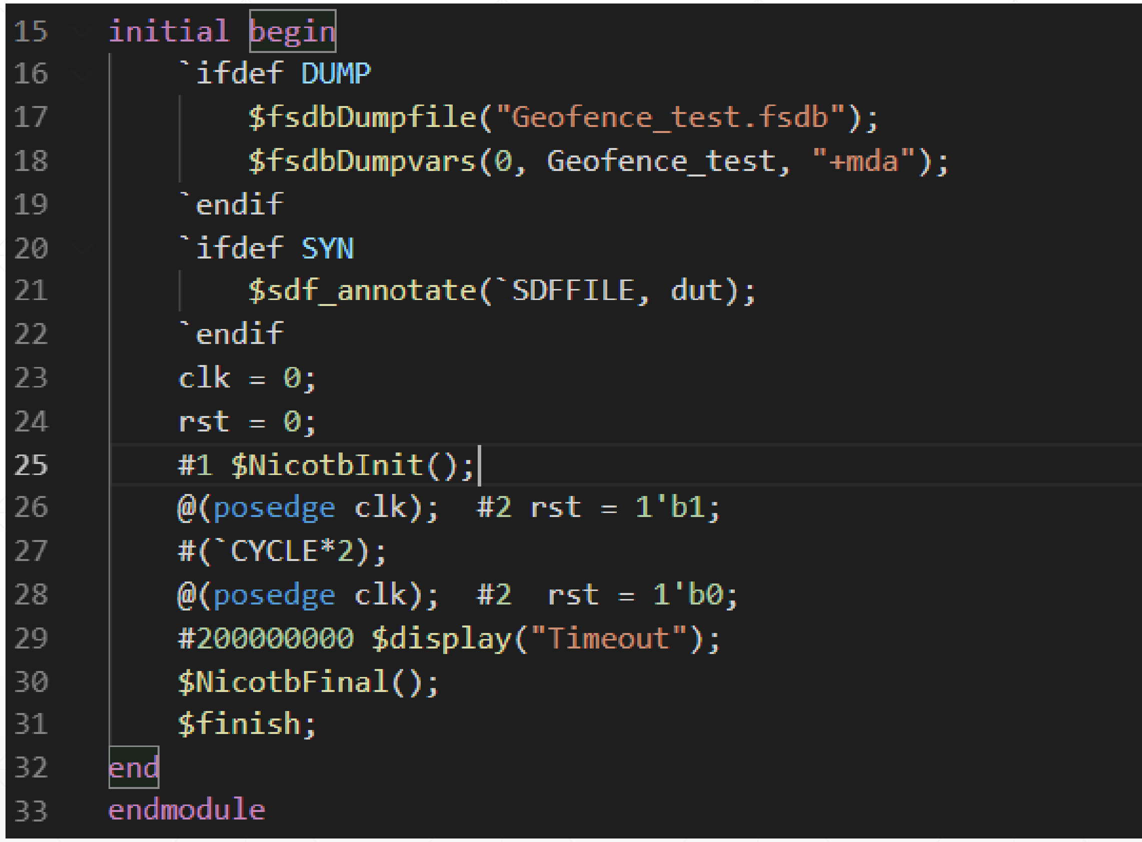
```
`NegIf(ck_ev, clk, ~rst)
```

正緣給值用`Pos
負緣給值用`Neg
Synchronous reset用`Pos/`Neg
ASynchronous reset用`PosIf/`NegIf

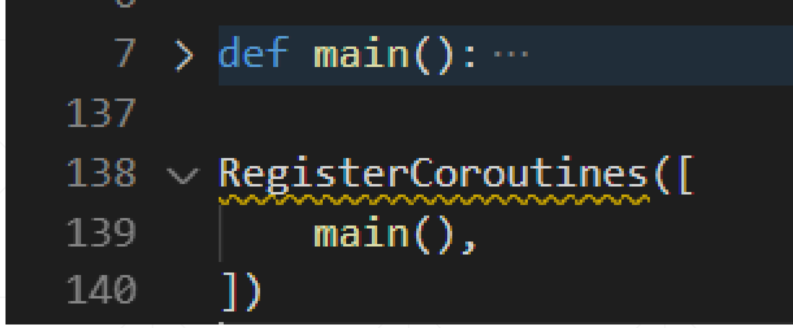
Active high寫rst
Active low寫~rst

_test.sv-內容

- DUMP波型
- 控制clock, reset
- Timeout



```
initial begin
    `ifdef DUMP
        $fsdbDumpfile("Geofence_test.fsdb");
        $fsdbDumpvars(0, Geofence_test, "+mda");
    `endif
    `ifdef SYN
        $sdf_annotate(`SDFFILE, dut);
    `endif
    clk = 0;
    rst = 0;
    #1 $NicotbInit();
    @(posedge clk);  #2 rst = 1'b1;
    #(`CYCLE*2);
    @(posedge clk);  #2  rst = 1'b0;
    #200000000 $display("Timeout");
    $NicotbFinal();
    $finish;
end
endmodule
```


```
def main(): ···

RegisterCoroutines([
    main(),
])
```

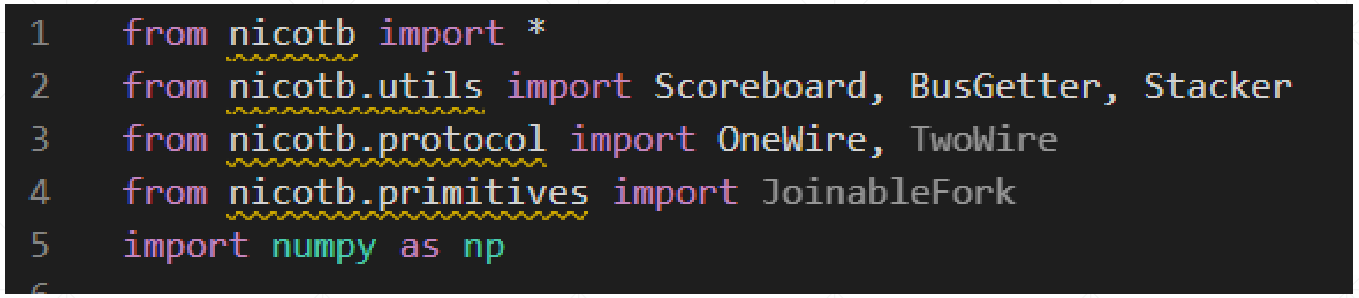
_test.py overview

- 準備input & 算golden pattern
- Master
  - 根據dut的反應傳送input data
- Slave
  - 收集dut的output
- Scoreboard
  - 對答案

需要寫的部分
寫在main()

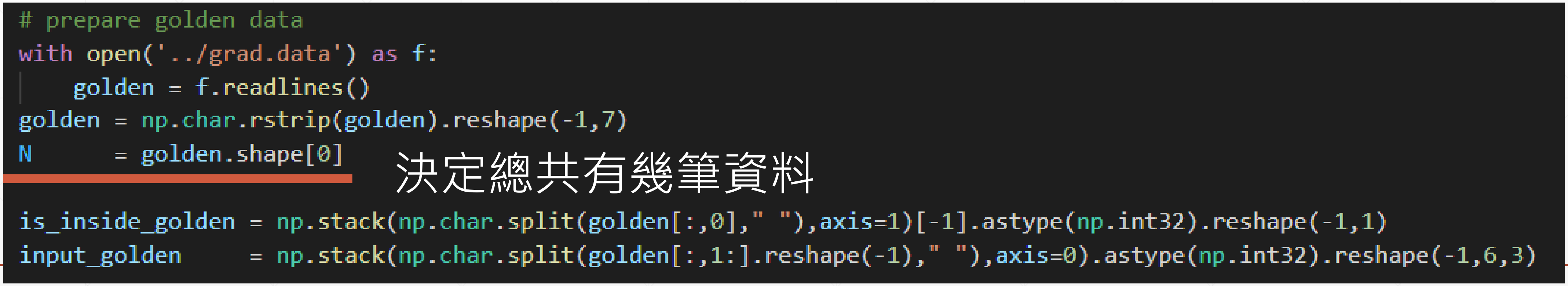
_test.py-import & data

- Import要用的package (numpy) 或自己的code (如behavior model)



```python
from nicotb import *
from nicotb.utils import Scoreboard, BusGetter, Stacker
from nicotb.protocol import OneWire, TwoWire
from nicotb.primitives import JoinableFork
import numpy as np
```

- 準備input & 算golden pattern



```python
# prepare golden data
with open('../grad.data') as f:
    golden = f.readlines()
golden = np.char.rstrip(golden).reshape(-1,7)
N      = golden.shape[0]

is_inside_golden = np.stack(np.char.split(golden[:,0]," "),axis=1)[-1].astype(np.int32).reshape(-1,1)
input_golden     = np.stack(np.char.split(golden[:,1:].reshape(-1)," "),axis=0).astype(np.int32).reshape(-1,6,3)
```

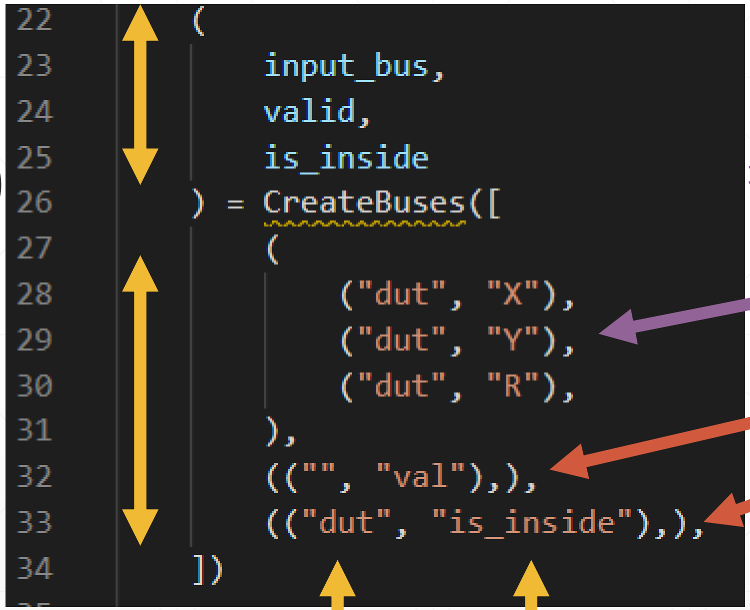
決定總共有幾筆資料

_test.py-宣告bus

- 利用CreateBuses()

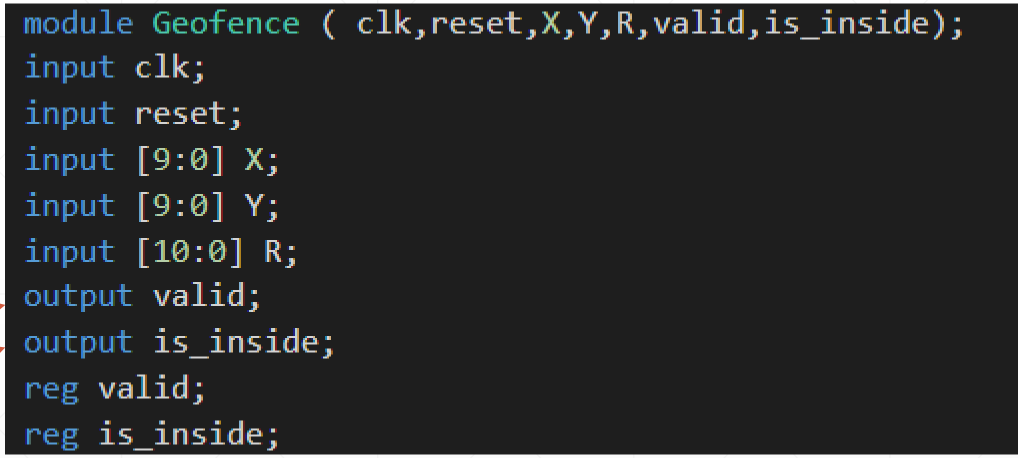
Python的變數
(numpy array)

Verilog的訊號



```
(
    input_bus,
    valid,
    is_inside
) = CreateBuses([
    (
        ("dut", "X"),
        ("dut", "Y"),
        ("dut", "R"),
    ),
    (("", "val"),),
    (("dut", "is_inside"),),
])
```

把訊號包成bus



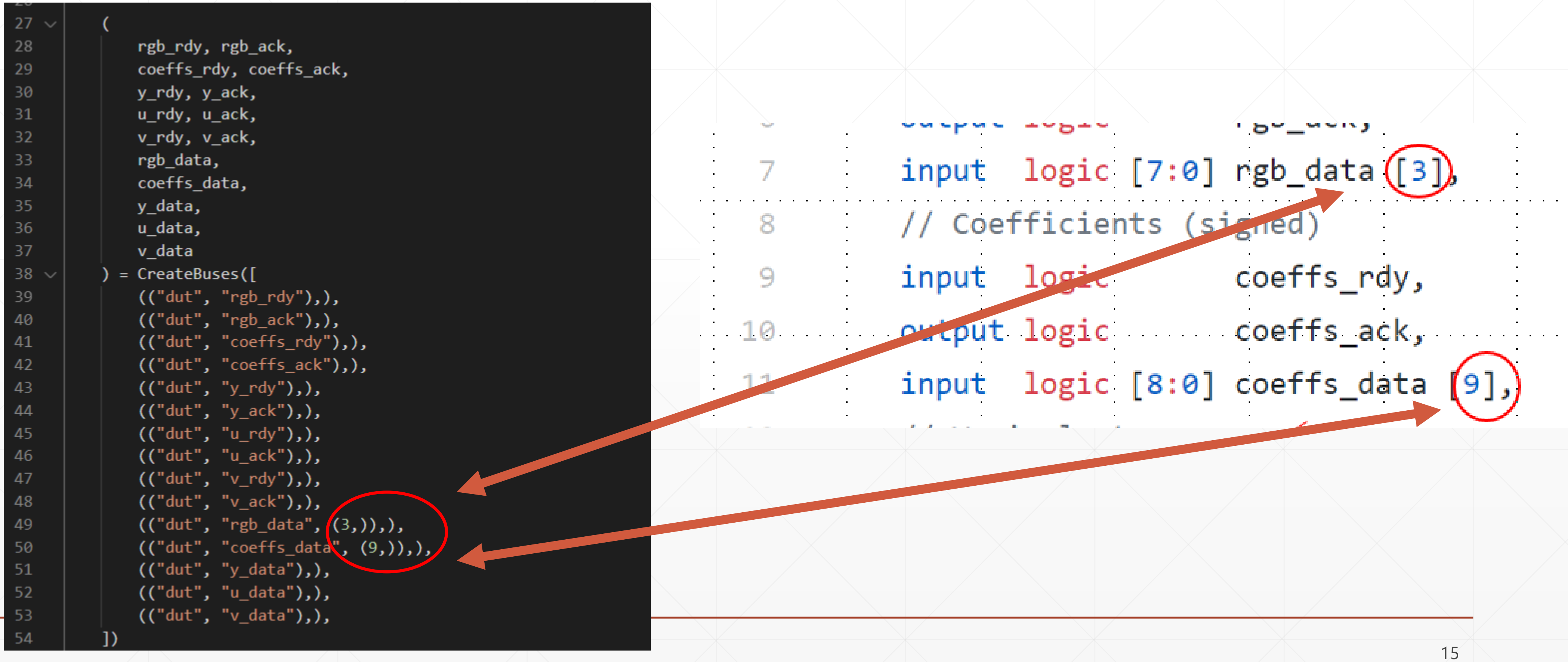
```
module Geofence ( clk,reset,X,Y,R,valid,is_inside);
input clk;
input reset;
input [9:0] X;
input [9:0] Y;
input [10:0] R;
output valid;
output is_inside;
reg valid;
reg is_inside;
```

訊號名字

訊號的階層
如dut.u1.u2.u3
若在_test.sv則可以寫“”

_test.py-宣告bus

- 二維array範例



```
    (
        rgb_rdy, rgb_ack,
        coeffs_rdy, coeffs_ack,
        y_rdy, y_ack,
        u_rdy, u_ack,
        v_rdy, v_ack,
        rgb_data,
        coeffs_data,
        y_data,
        u_data,
        v_data
    ) = CreateBuses([
        (("dut", "rgb_rdy"),),
        (("dut", "rgb_ack"),),
        (("dut", "coeffs_rdy"),),
        (("dut", "coeffs_ack"),),
        (("dut", "y_rdy"),),
        (("dut", "y_ack"),),
        (("dut", "u_rdy"),),
        (("dut", "u_ack"),),
        (("dut", "v_rdy"),),
        (("dut", "v_ack"),),
        (("dut", "rgb_data", (3,)),),
        (("dut", "coeffs_data", (9,)),),
        (("dut", "y_data"),),
        (("dut", "u_data"),),
        (("dut", "v_data"),),
    ])
```

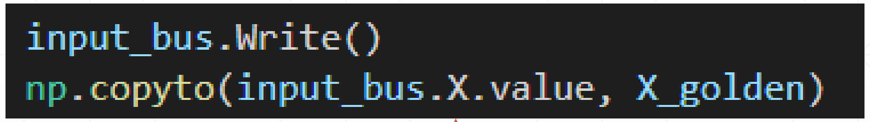
```
input   logic [7:0] rgb_data [3],
// Coefficients (signed)
input   logic       coeffs_rdy,
output  logic       coeffs_ack,
input   logic [8:0] coeffs_data [9],
```

_test.py-bus讀寫

- **Bus.value** : bus現在的值 (numpy array)

- Write

每次寫新的值
都要.Write()



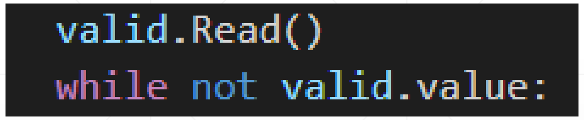
```
input_bus.Write()
np.copyto(input_bus.X.value, X_golden)
```

指定bus中的其中1條訊號

Numpy array

- Read

每次讀
都要.Read()

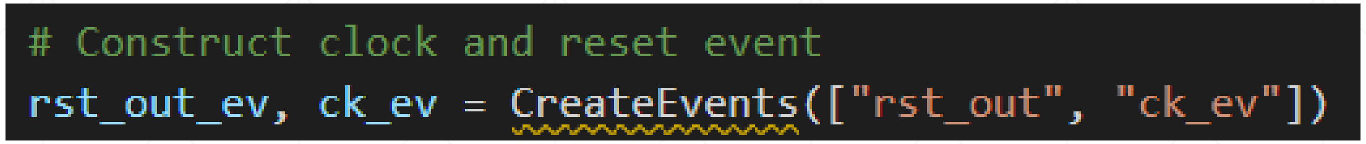


```
valid.Read()
while not valid.value:
```

Bus.value會回傳bus現在的值

_test.py-宣告event

- 利用CreateEvents()

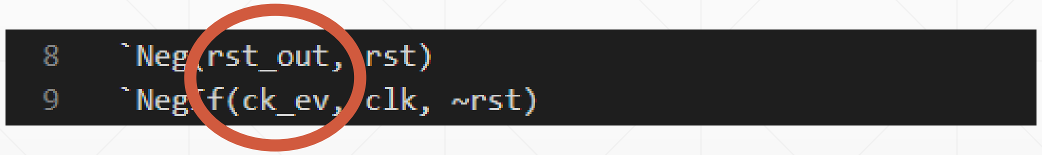


```python
# Construct clock and reset event
rst_out_ev, ck_ev = CreateEvents(["rst_out", "ck_ev"])
```

Python event名字

verilog event名字

宣告隱藏在_test.sv這兩行中:



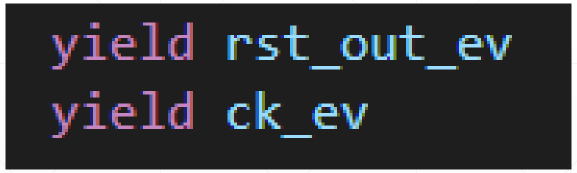
```
`Neg(rst_out, rst)
`NegIf(ck_ev, clk, ~rst)
```

_test.py-等待event發生

- 利用yield

程式執行到yield會等event發生才往下執行

在_test.sv宣告event時決定正 or 負緣

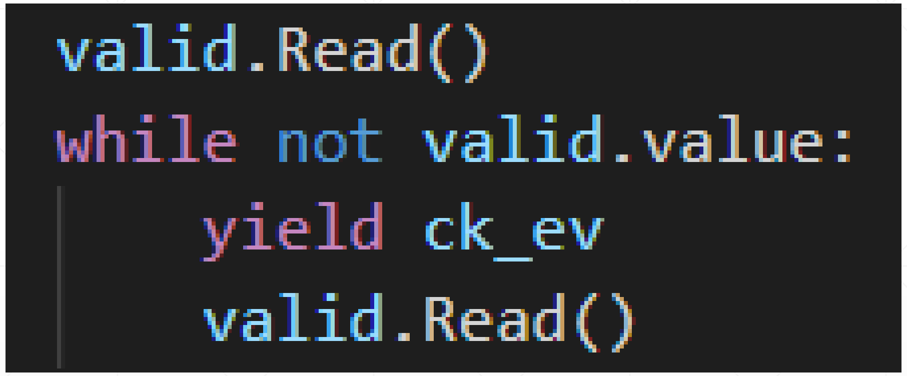


```
yield rst_out_ev
yield ck_ev
```

≈

@(posedge rst)
@(posedge clk)

_test.py-event範例



```
valid.Read()
while not valid.value:
    yield ck_ev
    valid.Read()
```

≈

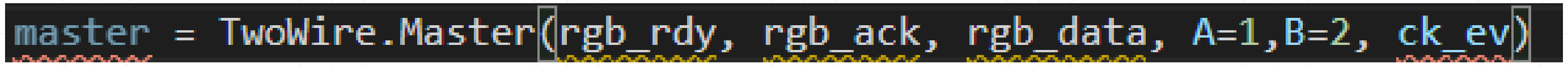
```
val_ev = CreateEvent(“val_ev”)
yield val_ev
```

_test.py-Master

根據dut反應傳送input data

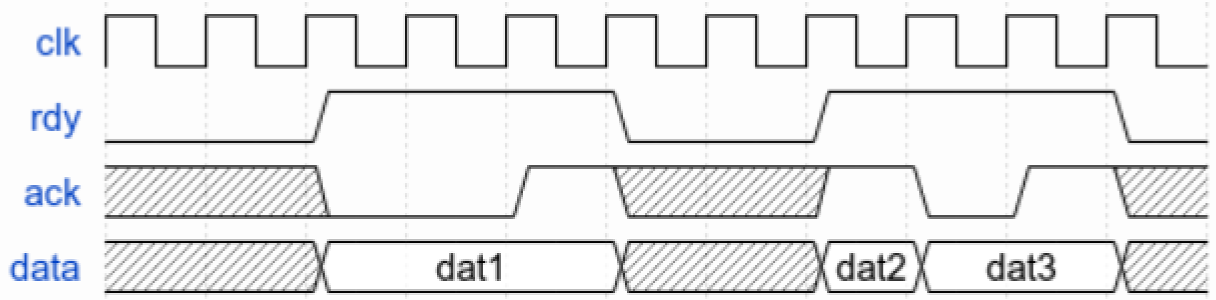
- Two Wire

有A/B 的機率會傳valid / ack



```
master = TwoWire.Master(rgb_rdy, rgb_ack, rgb_data, A=1,B=2, ck_ev)
```

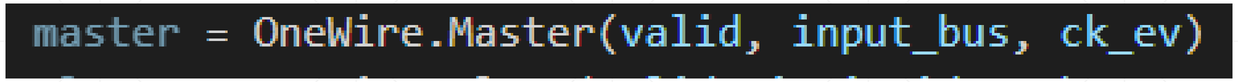
Handshake or 2-wire interface. A rdy signal is used to indicate whether the bundled data is valid at this clock, but an extra ack signal is used to hold the data until ack is asserted. In AXI bus protocol, it is known as the ready/valid pair of both address and data channels. Sometimes hardware designers might require ack to be de-asserted when rdy is de-asserted.



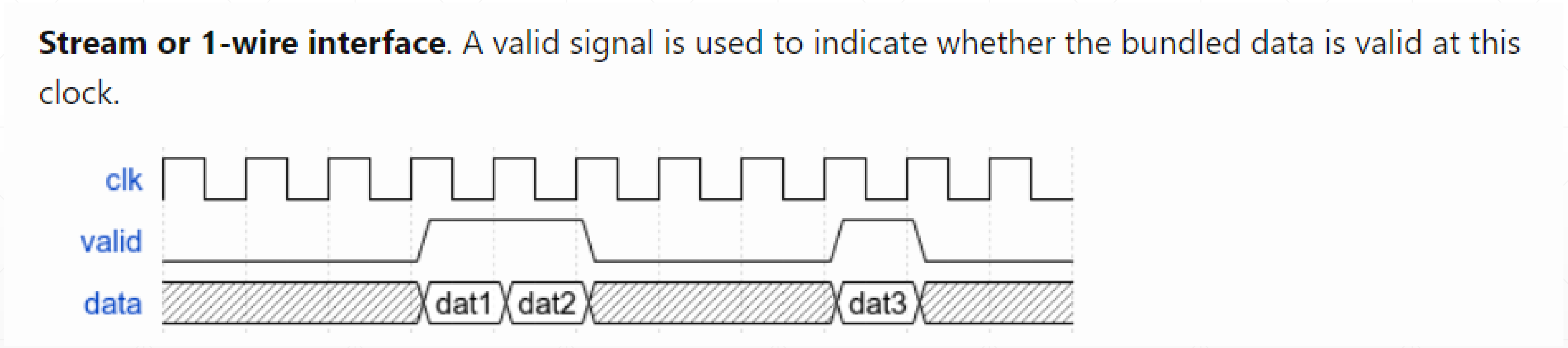
_test.py-Master

根據dut反應傳送input data

- One Wire

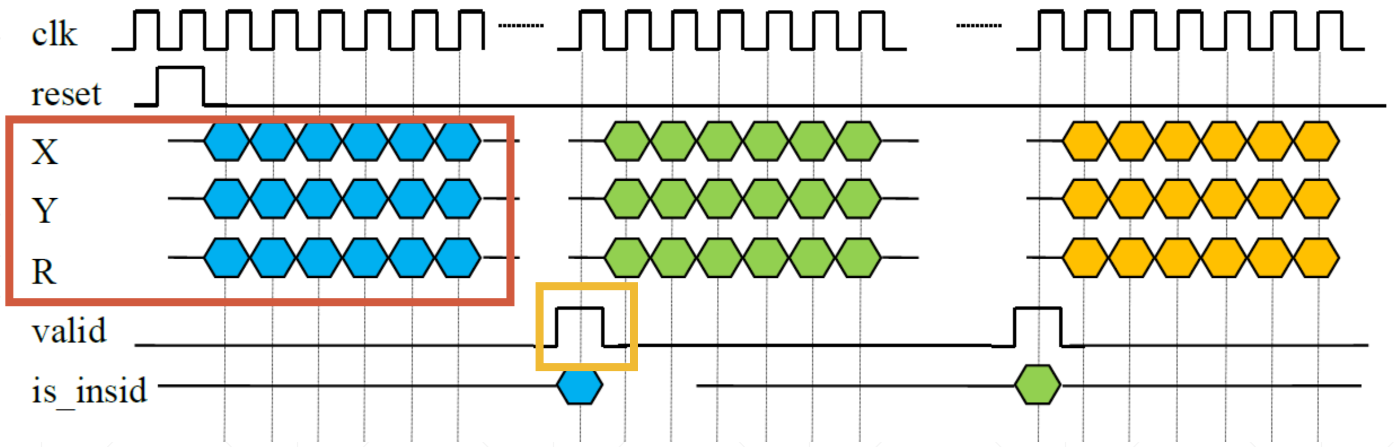


```
master = OneWire.Master(valid, input_bus, ck_ev)
```



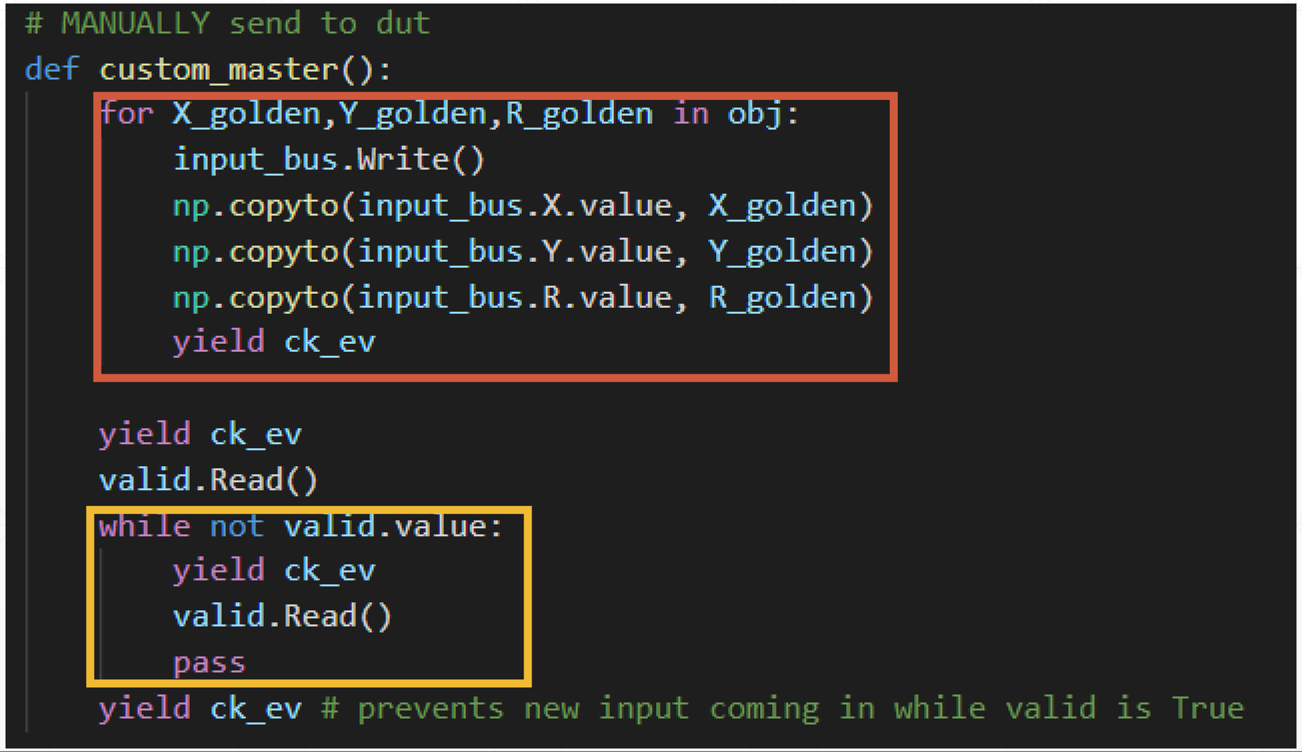
Stream or 1-wire interface. A valid signal is used to indicate whether the bundled data is valid at this clock.

_test.py-Master

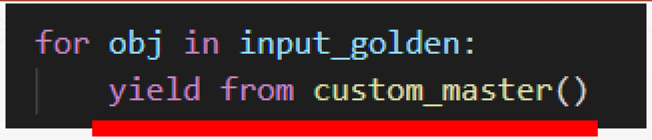


根據dut反應傳送input data

- Custom



```
# MANUALLY send to dut
def custom_master():
    for X_golden,Y_golden,R_golden in obj:
        input_bus.Write()
        np.copyto(input_bus.X.value, X_golden)
        np.copyto(input_bus.Y.value, Y_golden)
        np.copyto(input_bus.R.value, R_golden)
        yield ck_ev

    yield ck_ev
    valid.Read()
    while not valid.value:
        yield ck_ev
        valid.Read()
        pass
    yield ck_ev # prevents new input coming in while valid is True
```



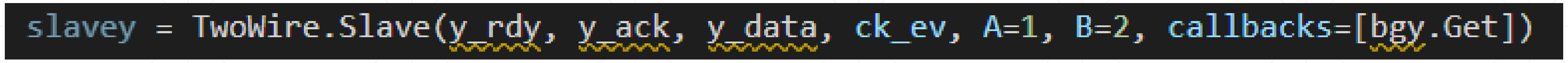
```
for obj in input_golden:
    yield from custom_master()
```

執行custom_master()時每到yield時就跳出程式回到_test.sv

_test.py-Slave

根據dut反應收集output

- Two Wire



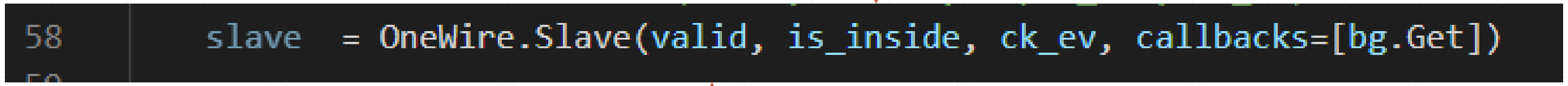
```
slavey = TwoWire.Slave(y_rdy, y_ack, y_data, ck_ev, A=1, B=2, callbacks=[bgy.Get])
```

- One Wire
- Custom

_test.py-Slave

根據dut反應收集output

- One Wire

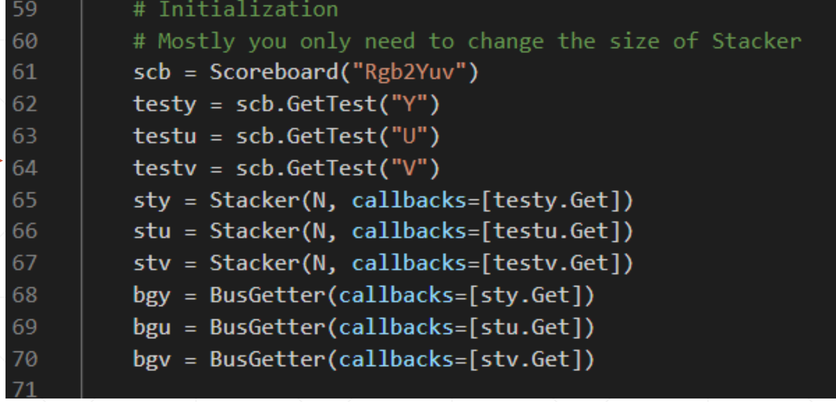


```
    slave   = OneWire.Slave(valid, is_inside, ck_ev, callbacks=[bg.Get])
```

_test.py-對答案

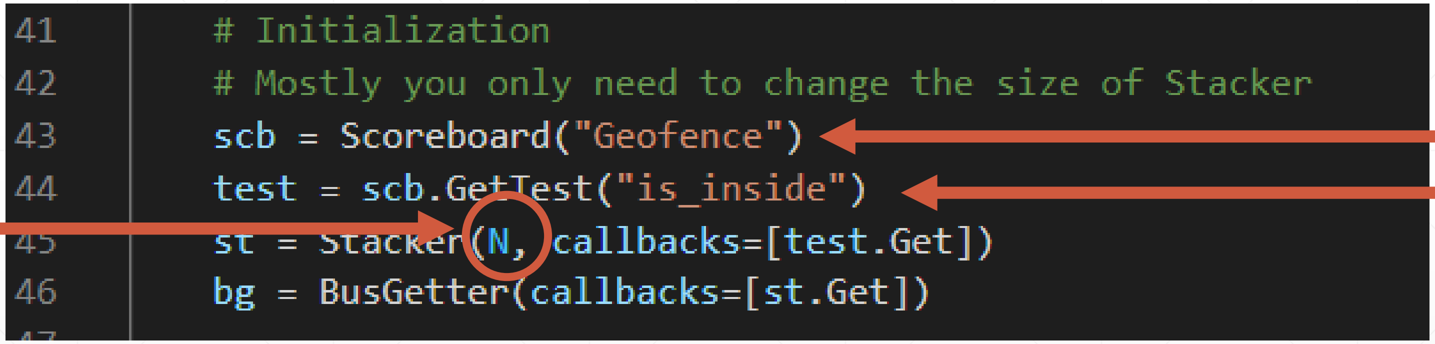
Scoreboard, gettest, stacker, busgetter

如果很多output訊號要對答案
就宣告很多組
但scoreboard只需一個



```
# Initialization
# Mostly you only need to change the size of Stacker
scb = Scoreboard("Rgb2Yuv")
testy = scb.GetTest("Y")
testu = scb.GetTest("U")
testv = scb.GetTest("V")
sty = Stacker(N, callbacks=[testy.Get])
stu = Stacker(N, callbacks=[testu.Get])
stv = Stacker(N, callbacks=[testv.Get])
bgy = BusGetter(callbacks=[sty.Get])
bgu = BusGetter(callbacks=[stu.Get])
bgv = BusGetter(callbacks=[stv.Get])
```

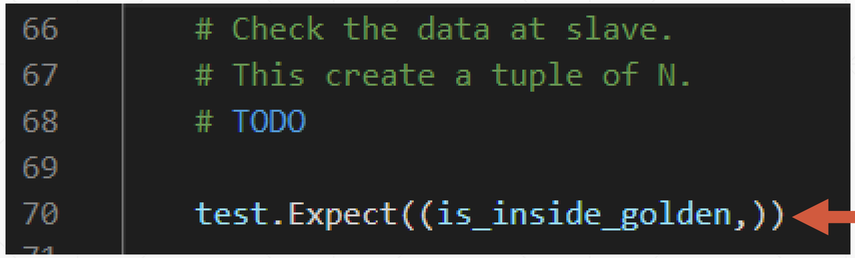
總共幾筆data要對答案
在產生input時決定



```
# Initialization
# Mostly you only need to change the size of Stacker
scb = Scoreboard("Geofence")
test = scb.GetTest("is_inside")
st = Stacker(N, callbacks=[test.Get])
bg = BusGetter(callbacks=[st.Get])
```

Scoreboard大標題

Scoreboard小標題

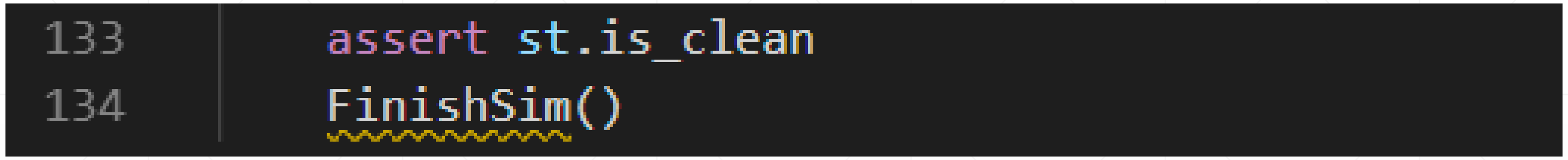


```
# Check the data at slave.
# This create a tuple of N.
# TODO

test.Expect((is_inside_golden,))
```

填入golden pattern

_test.py-結束模擬

確定dut的output和golden pattern的數量一致



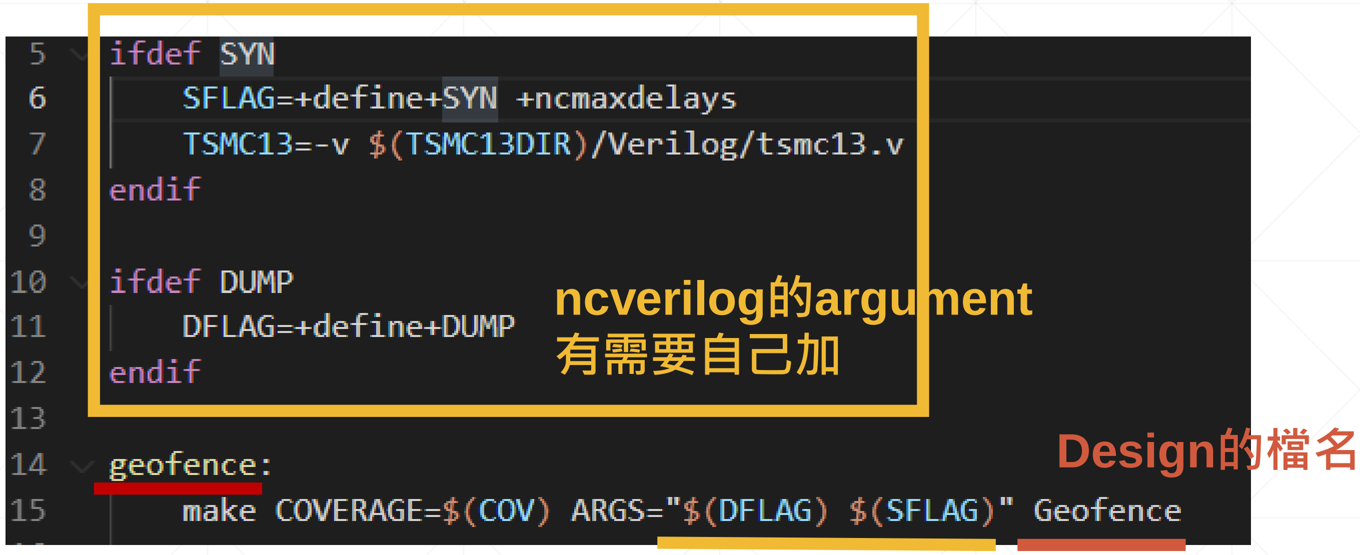
```
    assert st.is_clean
    FinishSim()
```

makefile

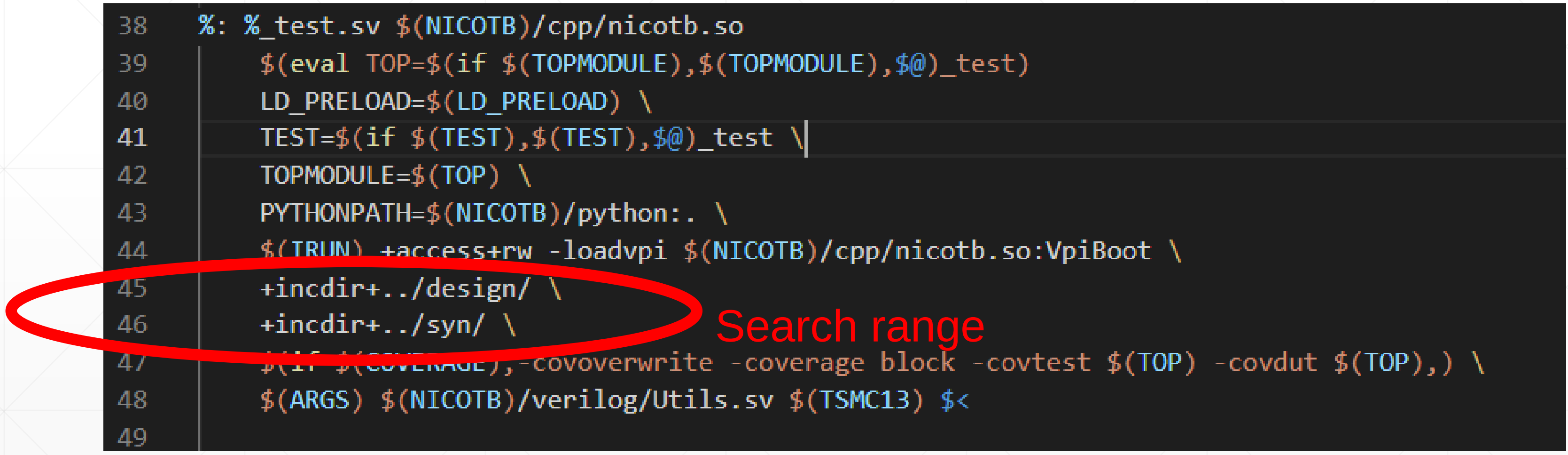
用來執行command line

如ncverilog _test.sv design.v ....

大部分不須更動

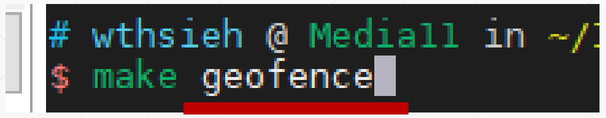


```
ifdef SYN
	SFLAG=+define+SYN +ncmaxdelays
	TSMC13=-v $(TSMC13DIR)/Verilog/tsmc13.v
endif

ifdef DUMP
	DFLAG=+define+DUMP
endif

geofence:
	make COVERAGE=$(COV) ARGS="$(DFLAG) $(SFLAG)" Geofence
```



```
%: %_test.sv $(NICOTB)/cpp/nicotb.so
	$(eval TOP=$(if $(TOPMODULE),$(TOPMODULE),$@)_test)
	LD_PRELOAD=$(LD_PRELOAD) \
	TEST=$(if $(TEST),$(TEST),$@)_test \
	TOPMODULE=$(TOP) \
	PYTHONPATH=$(NICOTB)/python:. \
	$(IRUN) +access+rw -loadvpi $(NICOTB)/cpp/nicotb.so:VpiBoot \
	+incdir+../design/ \
	+incdir+../syn/ \
	$(if $(COVERAGE),-covoverwrite -coverage block -covtest $(TOP) -covdut $(TOP),) \
	$(ARGS) $(NICOTB)/verilog/Utils.sv $(TSMC13) $<
```

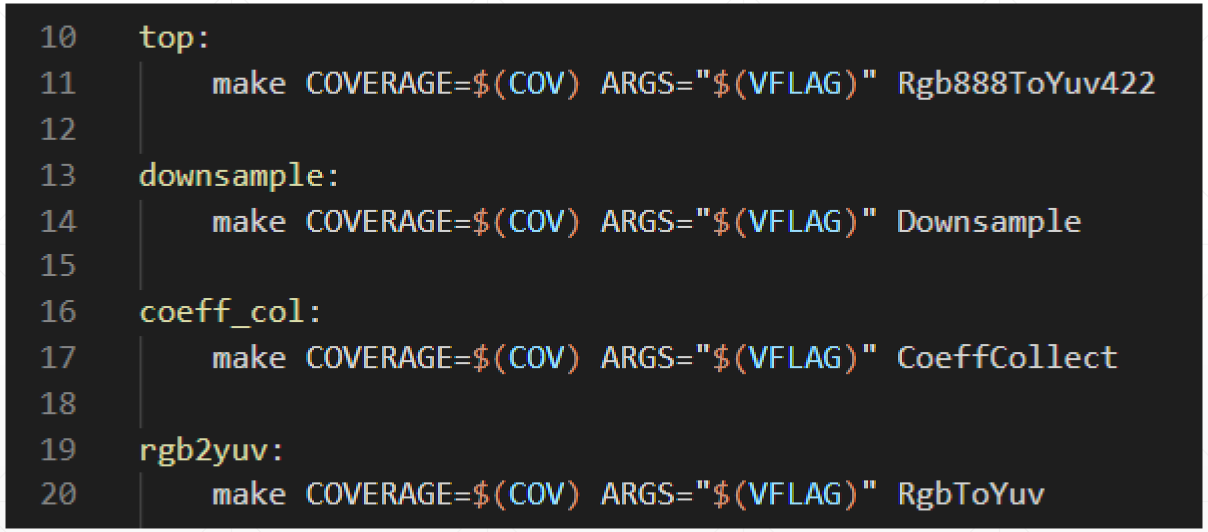
開始模擬:



```
# wthsieh @ Media11 in ~/
$ make geofence
```

makefile

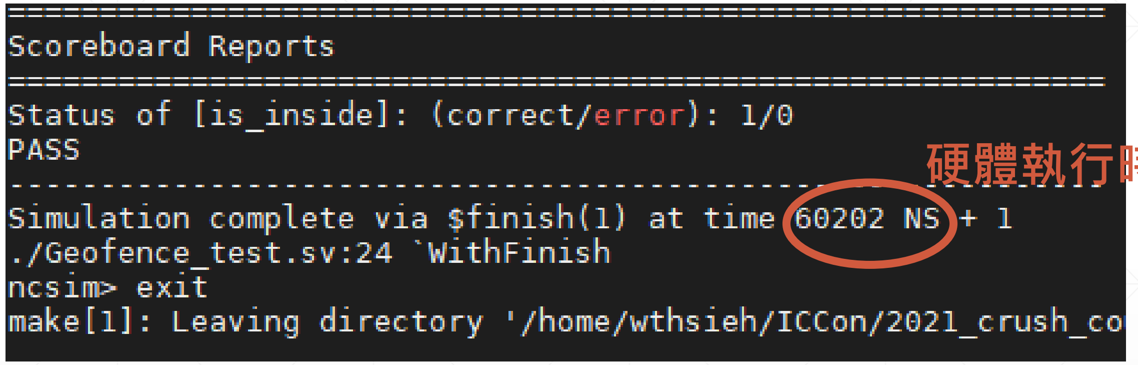
可以用同一份makefile控制很多要測的module



```
top:
    make COVERAGE=$(COV) ARGS="$(VFLAG)" Rgb888ToYuv422

downsample:
    make COVERAGE=$(COV) ARGS="$(VFLAG)" Downsample

coeff_col:
    make COVERAGE=$(COV) ARGS="$(VFLAG)" CoeffCollect

rgb2yuv:
    make COVERAGE=$(COV) ARGS="$(VFLAG)" RgbToYuv
```

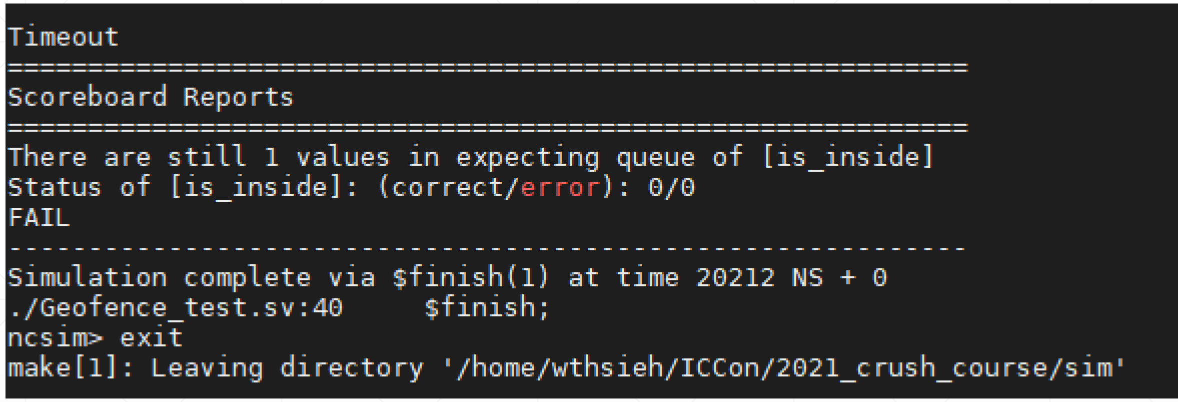
結果

- Pass



```
=============================================================
Scoreboard Reports
=============================================================
Status of [is_inside]: (correct/error): 1/0
PASS
-------------------------------------------------------------
Simulation complete via $finish(1) at time 60202 NS + 1
./Geofence_test.sv:24 `WithFinish
ncsim> exit
make[1]: Leaving directory '/home/wthsieh/ICCon/2021_crush_co
```

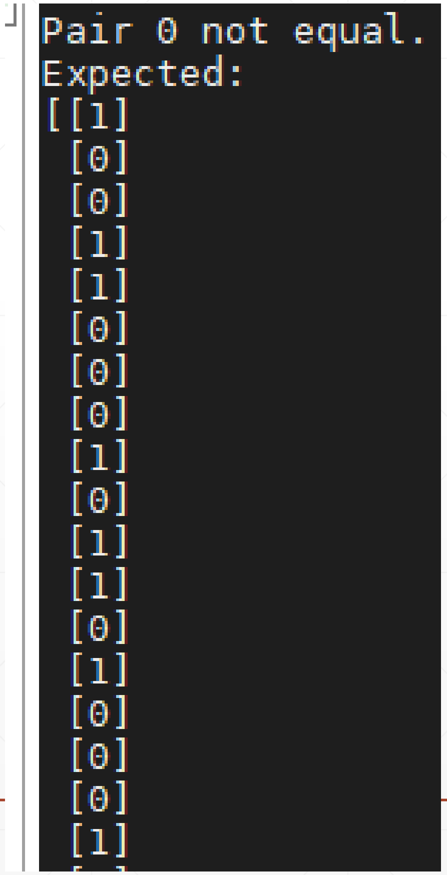
- Timeout



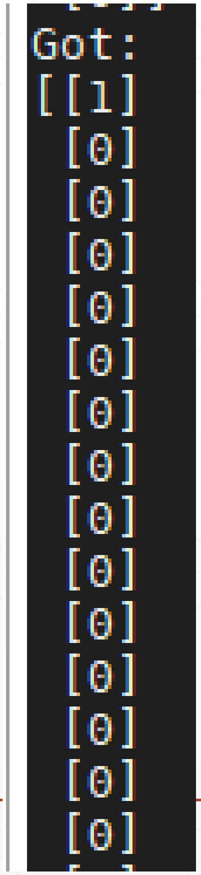
```
Timeout
=====================================================
Scoreboard Reports
=====================================================
There are still 1 values in expecting queue of [is_inside]
Status of [is_inside]: (correct/error): 0/0
FAIL
-----------------------------------------------------
Simulation complete via $finish(1) at time 20212 NS + 0
./Geofence_test.sv:40     $finish;
ncsim> exit
make[1]: Leaving directory '/home/wthsieh/ICCon/2021_crush_course/sim'
```

結果

- Fail



```
Pair 0 not equal.
Expected:
[[1]
 [0]
 [0]
 [1]
 [1]
 [0]
 [0]
 [0]
 [1]
 [0]
 [1]
 [1]
 [0]
 [1]
 [0]
 [0]
 [0]
 [1]
```



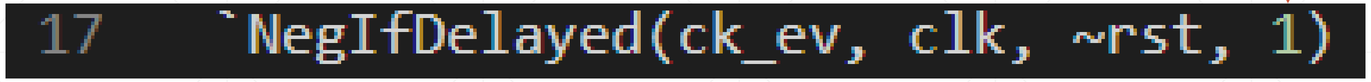
```
Got:
[[1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
```

Gate-level or post-layout 模擬

Delay 1 ns再觸發event

- _test.sv

Clock event換成有delay的版本

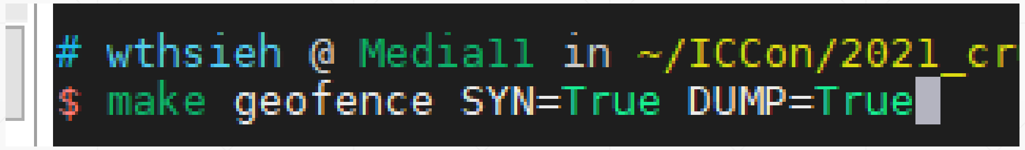


```
    `NegIfDelayed(ck_ev, clk, ~rst, 1)
```

Annotate SDF

- Comment line

加上ncverilog的argument



```
# wthsieh @ Media11 in ~/ICCon/2021_cr
$ make geofence SYN=True DUMP=True
```

Conclusion

- Use nicotb only when the module is large.

Reference

- https://johnjohnlin.github.io/nicotb/
- 洋彬