# Verilog Generator

## User Guide

**version 1.02**

**June 12, 2023**

# Important Notice

**『Freely available under the terms of the 3-Clause BSD License』**

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Document Revision History

| Doc Revision Number | Date | Description |
|---|---|---|
| 1.02 | June 12, 2023 | add default clock, add vfunctions ($LOG2, $RANGE, $DEMUX_BY_EN, $MULTICYCLE) |
| 1.01 | June 8, 2023 | add interface:set_top_uppercase function |
| 1.00 | May 8, 2023 | Initial relase |

# Table of Contents

# List of Tables

**Table** **Title** **Page**

# List of Figures

# List of Terms

| List | Description |
| --- | --- |
| TestDrive | TestDrive Profiling Master (https://testdrive-profiling-master.github.io/) |
| Lua | Lua script language (Wiki, Homepage) |

# 1. 소개

*"성능이란 건 편리함을 이기지 못한다."*

대단위 프로젝트의 verilog 디자인을 설계할 때, 가장 문제가 되는 부분 중 하나는 모듈간 control path 구성에 상당한 시간과 노력이 소요되다는 점입니다. 또한 많은 시간과 노력을 들여 한번 완성된 디자인의 control path 일부를 수정하거나, 대대적인 변경이 필요한 경우는 더더욱 신중을 기해야만 합니다. 그렇지 않으면 새로운 오류를 유발시키거나 디자인을 첨부터 다시 만드는 것과 동일한 노력이 필요하게 될 것입니다.

이에 최소한의 디자인 설계만으로 control path 를 생성해 주는 verigen 툴을 구상했습니다. 이 툴은 control path 를 프로그래밍 방식으로 최소한의 노력을 통해 쉽고 빠르게 구축할 수 있게 하며, 짜여진 control path 를 한 눈에 design hierarchy 로 확인 가능케 하는 기능을 가지고 있습니다. 또한 이로써 보다 빠른 설계 변경과 다른 팀원과의 디자인 공유가 가능할 수 있습니다.

**NOTE:** 신규 제안할 기능이 있거나 개선사항이나 버그를 발견하시면, 문의(clonextop@gmail.com) 부탁드립니다.

## 1.1 주요 기능

verigen 은 TestDrive Profiling Master 의 codegen 을 이용하여 제작되었습니다. 이 툴은 lua 로 작성된 코드를 실행하여, verilog 디자인을 구축하며, codegen 기능 전부를 포함하고 있으며, 아래 파일들을 생성합니다.

- verilog 디자인(.sv, .f) 자동 생성
- constraint(.xdc) 자동 생성
- hierarchy diagram(.svg), HTML highlited source code(.html) 자동 생성

## 1.2 verilog 자동화 생성 과정

verigen 을 통한 프로젝트 작성은 아래와 같은 단계로 진행됩니다.

1. Lua 스크립트 작성
    1). 모듈 생성
    2). 모듈 연결
    3). 모듈에 parameter 선언 및 연결
    4). 모듈에 interface 선언 및 연결
2. verigen 실행하여 코드 생성

## 1.3 실행 방법

verigen을 실행하기 위해서는 아래와 같이 실행합니다.

```
> verigen

Verilog Generator for TestDrive Profiling Master. v1.00
Usage: verigen  [--help] input_file [output_path]

      --help                display this help and exit
  input_file                input Lua file
  output_path               output path
                            default : ./output
```

**NOTE:** 실행 명령 : verigen INPUT_LUA_FILE  OUTPUT_PATH

INPUT_LUA_FILE에 해당하는 Lua 스크립트를 작성하여 실행하게 되며, OUTPUT_PATH을 지정하지 않을 경우 기본 "./output" 폴더에 결과를 생성합니다.

## 1.4 라이센스 허가

verigen에 구현된 소스는 BSD 라이센스를 준수하며, verilog 생성에 사용된 사용자의 개별 스크립트나 verilog 등의 2차 저작물은 온전히 사용자의 소유입니다.

# 2. Fast follow

이 단은 예제 중심으로 빠르게 설명하고 있습니다. 사전 형식으로 class 와 method 를 확인하려면, 다음 단 'Class and Method'를 참조하시기 바랍니다.

## 2.1 Step #1 : 모듈의 생성

아래와 같이 스크립트 코드를 생성하고 실행합니다.

**[main.lua 파일]**

```
 1: -- modules
 2: core_wrapper    = module:new("test_wrapper")        -- top
 3: core            = {}
 4: core.top        = module:new("test_core")
 5: core.slave_ctrl = module:new("slave_ctrl")
 6: core.core_if    = module:new("core_if")
 7: core.core_ex    = module:new("core_ex")
 8: core.core_wb    = module:new("core_wb")
 9: core.mem_ctrl   = module:new("mem_ctrl")
10: core.reg_ctrl   = module:new("reg_ctrl")
11: core.busy_ctrl  = module:new("busy_ctrl")
12:
13: -- make code
14: core_wrapper:make_code()
```

core_wrapper 모듈과 core 관련 모듈을 lua table 에 담아 생성한 것입니다.

**[실행]**

```
> verigen main.lua
*I: Build TOP design : test_wrapper.sv
*W: Empty port module : 'test_wrapper' module
*I: Build contraint : test_wrapper_constraint.xdc
*I: Make design hierarchy : test_wrapper_hierarchy.svg
*I: Make common defines : test_wrapper_defines.vh
*I: Make design file list : test_wrapper.f
```

module:new 메소드를 통해 사용 되어질 모듈들을 간략히 선언합니다. 그리고 마지막 module:make_code 메소드는 실제 verilog 코드, constraint 파일과 hierarchy diagram 을 생성해 냅니다. 현재는 여러개의 모듈을 선언하였으나, core_wrapper 연결된 모듈이 없기 때문에, verilog 디자인은 test_wrapper.sv 파일 하나만 생성합니다.

나머지 test_wrapper_constraint.xdc, test_wrapper_defines.vh 등은 비어있는 상태입니다.

**[결과 : test_wrapper.sv]**

```
`include "test_wrapper_defines.vh"
```

```
module test_wrapper ();

endmodule
```

결과 디자인은 말 그대로 빈 모듈 파일이며, 다이어그램도 역시 비어있는 상태입니다.

**[결과 : test_wrapper_hierarchy.svg]**

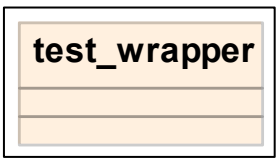


**Figure 2-1. Step #1 Hierarchy Diagram**

## 2.2 Step #2 : 모듈의 연결

아래와 같이 lua 스크립트를 수정하여 실행합니다.

**[main.lua 파일]**

```
 1: RunScript("test_definition.lua")
 2:
 3: -- modules
 4: core_wrapper    = module:new("test_wrapper")        -- top
 5: core            = {}
 6: core.top        = module:new("test_core")
 7: core.slave_ctrl = module:new("slave_ctrl")
 8: core.core_if    = module:new("core_if")
 9: core.core_ex    = module:new("core_ex")
10: core.core_wb    = module:new("core_wb")
11: core.mem_ctrl   = module:new("mem_ctrl")
12: core.reg_ctrl   = module:new("reg_ctrl")
13: core.busy_ctrl  = module:new("busy_ctrl")
14:
15: -- module connection
16: core_wrapper:add_module(core.reg_ctrl)
17: core_wrapper:add_module(core.mem_ctrl)
18: core.reg_ctrl:add_module(core.busy_ctrl)
19: core.reg_ctrl:add_module(core.slave_ctrl)
20:
21: core.top:add_module(core.core_if)
22: core.top:add_module(core.core_ex)
23: core.top:add_module(core.core_wb)
24:
25: -- multi-core genration
26: for i = 1, config.core_size, 1 do
27:    core_wrapper:add_module(core.top)
28: end
```

```
29:
30: -- make code
31: core_wrapper:make_code()
```

이제 추가된 15 ~ 28 라인에서 각 모듈을 module:add_module 함수로 연결하고, core 도 4개 모듈을 생성하여 연결했습니다. 미리 정의된 config.core_size 값을 사용하기 위해 상단(line #1)에 "Appendix : test_definition.lua" 를 포함하고 있습니다.

**[실행]**

```
> verigen main.lua
*I: Build sub design : mem_ctrl.sv
*W: Empty port module : 'mem_ctrl' module
*I: Build sub design : busy_ctrl.sv
*W: Empty port module : 'busy_ctrl' module
*I: Build sub design : slave_ctrl.sv
*W: Empty port module : 'slave_ctrl' module
*I: Build sub design : reg_ctrl.sv
*W: Empty port module : 'reg_ctrl' module
*I: Build sub design : core_ex.sv
*W: Empty port module : 'core_ex' module
*I: Build sub design : core_if.sv
*W: Empty port module : 'core_if' module
*I: Build sub design : core_wb.sv
*W: Empty port module : 'core_wb' module
*I: Build sub design : test_core.sv
*W: Empty port module : 'test_core' module
*I: Build TOP design : test_wrapper.sv
*W: Empty port module : 'test_wrapper' module
*I: Build contraint : test_wrapper_constraint.xdc
*I: Make design hierarchy : test_wrapper_hierarchy.svg
*I: Make common defines : test_wrapper_defines.vh
*I: Make design file list : test_wrapper.f
```

실행된 결과에서 test_wrapper.sv 파일 말고도 포함된 다른 파일들도 자동 생성되며, top design 을 본다면 아직 모듈만 추가 선언된 상태이고 포트는 기술되지 않았으므로, 경고를 발생하지만 아래처럼 각 서브 모듈이 자동으로 추가된 모습을 볼 수 있습니다.

**[결과 : test_wrapper.sv]**

```
`include "test_wrapper_defines.vh"

module test_wrapper ();

mem_ctrl mem_ctrl (
);

reg_ctrl reg_ctrl (
);

test_core test_core_0 (
);
```

```
test_core test_core_1 (
);

test_core test_core_2 (
);

test_core test_core_3 (
);

endmodule
```

**[결과 : test_wrapper_hierarchy.svg]**

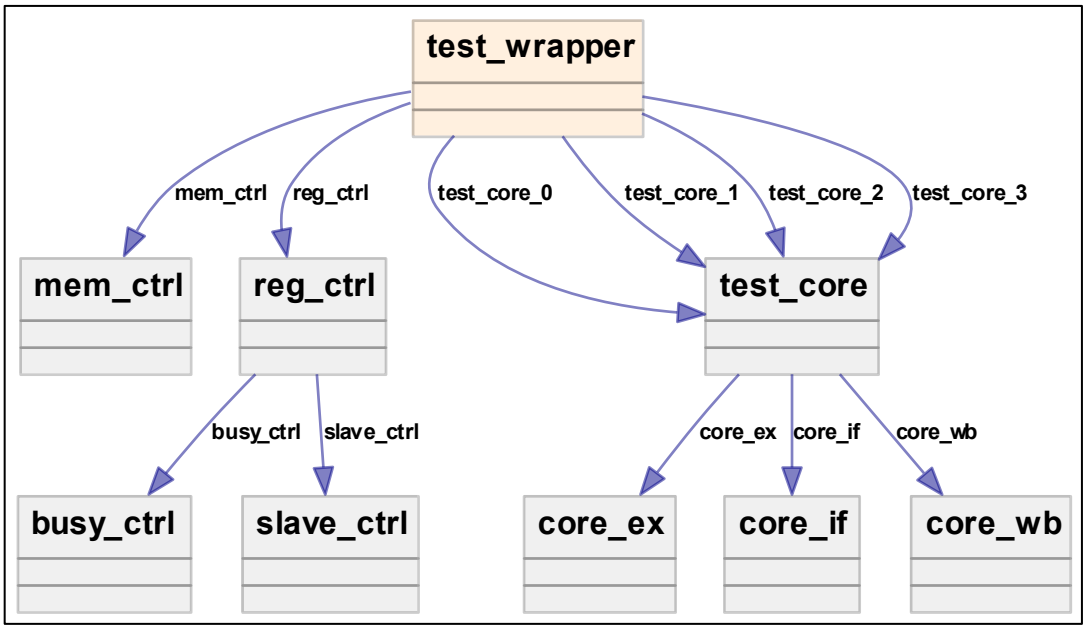


**Figure 2-2. Step #2 Hierarchy Diagram**

test_wrapper.sv 이외에도 다른 reg_ctrl.sv, test_core.sv 안에서도 Figure 2-2에서 볼 수 있듯이, 각각 하위 모듈들을 포함하고 있습니다.

## 2.3 Step #3 : parameter 추가

아래와 같이 lua 스크립트를 수정하여 실행합니다.

**[main.lua 파일]**

```
1: RunScript("test_definition.lua")
2:
3: -- modules
4: core_wrapper     = module:new("test_wrapper")        -- top
5: core             = {}
```

```
 6: core.top        = module:new("test_core")
 7: core.slave_ctrl = module:new("slave_ctrl")
 8: core.core_if    = module:new("core_if")
 9: core.core_ex    = module:new("core_ex")
10: core.core_wb    = module:new("core_wb")
11: core.mem_ctrl   = module:new("mem_ctrl")
12: core.reg_ctrl   = module:new("reg_ctrl")
13: core.busy_ctrl  = module:new("busy_ctrl")
14:
15: -- module connection
16: core_wrapper:add_module(core.reg_ctrl)
17: core_wrapper:add_module(core.mem_ctrl)
18: core.reg_ctrl:add_module(core.busy_ctrl)
19: core.reg_ctrl:add_module(core.slave_ctrl)
20:
21: core.top:add_module(core.core_if)
22: core.top:add_module(core.core_ex)
23: core.top:add_module(core.core_wb)
24:
25: -- parameters
26: core.core_if:set_param("CORE_ID", "0")
27: core.reg_ctrl:set_param("BASE_ADDR", "32'h10000000")
28:
29: -- multi-core genration
30: for i = 1, config.core_size, 1 do
31:    local    core_inst   = core_wrapper:add_module(core.top)
32:    core_inst:set_param("CORE_ID", i)
33: end
34:
35: -- make code
36: core_wrapper:make_code()
```

추가또는 변경된 코드는 line #26,27,31,32 입니다.

core 의 인스턴스 마다 CORE_ID 를 부여하기 위해, line #26 에서 module:set_param 을 통해 지정했습니다. 또한 line #31 에서 생성한 모든 core module instance 에 각각 CORE_ID 를 지정합니다.

reg_ctrl 의 경우는 BASE_ADDR parameter 를 추가했는데, 상위에서 이를 지정하지 않았기 때문에, Figure 2-3과 같이 그대로 top 까지 parameter 가 전달되어 선언됩니다.

**[결과 : test_wrapper_hierarchy.svg]**

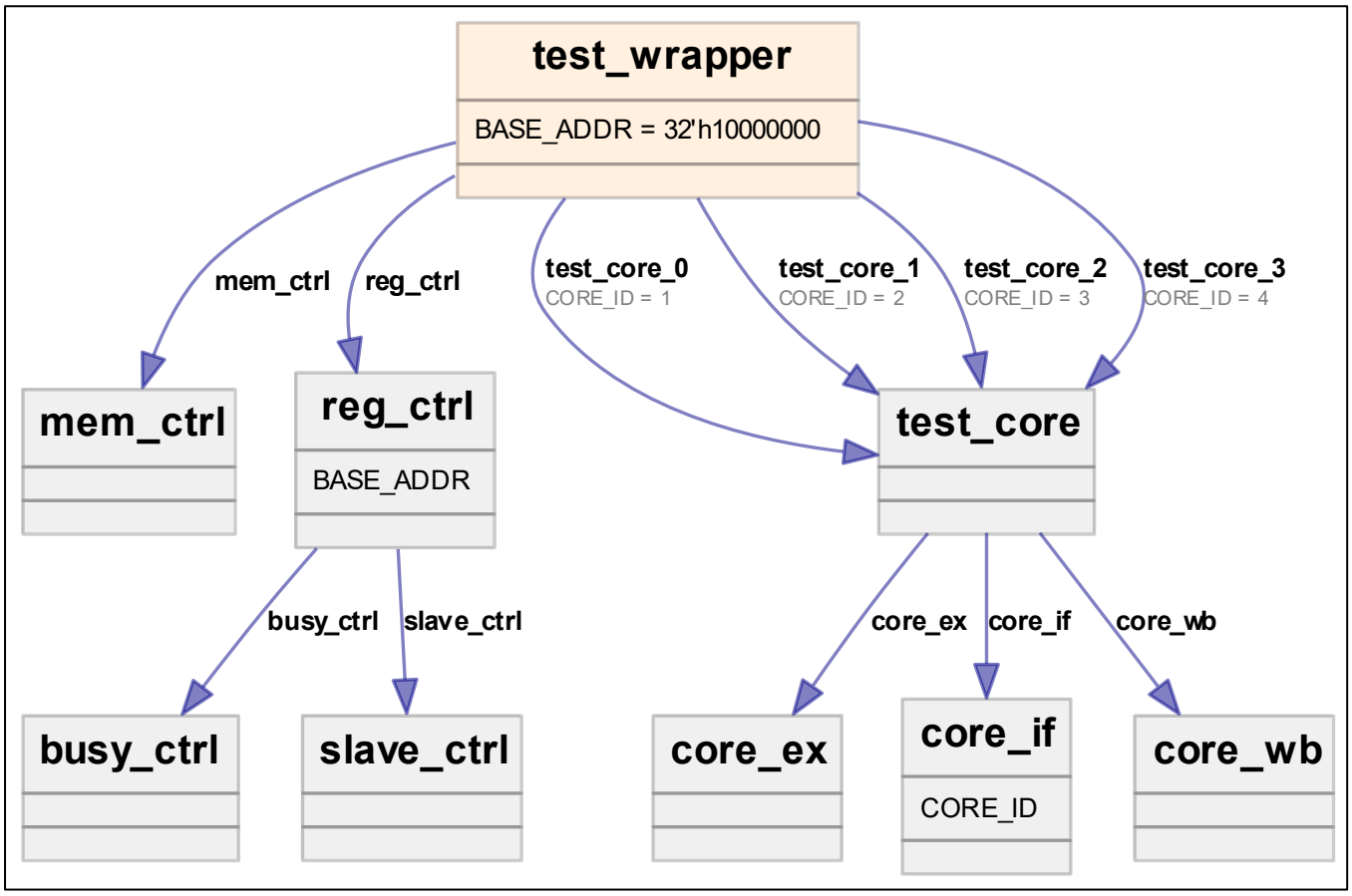


**Figure 2-3. Step #3 Hierarchy Diagram**

아래는 top 의 결과물입니다.

**[결과 : test_wrapper.sv]**

```
`include "test_wrapper_defines.vh"

module test_wrapper #(
    parameter BASE_ADDR         = 32'h10000000
) ();

mem_ctrl mem_ctrl (
);

reg_ctrl #(
    .BASE_ADDR          (BASE_ADDR)
) reg_ctrl (
);

test_core #(
    .CORE_ID            (1)
) test_core_0 (
);

test_core #(
```

```
    .CORE_ID            (2)
) test_core_1 (
);

test_core #(
    .CORE_ID            (3)
) test_core_2 (
);

test_core #(
    .CORE_ID            (4)
) test_core_3 (
);

endmodule
```

## 2.4 Step #4 : interface 추가

아래와 같이 lua 스크립트를 수정하여 실행합니다.

**[main.lua 파일]**

```
 1: RunScript("test_definition.lua")
 2:
 3: -- modules
 4: core_wrapper    = module:new("test_wrapper")        -- top
 5: core            = {}
 6: core.top        = module:new("test_core")
 7: core.slave_ctrl = module:new("slave_ctrl")
 8: core.core_if    = module:new("core_if")
 9: core.core_ex    = module:new("core_ex")
10: core.core_wb    = module:new("core_wb")
11: core.mem_ctrl   = module:new("mem_ctrl")
12: core.reg_ctrl   = module:new("reg_ctrl")
13: core.busy_ctrl  = module:new("busy_ctrl")
14:
15: -- module connection
16: core_wrapper:add_module(core.reg_ctrl)
17: core_wrapper:add_module(core.mem_ctrl)
18: core.reg_ctrl:add_module(core.busy_ctrl)
19: core.reg_ctrl:add_module(core.slave_ctrl)
20:
21: core.top:add_module(core.core_if)
22: core.top:add_module(core.core_ex)
23: core.top:add_module(core.core_wb)
24:
25: -- parameters
26: core.core_if:set_param("CORE_ID", "0")
27: core.reg_ctrl:set_param("BASE_ADDR", "32'h10000000")
28:
29: -- add slave bus interface
```

```
30: i_apb = core.slave_ctrl:add_interface(bus.apb, "s_apb")
31: i_apb:set_port("m")
32: i_apb:set_desc("APB's control bus")
33:
34: -- add master bus interface
35: bus.mbus    = bus.maxi4:new("mbus")
36: bus.mbus:set_param("DATA_WIDTH", 512)
37: bus.mbus:set_param("ADDR_WIDTH", 36)
38: bus.mbus:set_prefix("M#")
39:
40: core.mem_ctrl:add_interface(bus.mbus, "maxi"):set_port("m")
41:
42: -- add busy signal interface
43: core_busy_all   = new_signal("core_busy_all", config.core_size)
44: core_busy       = new_signal("core_busy")
45:
46: core.top:add_interface(core_busy):set_port("m")
47: core_wrapper:add_interface(core_busy_all)
48: core.busy_ctrl:add_interface(core_busy_all):set_port("s")
49:
50: -- multi-core genration
51: for i = 1, config.core_size, 1 do
52:    local    core_inst   = core_wrapper:add_module(core.top)
53:    core_inst:set_param("CORE_ID", i)
54:    core_inst:set_port("core_busy", "core_busy_all[" .. (i-1) .. "]")
55: end
56:
57: -- add instruction interface
58: core.core_if:add_interface(core_i.inst, "if_inst"):set_port("m")
59: core.core_ex:add_interface(core_i.inst, "if_inst"):set_port("s")
60:
61: core.core_ex:add_interface(core_i.inst, "ex_inst"):set_port("m")
62: core.core_wb:add_interface(core_i.inst, "ex_inst"):set_port("s")
63:
64: -- make code
65: core_wrapper:make_code()
```

### 2.4.1 Slave APB 연결

먼저 line #29~32 사이의 APB 버스 추가입니다.

```
...
-- add slave bus interface
i_apb = core.slave_ctrl:add_interface(bus.apb, "s_apb")
i_apb:set_port("m")
i_apb:set_desc("APB's control bus")
...
```

bus.apb 는 "Appendix : test_definition.lua"에서 먼저 정의된 APB bus 입니다. 위와 같이 module:add_interface 함수로 APB bus interface 의 instance 를 생성하고, interface_i:set_port 함수로 master(systemverilog 의 modport) 로 포트로 기술합니다. (포트로 선언하지 않으면, 기본은 내부 시그널

라우팅용 선언입니다.)

interface_i:set_desc 함수는 주석 설명을 기술하는 방법입니다.

이렇게 생성된 버스 인터페이스 instance는 Figure 2-4 에서 볼 수 있듯이 상위 모듈 reg_ctrl 과 top 모듈에서 별다른 언급이 없기 때문에, 최종 top 의 port 로 연결되집니다.

### 2.4.2 Master AXI 연결

두번째로 line #34~38 의 master AXI 버스 추가입니다.

```
...
-- add master bus interface
bus.mbus    = bus.maxi4:new("mbus")
bus.mbus:set_param("DATA_WIDTH", 512)
bus.mbus:set_param("ADDR_WIDTH", 36)
bus.mbus:set_prefix("M#")

core.mem_ctrl:add_interface(bus.mbus, "maxi"):set_port("m")
...
```

이번엔 버스 인터페이스를 바로 추가하지 않고 복제하여, 수정하여 추가하는 방법입니다.

"Appendix : test_definition.lua"에서 이미 선언된 bus.maxi4 인터페이스를 'mbus' 이름으로 interface:new 함수를 통해 bus.mbus 에 복제한 후, DATA_WIDTH 와 ADDR_WIDTH 에 원하는 값을 interface:set_param 함수로 설정합니다.

interface:set_prefix 를 사용하는 이유는 top design 의 포트로 출력될 때, 내부 시그널들이 input/output 등으로 출력되는데, 이 시그널 앞에 붙이는 문자열을 의미합니다. 여기서는 'mbus' 이름의 대문자로 "MBUS#" 으로 붙이게 되며, 이를 강제로 'M#' 으로 변경하게 됩니다.

예를 들어 내부 ARADDR 신호의 경우 M0_ARADDR 로 변경됨을 의미합니다. 만약 중복된 종류의 다른 인터페이스가 없다면 'M' 뒤의 숫자 '#'는 제거되어 M_ARADDR 로 정의됩니다.

최종으로 mem_ctrl 모듈에 module:add_interface 함수를 통해 interface 를 붙인 후, module_i:set_port 함수를 이용해 master port 로 선언합니다. 이 인터페이스 역시 상위 모듈에서 언급한 내용이 없으므로 top 의 port 까지 이어집니다.

### 2.4.3 하위 모듈간의 시그널 연결

세번째로 하위 모듈간의 시그널 연결하는 예시(line #42~48, 54)입니다.

```
...
-- add busy signal interface
```

```
core_busy_all   = new_signal("core_busy_all", config.core_size)
core_busy       = new_signal("core_busy")

core.top:add_interface(core_busy):set_port("m")
core_wrapper:add_interface(core_busy_all)
core.busy_ctrl:add_interface(core_busy_all):set_port("s")

for i = 1, config.core_size, 1 do
   ...
   core_inst:set_port("core_busy", "core_busy_all[" .. (i-1) .. "]")
end
...
```

인터페이스가 아닌 시그널 단일 신호의 경우, new_signal 함수로 생성하지만, 내부적으로는 interface 로 구현되어집니다. 이러한 인터페이스를 bared interface 라 칭하며, new signal 함수 형태로 아래와 같이 구현되어 있음을 참고 바랍니다.

```
function new_signal(name, width)
    local   signal  = interface:new(name)

    if width == nil then
        width   = 1
    end

    signal:set_param("WIDTH", width)
    signal:set_signal(name, "WIDTH")
    signal:set_modport("s", {["input" ] = {name}})
    signal:set_modport("m", {["output"] = {name}})
    signal:set_prefix()     -- none prefix
    signal:set_bared()      -- bared signals
    return signal
end
```

추가된 코드를 설명하자면, 4(config.core_size) bit 의 core_busy_all 신호와, core_busy 신호 인터페이스를 추가합니다.

```
core.top:add_interface(core_busy):set_port("m")
```

: 각 코어에 core_busy 를 출력할 수 있도록 core.top 에 core_busy 를 master output으로 선언하고,

```
core.busy_ctrl:add_interface(core_busy_all):set_port("s")
```

: 이를 실제 사용할 busy_ctrl 모듈에 core_busy_all 을 slave input으로 선언합니다.

```
core_wrapper:add_interface(core_busy_all)
```

: 중간에 연결될 부분 core_wrapper 모듈에 동일한 이름으로 interface를 추가하여, busy_ctrl과 연결되도록 합니다.

```
core_inst:set_port("core_busy", "core_busy_all[" .. (i-1) .. "]")
```

: 마지막으로  코어마다 core_busy_all[#] 시그널이 연결되도록 포트 선언을 마치면, 각 core 의 instance 에서 busy_ctrl 모듈쪽으로 자동 연결됩니다.

### 2.4.4 하위 모듈간의 인터페이스 연결

네번째로 하위 모듈간의 인터페이스 연결하는 예시(line #57~65)입니다.

```
...
-- add instruction interface
core.core_if:add_interface(core_i.inst, "if_inst"):set_port("m")
core.core_ex:add_interface(core_i.inst, "if_inst"):set_port("s")

core.core_ex:add_interface(core_i.inst, "ex_inst"):set_port("m")
core.core_wb:add_interface(core_i.inst, "ex_inst"):set_port("s")
...
```

인터페이스 연결은 단일 시그널 연결보다 보다 단순하고 간결합니다. 가능한 인터페이스 단위로 연결을 하시길 바랍니다.

설명을 하자면 인터페이스 core_i.inst 의 instance 인 if_inst 와 ex_inst 를 각각 core_if -> core_ex 로, core_ex -> core_wb 으로 연결한 예입니다. if_inst 하나를 예를 들면 core_if 에 add_interface 로 추가하고 master 포트로 지정합니다. 이와 똑같은 이름을 core_ex 에 생성하고 여기서는 slave 포트로 지정합니다.

이때 두 인터페이스의 중간 연결 지점(core.top)을 찾아 같은 이름의 interface 가 자동으로 선언되어 연결됩니다.

최종 연결 결과는 Figure 2-4과 같이 구성됨을 확인 할 수 있습니다.

**[최종 결과 : test_wrapper_hierarchy.svg]**

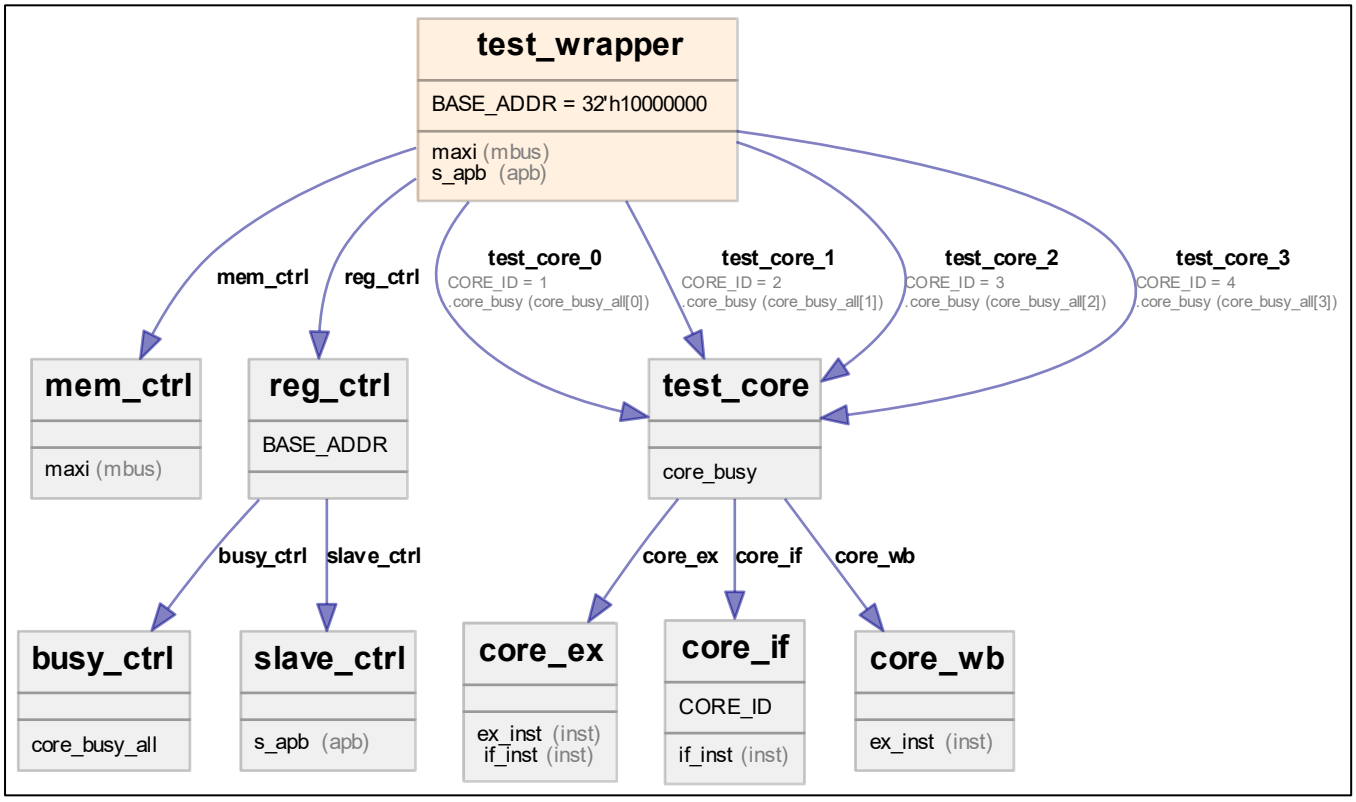

**Figure 2-4. Step #4 Hierarchy Diagram**

아래는 소스 일부인 test_wrapper.sv(top) 와 test_core.sv(core) 를 나열합니다.

**[최종 결과 : test_wrapper.sv]**

```
`include "test_wrapper_defines.vh"

module test_wrapper #(
    parameter BASE_ADDR         = 32'h10000000
) (
    // clock & reset
    input                       ACLK,         // AXI clock
    input                       PCLK,         // APB clock
    input                       PRESETn,      // reset of 'PCLK' (active low)
    input                       nRST,         // default global reset (active low)

    // maxi
    input                       M_AWREADY,
    input                       M_WREADY,
    input                       M_BVALID,
    input  [1:0]                M_BRESP,
    input  [3:0]                M_BID,
    input                       M_ARREADY,
    input                       M_RVALID,
    input  [511:0]              M_RDATA,
    input  [1:0]                M_RRESP,
    input  [3:0]                M_RID,
    input  [3:0]                M_ARQOS,
    input  [3:0]                M_ARREGION,
    output                      M_AWVALID,
    output [35:0]               M_AWADDR,
    output [2:0]                M_AWSIZE,
    output [1:0]                M_AWBURST,
    output [3:0]                M_AWCACHE,
    output [2:0]                M_AWPROT,
    output [3:0]                M_AWID,
    output [7:0]                M_AWLEN,
    output                      M_AWLOCK,
    output                      M_WVALID,
    output                      M_WLAST,
    output [511:0]              M_WDATA,
    output [63:0]               M_WSTRB,
    output [3:0]                M_WID,
    output                      M_BREADY,
    output                      M_ARVALID,
    output [35:0]               M_ARADDR,
    output [2:0]                M_ARSIZE,
    output [1:0]                M_ARBURST,
    output [3:0]                M_ARCACHE,
```

```
    output [2:0]                M_ARPROT,
    output [3:0]                M_ARID,
    output [7:0]                M_ARLEN,
    output                      M_ARLOCK,
    output                      M_RREADY,
    output                      M_RLAST,
    output [3:0]                M_AWQOS,
    output [3:0]                M_AWREGION,

    // s_apb (APB's control bus)
    input                       S_PREADY,
    input  [31:0]               S_PRDATA,
    input                       S_PSLVERR,
    output [1:0]                S_PSEL,
    output                      S_PENABLE,
    output                      S_PWRITE,
    output [15:0]               S_PADDR,
    output                      S_PWDATA
);

// bared interface : core_busy_all
logic [3:0]                     core_busy_all;

// interface : maxi
i_mbus                  maxi;
assign maxi.AWREADY             = M_AWREADY;
assign maxi.WREADY              = M_WREADY;
assign maxi.BVALID              = M_BVALID;
assign maxi.BRESP               = M_BRESP;
assign maxi.BID                 = M_BID;
assign maxi.ARREADY             = M_ARREADY;
assign maxi.RVALID              = M_RVALID;
assign maxi.RDATA               = M_RDATA;
assign maxi.RRESP               = M_RRESP;
assign maxi.RID                 = M_RID;
assign maxi.ARQOS               = M_ARQOS;
assign maxi.ARREGION            = M_ARREGION;
assign M_AWVALID                = maxi.AWVALID;
assign M_AWADDR                 = maxi.AWADDR;
assign M_AWSIZE                 = maxi.AWSIZE;
assign M_AWBURST                = maxi.AWBURST;
assign M_AWCACHE                = maxi.AWCACHE;
assign M_AWPROT                 = maxi.AWPROT;
assign M_AWID                   = maxi.AWID;
assign M_AWLEN                  = maxi.AWLEN;
assign M_AWLOCK                 = maxi.AWLOCK;
assign M_WVALID                 = maxi.WVALID;
assign M_WLAST                  = maxi.WLAST;
assign M_WDATA                  = maxi.WDATA;
assign M_WSTRB                  = maxi.WSTRB;
assign M_WID                    = maxi.WID;
assign M_BREADY                 = maxi.BREADY;
```

```
assign M_ARVALID                = maxi.ARVALID;
assign M_ARADDR                 = maxi.ARADDR;
assign M_ARSIZE                 = maxi.ARSIZE;
assign M_ARBURST                = maxi.ARBURST;
assign M_ARCACHE                = maxi.ARCACHE;
assign M_ARPROT                 = maxi.ARPROT;
assign M_ARID                   = maxi.ARID;
assign M_ARLEN                  = maxi.ARLEN;
assign M_ARLOCK                 = maxi.ARLOCK;
assign M_RREADY                 = maxi.RREADY;
assign M_RLAST                  = maxi.RLAST;
assign M_AWQOS                  = maxi.AWQOS;
assign M_AWREGION               = maxi.AWREGION;

// interface : s_apb (APB's control bus)
i_apb                   s_apb;
assign s_apb.PREADY             = S_PREADY;
assign s_apb.PRDATA             = S_PRDATA;
assign s_apb.PSLVERR            = S_PSLVERR;
assign S_PSEL                   = s_apb.PSEL;
assign S_PENABLE                = s_apb.PENABLE;
assign S_PWRITE                 = s_apb.PWRITE;
assign S_PADDR                  = s_apb.PADDR;
assign S_PWDATA                 = s_apb.PWDATA;

mem_ctrl mem_ctrl (
    .ACLK               (ACLK),
    .nRST               (nRST),
    .maxi               (maxi)
);

reg_ctrl #(
    .BASE_ADDR          (BASE_ADDR)
) reg_ctrl (
    .PCLK               (PCLK),
    .PRESETn            (PRESETn),
    .core_busy_all      (core_busy_all),
    .s_apb              (s_apb)
);

test_core #(
    .CORE_ID            (1)
) test_core_0 (
    .core_busy          (core_busy_all[0])
);

test_core #(
    .CORE_ID            (2)
) test_core_1 (
    .core_busy          (core_busy_all[1])
);

test_core #(
```

```
    .CORE_ID            (3)
) test_core_2 (
    .core_busy          (core_busy_all[2])
);

test_core #(
    .CORE_ID            (4)
) test_core_3 (
    .core_busy          (core_busy_all[3])
);

endmodule
```

**[최종 결과 : test_core.sv]**

```
`include "test_wrapper_defines.vh"

module test_core #(
    parameter CORE_ID          = 0
) (
    // core_busy
    output                     core_busy
);

// interface : ex_inst
i_inst                ex_inst;

// interface : if_inst
i_inst                if_inst;

core_ex core_ex (
    .ex_inst            (ex_inst),
    .if_inst            (if_inst)
);

core_if #(
    .CORE_ID            (CORE_ID)
) core_if (
    .if_inst            (if_inst)
);

core_wb core_wb (
    .ex_inst            (ex_inst)
);

endmodule
```

# 3. Macro functions

코드 구성에 유용한 메크로 함수를 제공합니다.

**Table 3-1. 메크로 함수 요약**

| Macro | Type | Description |
|---|---|---|
| _V | function | 문자열 확장 조작 |
| vfunction | function | verilog 내 사용 함수 선언 |

아래 함수는 verilog에서 사용 가능한 vfunction으로 미리 정의된 함수들입니다. "$함수명(~)"으로 사용할 수 있습니다.

**Table 3-2. 미리 정의된 vfunction 목록**

| vfunction | Type | Description |
|---|---|---|
| LOG2 | function | log2(X) 값을 반환한다. |
| RANGE | function | verilog range 구현 template |
| DEMUX_BY_EN | function | demux 구현 template |
| MULTICYCLE | function | multicycle 구현 template |

## 3.1 _V 메크로

| 함수 원형 | function _V(s, [start], [end], [step]) |
|---|---|
| 반환값 | string |
| 설명 | 문장을 start 에서 end 까지 step 만큼 증가시켜, 문장을 확장합니다.<br>만약 문장이 $(...) 로 구현된 부분이 있다면 해당 부분만, 확장합니다. 없을 경우 전체 문장을 확장하며, 문장내 '#' 문자는 start 에서 end 까지 반복되는 값이 대입됩니다. |
| start | 시작 값 |
| end | 끝 값 (생략할 경우 start 값과 동일하게 취급합니다.) |
| step | 증가 값 (생략할 경우 1또는 -1씩 증감 합니다. end-start 값의 부호에 따라.) |

ex) _V 메크로 예시

```
print(_V("assign A = {$(B[#],)};", 0, 3))
```

[출력]

```
assign A = {B[0], B[1], B[2], B[3]};
```

## 3.2 vfunction 메크로

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function vfunction(name, func) |
| 반환값 | - |
| 설명 | verilog 내에서 "$함수(...)" 로 lua 함수를 호출할 수 있습니다. |
| name | verilog 내에서 사용할 함수 이름. |
| func | verilog에서 사용할 lua function |

**NOTE:** 기본적으로 "_V" 메크로가 vfunction 으로 선언되어 있어서, verilog 내에서 $_V(...) 함수를 동일하게 사용할 수 있습니다.

ex) vfunction 메크로 예시

```
vfunction("RANGE", function(size,step)
    return ("[" .. ((size*(step+1))-1) .. ":" .. (size*step) .. "]")
end)
```

[소스 입력]

```
wire    $RANGE(32,1)    T;
```

[출력]

```
wire    [63:32]     T;
```

## 3.3 $LOG2 함수

| 구분 | 표현식 |
|---|---|
| 함수 원형 | $LOG2(val, [bOverflow]) |
| 반환값 | number |
| 설명 | log2(val) 값을 반환한다. |
| val | log2 입력 값 |
| bOverflow | val 은 2^N 에 해당하는 정수 이어야 합니다.<br>그렇지 않을 경우 에러를 반환합니다.<br>결과 값에 대해 강제로 반올림 처리가 필요할 경우 이 값을 true 로 지정합니다.<br>생략할 경우 false 로 간주합니다. |

ex) $LOG2 예시

```
val_a = 16
```

[소스 입력]

```
localparam      BITS    = $LOG2(val_a);
```

[출력]

```
localparam      BITS    = 4;
```

## 3.4 $DEMUX_BY_EN 함수

| 구분 | 표현식 |
|---|---|
| 함수 원형 | $DEMUX_BY_EN(width, channel_count, en, data_in, data_out) |
| 반환값 | string |
| 설명 | demux_by_enable 모듈을 이용하여, demux를 구현합니다. |
| width | 1개 data 당 bitwidth |
| channel_count | 입력 채널 개수 |
| en | 입력 enable 신호(string) |
| data_in | 입력 데이터 (총 channel_count 개수 만큼, string) |
| data_out | 출력 데이터 (string) |

ex) $DEMUX_BY_EN 예시

[소스 입력]

```
wire    [31:0]      a,b,c,d;
wire    [3:0]       en;
wire    [31:0]      odata;

$DEMUX_BY_EN(32, 4, "en", "{a,b,c,d}", "odata")
```

[출력]

```
wire    [31:0]      a,b,c,d;
wire    [3:0]       en;
wire    [31:0]      odata;

demux_by_enable #(
    .WIDTH              (32),
    .CHANNELS           (4),
    .TRISTATE           (1)
) demux_en_pc (
    .EN_BUS             (en),
    .DIN_BUS            ({a,b,c,d}),
    .DOUT               (odata)
);
```

## 3.5 $MULTICYCLE 함수

| 구분 | 표현식 |
|---|---|
| 함수 원형 | $MULTICYCLE(module_inst_name, if_name, cycle_count, [instance_count], [clk]) |
| 반환값 | string |
| 설명 | multicycle 구현 template "MultiCyclePath" 모듈 또는 "MultiCyclePathEx"를 이용하여, interface 1개인 모듈을 multicycle로 구현한다. |
| module_inst_name | 현재 모듈의 하위에 포함된 모듈 이름 |
| if_name | module_inst_name 의 모듈에 대응하는 interface instance 명을 지정합니다. |
| cycle_count | cycle 수 (2 ≤ cycle_count ≤ 12) |
| instance_count | 하위 모듈의 instance 수 (1 ≤ instance_count ≤ cycle_count)<br>생략할 경우 cycle_count 와 동일한 수로 간주한다. |
| clk | multicycle 구현에 사용할 클럭<br>생략할 경우 default clock 을 사용한다. (clock:set_default() 함수 참조) |

ex) $MULTICYCLE 예시

[소스 입력]

```
$MULTICYCLE("MTSP_Synchronize", "mtsp_sync", 2, 1)
```

[출력]

```
genvar i;
// multicycle design for MTSP_Synchronize
i_mtsp_sync mtsp_sync();
wire    mtsp_sync_ie, mtsp_sync_oe, mtsp_sync_iready;
generate
wire    [7:0]   pipe_i;
wire    [1:0]   pipe_o;
wire    [1:0]   __o;

MultiCyclePathEx #(
    .IWIDTH     (8),
    .OWIDTH     (2),
    .CYCLE      (2),
    .COUNT      (1)
) multi_pipe (
    .CLK        (MCLK),
    .nRST       (nRST),
    .IE         (mtsp_sync_ie),
    .IDATA      ({mtsp_sync.sync, mtsp_sync.eop}),
    .IREADY     ({mtsp_sync_iready}),
```

```
    .PIPE_I     (pipe_i),
    .PIPE_O     (pipe_o),
    .OE         (mtsp_sync_oe),
    .ODATA      (__o)
);

assign  {mtsp_sync.awake, mtsp_sync.done}   = __o;

for(i=0; i<1; i=i+1) begin
    i_mtsp_sync     __temp;
    assign  {__temp.sync, __temp.eop}   = pipe_i[`BUS_RANGE(8, i)];
    assign  pipe_o[`BUS_RANGE(2,i)] = {__temp.awake, __temp.done};

    MTSP_Synchronize MTSP_Synchronize (
        .mtsp_sync          (__temp)
    );
end
endgenerate
```

# 4. Class and Method

아래와 같이 총 3개의 객체 종류가 존재합니다.

- clock
  - 클럭을 생성하고, interface 에 할당할 수 있습니다. 해당 interface 가 사용될 때, 자동으로 할당된 클럭과 클럭에 매칭되는 reset 이 port 로 선언됩니다. 만약 reset 이 선언되어 있지 않는다면, 자동으로 기본 리셋 nRST 신호가 생성됩니다.
- interface
  - module 에서 사용할 interface 를 생성합니다. interface 는 다른 interface 로 부터 상속을 받아 생성될 수 있습니다. 모듈에 add_interface 로 부탁된 instance 는 interface_i:* 함수만 호출 가능합니다.
- module
  - 모듈을 생성하고, module:add_module 함수를 통해 다른 하위 모듈을 포함하거나, module:add_interface 를 호출하여 interface instance 를 생성할 수 있습니다.

## 4.1 clock

interface 에 할당할 클럭을 생성하거나 관리합니다. 할당된 클럭은 해당 interface 가 사용될 때 자동으로 module의 port에 따라 선언되게 됩니다.

또한 contraint 에 해당 클럭의 speed 가 정의되며, 이종 클럭간의 register 들과 클럭에 설정된 reset 에 대해서도 false_path 를 자동으로 지정하게 됩니다.

**Table 4-1. clock 객체 요약**

| Member | Type | Description |
|---|---|---|
| .name | string | clock 이름 |
| :new | function | clock 생성 |
| :set_reset | function | clock에 reset 설정 |
| :get_reset | function | clock에 할당된 reset 을 반환 |
| :set_speed | function | clock에 동작속도 설정 |
| :set_default | function | 현 클럭을 기본 클럭으로 설정한다. |
| .find | function | clock 찾기 |
| .is_valid | function | clock 여부 확인 |
| .get_default | function | 기본 클럭을 얻는다. |

### 4.1.1 clock:new

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock:new(name, [desc]) |
| 반환값 | clock |
| 설명 | name 이름을 가지는 clock을 생성합니다. |
| name | clock 이름. |
| desc | clock 설명, 주석에 사용된다. (생략 가능) |

ex) 생성 예시

```
new_clock   = clock:new("CLK")          -- 기본 클럭으로 부터 생성
new_clock:set_reset("GRSTn")

new_clock2  = new_clock:new("ACLK")     -- new_clock 을 복제한 클럭. reset, speed 를 상속 받는다.
```

### 4.1.2 clock:set_reset

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock:set_reset(name) |
| 반환값 | - |
| 설명 | clock에 name 이름의 reset 을 생성합니다.<br>이 함수를 사용하여 reset 을 생성하지 않을 경우, 기본 리셋 'nRST' 신호가 자동으로 사용된다. |
| name | reset 이름. (active low) |

ex) reset 지정 예시

```
aclock  = clock:new("ACLK")
aclock:set_reset("ARSTn")                -- 리셋 ARSTn 지정
```

### 4.1.3 clock:get_reset

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock:get_reset() |
| 반환값 | string |
| 설명 | clock에 할당된 reset을 반환한다.<br>할당된 reset 이 없다면, 기본 reset 신호가 반환된다. |

### 4.1.4 clock:set_speed

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock:set_speed(mhz) |
| 반환값 | - |
| 설명 | clock의 동작속도를 지정합니다.<br>이 함수를 사용하여 지정하지 않을 경우, 기본 클럭 100MHz 가 설정됩니다. |
| mhz | 지정할 동작 속도 값. (MHz 단위) |

ex) 동작 속도 지정 예시

```
aclock  = clock:new("ACLK")
aclock:set_speed(1000)                    -- ACK 에 1GHz 설정
```

### 4.1.5 clock:set_default

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock:set_default() |
| 반환값 | - |
| 설명 | 현재 클럭을 기본 클럭으로 설정한다.<br>처음 생성하는 클럭을 기본 클럭으로 설정하며, 별도로 특정 클럭을 명시적으로 클럭으로 설정할 때 사용한다. |

ex) 기본클럭 설정 예시

```
aclock  = clock:new("MCLK")
aclock:set_default()
```

### 4.1.6 clock.find

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock.find(name) |
| 반환값 | clock |
| 설명 | clock을 찾습니다.<br>찾지 못할 경우 nil 값을 반환합니다. |
| name | 찾을 clock 이름. |

ex) clock 찾기 예시

```
aclock  = clock.find("ACLK")            -- ACLK 찾기

if aclock ~= nil then
    LOGI("ACLK is found.")              -- 찾았음.
end
```

### 4.1.7 clock.is_valid

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock.is_valid(obj) |
| 반환값 | boolean |
| 설명 | clock 객체가 맞는지 확인합니다. |
| obj | 확인할 clock 객체. |

ex) clock 객체 확인 예시

```
aclock  = clock:new("ACLK")

if clock.is_valid(aclock) then
    LOGI("aclock is clock object.")      -- clock 객체가 맞음.
end
```

### 4.1.8 clock.get_default

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function clock.get_default() |
| 반환값 | clock |
| 설명 | 기본 클럭을 반환한다. |

## 4.2 interface

interface 객체는 systemverilog 의 interface 기술과 동일하게 작동합니다. systemverilog 의 interface 문법을 살펴보면 아래와 같습니다.

**[systemverilog interface 선언]**

```
interface my_intface;
    logic          a;
    logic   [3:0]  b;

    // modport example
    modport s (input a, output b);      // slave modport
    modport m (input a, input  b);      // master modport
endinterface
```

**NOTE:** systemverilog interface 의 상세한 설명은 외부링크 systemverilog modport 설명을 참조 바랍니다.

그 중 합성 가능한 modport 기능을 사용하여, port 구성을 시도하며, 기본 객체인 interface 와 module 객체에 add_interface 를 통해 생성된 interface_i 객체로 크게 나뉩니다.

**Table 4-2. interface 객체 요약**

| Member | Type | Description |
|---|---|---|
| .name | string | 인테페이스 이름 |
| :new | function | 인테페이스 생성 |
| new_signal | function | 단일 시그널 형태의 인터페이스 생성 |
| .find | function | 인터페이스 찾기 |
| .is_valid | function | 인터페이스 여부 확인 |
| :set_clock | function | 클럭 할당 |
| :get_clock | function | 클럭 얻기 |
| :set_signal | function | 신호 추가 |
| :signal_count | function | 신호 개수 얻기 |
| :set_param | function | parameter 추가 |
| :get_param | function | parameter 검색 |
| :set_modport | function | modport 설정 |
| :add_modport | function | modport 에 추가 |
| :get_modport | function | modport 찾기 |
| :set_prefix | function | port 출력시 prefix 지정 |
| :set_bared | function | interface 를 signal로 풀어서 적용 |

**Table 4-3. interface_i 객체 요약**

| :set_port | function | 인스턴스를 port로 지정 |
|---|---|---|
| :set_desc | function | 인스턴스의 설명 추가 |
| :set_prefix | function | 인스턴스의 prefix 지정 |
| :get_prefix | function | 인스턴스의 prefix 얻기 |

### 4.2.1 interface:new

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:new(name) |
| 반환값 | interface |
| 설명 | name 이름을 가지는 interface를 생성합니다.<br>생성할 때 기본 prefix 는 ('name 대문자' + '#') 로 지정됩니다. |
| name | interface 이름. |

ex) interface 생성 예시

```
i_apb       = interface:new("APB")       -- APB interface 생성
i_apb:set_signal("RARDDR", 32)
```

### 4.2.2 new_signal

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function new_signal(name, [width]) |
| 반환값 | interface |
| 설명 | name 이름을 가지는 bared interface를 생성합니다. |
| name | signal 이름. |
| width | 시그널 bitwidth. 생략할 경우 1로 지정됩니다. |

내부의 실제 구현은 아래와 같이 bared interface 를 생성하고, modport 's'는 input으로 modport 'm'은 output으로 설정합니다. 또한 bared interface 이기 때문에 [top_module]_include.vh 헤더에도 interface 로 기록되지 않습니다.

```
function new_signal(name, width)
    local   signal  = interface:new(name)

    if width == nil then
        width   = 1
    end

    signal:set_param("WIDTH", width)
    signal:set_signal(name, "WIDTH")
    signal:set_modport("s", {["input" ] = {name}})
    signal:set_modport("m", {["output"] = {name}})
    signal:set_prefix()     -- none prefix
    signal:set_bared()      -- bared signals
    return signal
end
```

ex) signal 생성 예시

```
s_BUSY      = new_signal("BUSY_ALL", 4)
```

### 4.2.3 interface.find

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface.find(name) |
| 반환값 | interface |
| 설명 | 생성된 interface 를 찾습니다. |
| name | 찾을 interface 이름 |

ex) interface 찾기 예시

```
i_APB       = interface:new("APB")

if interface.find("APB") ~= nil then
    LOGI("APB interface is existed.")
end
```

### 4.2.4 interface.is_valid

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface.is_valid(obj) |
| 반환값 | boolean |
| 설명 | interface 객체가 맞는지 확인합니다. |
| obj | 확인할 클럭 객체. |

ex) interface 객체 확인 예시

```
i_APB   = interface:new("APB")

if interface.is_valid(i_APB) then
    LOGI("i_APB is interface object.")      -- interface 객체가 맞음.
end
```

### 4.2.5 interface:set_clock

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:set_clock(clk) |
| 반환값 | - |
| 설명 | interface에 clock을 지정합니다. |
| clk | clock 객체 |

ex) interface 객체에 클럭 설정 예시

```
i_APB   = interface:new("APB")

PCLK    = clock:new("PCLK", "APB's clock")
PCLK:set_reset("PRSTn")

i_APB:set_clock(PCLK)           -- PCLK 설정
```

### 4.2.6 interface:get_clock

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:get_clock() |
| 반환값 | clock |
| 설명 | interface에서 할당된 clock을 반환합니다. |

ex) interface 객체에 클럭 설정 예시

```
i_APB   = interface:new("APB")

PCLK    = clock:new("PCLK", "APB's clock")
PCLK:set_reset("PRSTn")

i_APB:set_clock(PCLK)

LOGI("APB's clock is " .. i_APB:get_clock().name)    -- 클럭 이름 출력
```

### 4.2.7 interface:set_signal

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:set_signal(name, [bit_width]) |
| 반환값 | - |
| 설명 | interface에 signal 을 설정 또는 변경합니다. |
| name | 설정할 signal 이름 |
| bit_width | signal 의 bit width. 만약 설정하지 않으면 1로 간주합니다. 또한 명시적으로 0으로 설정할 경우 해당 시그널은 사용하지 않습니다. (상수 이외에 parameter 값이나 수식을 사용할 수 있습니다.) |

ex) interface 객체에 signal 추가 예시

```
i_axi3  = interface:new("AXI3")

-- parameter 설정
i_axi3:set_param("ADDR_WIDTH", 16)
i_axi3:set_param("DATA_WIDTH", 128)

-- signal 설정
i_axi3:set_signal("AWVALID")
i_axi3:set_signal("AWREADY")
i_axi3:set_signal("AWADDR", "ADDR_WIDTH")
i_axi3:set_signal("AWSIZE", 3)
i_axi3:set_signal("AWBURST", 2)
i_axi3:set_signal("WSTRB", "DATA_WIDTH/8")
```

### 4.2.8 interface:signal_count

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:signal_count() |
| 반환값 | number |
| 설명 | interface에 정의된 signal 개수를 반환합니다. |

### 4.2.9 interface:set_param

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:set_param(name, default_value) |
| 반환값 | - |
| 설명 | interface에 parameter를 추가 또는 변경합니다. |
| name | parameter 이름 |
| default_value | parameter 기본 값. (상수 또는 수식등이 포함될 수 있습니다.) |

ex) interface 객체에 parameter 추가 예시

```
i_axi3  = interface:new("AXI3")

-- parameter 설정
i_axi3:set_param("ADDR_WIDTH", 16)
i_axi3:set_param("DATA_WIDTH", 128)

-- prameter 수정
i_axi3:set_param("DATA_WIDTH", 256)
```

### 4.2.10 interface:get_param

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:get_param(name) |
| 반환값 | number 또는 string |
| 설명 | interface에 parameter를 반환합니다. |
| name | parameter 이름 |

ex) interface 객체에 parameter 얻기 예시

```
i_axi3  = interface:new("AXI3")

-- parameter 설정
i_axi3:set_param("ADDR_WIDTH", 16)
i_axi3:set_param("DATA_WIDTH", 128)

-- parameter 얻어 출력하기
LOGI("i_axi3's data width = " .. tostring(i_axi3:get_param("DATA_WIDTH")))
```

### 4.2.11 interface:set_modport

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function interface:set_modport(name, modport) |
| 반환값 | - |
| 설명 | interface에 modport를 추가합니다. |
| name | modport 이름 |
| modport | modport 구성 table 구조체 이며, 아래와 같은 형태로 기술합니다.<br>{["input"]={"", ...}, ["output"]={"", ...}, ["inout"]={"", ...}} |

ex) set_modport 예시

```
-- APB bus
bus_apb     = interface:new("apb")
bus_apb:set_param("ADDR_WIDTH", 16)
bus_apb:set_param("DATA_WIDTH", 32)
bus_apb:set_param("SEL_WIDTH", 2)
bus_apb:set_signal("PADDR", "ADDR_WIDTH")
bus_apb:set_signal("PSEL", "SEL_WIDTH")
bus_apb:set_signal("PENABLE")
bus_apb:set_signal("PWRITE")
bus_apb:set_signal("PWDATA", "DATA_WIDTH")
bus_apb:set_signal("PREADY")
bus_apb:set_signal("PRDATA", "DATA_WIDTH")
bus_apb:set_signal("PSLVERR")

bus_apb:set_modport("s", {["input" ]={"PSEL", "PENABLE", "PWRITE", "PADDR", "PWDATA"},
["output"]={"PREADY", "PRDATA", "PSLVERR"}})
bus_apb:set_modport("m", {["output"]={"PSEL", "PENABLE", "PWRITE", "PADDR", "PWDATA"},
["input" ]={"PREADY", "PRDATA", "PSLVERR"}})
```

### 4.2.12 interface:add_modport

| 함수 원형 | function interface:add_modport(name, modport) |
|---|---|
| 반환값 | - |
| 설명 | interface의 기존 modport에 signal을 추가합니다. |
| name | modport 이름 |
| modport | modport 구성 table 구조체 이며, 아래와 같은 형태로 기술합니다.<br>{["input"]={"", ...}, ["output"]={"", ...}, ["inout"]={"", ...}} |

ex) add_modport 예시

```
-- APB bus
bus_apb     = interface:new("apb")
bus_apb:set_param("ADDR_WIDTH", 16)
bus_apb:set_param("DATA_WIDTH", 32)
bus_apb:set_param("SEL_WIDTH", 2)
bus_apb:set_signal("PADDR", "ADDR_WIDTH")
bus_apb:set_signal("PSEL", "SEL_WIDTH")
bus_apb:set_signal("PENABLE")
bus_apb:set_signal("PWRITE")
bus_apb:set_signal("PWDATA", "DATA_WIDTH")
bus_apb:set_signal("PREADY")
bus_apb:set_signal("PRDATA", "DATA_WIDTH")
bus_apb:set_signal("PSLVERR")

bus_apb:set_modport("s", {["input" ]={"PSEL", "PENABLE", "PWRITE"}, ["output"]={"PREADY",
"PRDATA", "PSLVERR"}})
bus_apb:set_modport("m", {["output"]={"PSEL", "PENABLE", "PWRITE"}, ["input" ]={"PREADY",
"PRDATA", "PSLVERR"}})

bus_apb:add_modport('s', {["input" ]={"PADDR", "PWDATA"}})
bus_apb:add_modport('m', {["output"]={"PADDR", "PWDATA"}})
```

### 4.2.13 interface:get_modport

| 함수 원형 | function interface:get_modport(name, modport) |
|---|---|
| 반환값 | table |
| 설명 | interface의 기존 modport의 테이블을 반환합니다. |
| name | modport 이름 |

ex) add_modport 예시

```
-- APB bus
bus_apb     = interface:new("apb")
bus_apb:set_param("ADDR_WIDTH", 16)
bus_apb:set_param("DATA_WIDTH", 32)
bus_apb:set_param("SEL_WIDTH", 2)
bus_apb:set_signal("PADDR", "ADDR_WIDTH")
bus_apb:set_signal("PSEL", "SEL_WIDTH")
bus_apb:set_signal("PENABLE")
bus_apb:set_signal("PWRITE")
bus_apb:set_signal("PWDATA", "DATA_WIDTH")
bus_apb:set_signal("PREADY")
bus_apb:set_signal("PRDATA", "DATA_WIDTH")
bus_apb:set_signal("PSLVERR")

bus_apb:set_modport("s", {["input" ]={"PSEL", "PENABLE", "PWRITE"}, ["output"]={"PREADY",
"PRDATA", "PSLVERR"}})
bus_apb:set_modport("m", {["output"]={"PSEL", "PENABLE", "PWRITE"}, ["input" ]={"PREADY",
"PRDATA", "PSLVERR"}})

-- modeport 's' 의 'input' 나열
for i, signal_name in ipairs(bus_apb:get_modport("s").input) do
    LOGI("modport 's' input : " .. signal_name)
end
```

### 4.2.14 interface:set_prefix

| 함수 원형 | function interface:set_prefix(prefix) |
|---|---|
| 반환값 | - |
| 설명 | interface의 port 출력시 prefix 문자열을 지정합니다.<br>여러 개의 동일 interface 가 동시에 같은 module 내 port 출력이 되어 있다면,'#' 문자를 prefix 에 포함하여, 숫자 0 부터 1씩 증가하도록 변경됩니다. 만약 단일 interface 라면 문자 '#' 는 제거됩니다. |
| prefix | prefix 문자열 |

ex) set_prefix 예시

```
-- interface 예제
inst    = interface:new("inst")
inst:set_signal("EN")
inst:set_signal("INST", 32)
inst:set_modport("s", {["input" ]={"EN", "INST"}})
inst:set_modport("m", {["output"]={"EN", "INST"}})

inst:set_prefix("I#")   -- 시그널에 I#_* 로 시작하게 된다.

m       = module:new("top")
m:add_interface(inst, "inst_0", "m")
m:add_interface(inst, "inst_1", "m")

m:make_code()
```

**[실행 결과 : top_defines.vh]**

```
 1: `ifndef __TOP_DEFINES_VH__
 2: `define __TOP_DEFINES_VH__
 3: `include "testdrive_system.vh"      // default system defines
 4:
 5: //-----------------------------------------
 6: // interfaces
 7: //-----------------------------------------
 8: interface i_inst;
 9:     logic               EN;
10:     logic [31:0]        INST;
11:     modport m (
12:         output  EN, INST
13:     );
14:     modport s (
15:         input   EN, INST
16:     );
17: endinterface
18:
```

```
19: `endif//__TOP_DEFINES_VH__
```

**[실행 결과 : top.sv]**

```
 1: `include "top_defines.vh"
 2:
 3: module top (
 4:     // inst_0
 5:     output                  IO_EN,
 6:     output [31:0]           IO_INST,
 7:
 8:     // inst_1
 9:     output                  I1_EN,
10:     output [31:0]           I1_INST
11: );
12:
13: // interface : inst_0
14: i_inst                inst_0;
15: assign IO_EN                  = inst_0.EN;
16: assign IO_INST                = inst_0.INST;
17:
18: // interface : inst_1
19: i_inst                inst_1;
20: assign I1_EN                  = inst_1.EN;
21: assign I1_INST                = inst_1.INST;
22:
23:
24: endmodule
```

### 4.2.15 interface:set_bared

| 함수 원형 | function interface:set_bared(bared) |
|---|---|
| 반환값 | - |
| 설명 | interface의 구조체를 풀어 적용합니다. |
| bared | boolean 값의 bared 여부, 지정하지 않을 경우 true 로 지정됩니다. |

bared signal 구성시 사용됩니다.

ex) set_bared 예시

```
-- interface 예제
inst    = interface:new("inst")
inst:set_signal("EN")
inst:set_signal("INST", 32)
inst:set_modport("s", {["input" ]={"EN", "INST"}})
inst:set_modport("m", {["output"]={"EN", "INST"}})

inst:set_bared()            -- bared interface 설정
```

#### 4.2.16 interface:set_top_uppercase

| 함수 원형 | function interface:set_top_uppercase(en) |
|---|---|
| 반환값 | - |
| 설명 | interface 의 top 의 port 출력시 강제 대문자 또는 소문자 이름으로 강제합니다. |
| en | uppercase 여부, true(uppercase), false(lowercase), nil(원본) |

### 4.2.17 interface_i:set_port

| 함수 원형 | function interface_i:set_port(modport_name) |
|---|---|
| 반환값 | - |
| 설명 | module:add_interface 함수로 추가된 interface instance 를 port 출력으로 지정합니다. |
| modport_name | modport 이름 |

기본적인 interface 를 module에 추가하였을 때, 이 함수를 통해 port 출력(input, output, inout)이 결정되게 됩니다.

ex) interface_i:set_port 예시

```
-- interface 예제
inst    = interface:new("inst")
inst:set_signal("EN")
inst:set_signal("INST", 32)
inst:set_modport("s", {["input" ]={"EN", "INST"}})
inst:set_modport("m", {["output"]={"EN", "INST"}})

top         = module:new("top")

top:add_interface(inst):set_port("m")    -- inst interface 를 modport 'm' 으로 top 출력 지정
```

### 4.2.18 interface_i:set_desc

| 함수 원형 | function interface_i:set_desc(desc) |
|---|---|
| 반환값 | - |
| 설명 | module:add_interface 함수로 추가된 interface instance 의 주석으로 쓰일 설명을 추가합니다. |
| desc | 추가 설명 문자열 |

ex) interface_i:set_desc 예시

```
-- interface 예제
inst    = interface:new("inst")
inst:set_signal("EN")
inst:set_signal("INST", 32)
inst:set_modport("s", {["input" ]={"EN", "INST"}})
inst:set_modport("m", {["output"]={"EN", "INST"}})

top         = module:new("top")

i_int = top:add_interface(inst)
i_int:set_port("m") -- inst interface 를 modport 'm' 으로 top 출력 지정
i_int:set_desc("main instruction")  -- 주석 설명
```

### 4.2.19 interface_i:set_prefix

| 함수 원형 | function interface_i:set_prefix(prefix) |
|---|---|
| 반환값 | - |
| 설명 | module:add_interface 함수로 추가된 interface instance 의 prefix를 지정합니다.<br>지정되지 않았을 경우 원본 interface 의 prefix 가 사용됩니다. |
| prefix | prefix 문자열 |

ex) interface_i:set_prefix 예시

```
-- interface 예제
inst    = interface:new("inst")
inst:set_signal("EN")
inst:set_signal("INST", 32)
inst:set_modport("s", {["input" ]={"EN", "INST"}})
inst:set_modport("m", {["output"]={"EN", "INST"}})

top         = module:new("top")

i_int = top:add_interface(inst)
i_int:set_port("m")     -- inst interface 를 modport 'm' 으로 top 출력 지정
i_int:set_prefix("IF")  -- prefix 지정
```

이 interface instance 는 IF_EN, IF_INST 로 전환되어 port 로 출력됩니다.

### 4.2.20 interface_i:get_prefix

| 함수 원형 | function interface_i:get_prefix() |
|---|---|
| 반환값 | string |
| 설명 | module:add_interface 함수로 추가된 interface instance 의 prefix를 반환합니다. 만약 interface instance 에서 prefix 가 지정되지 않았다면, 원본 interface 의 prefix 를 반환합니다. |

ex) interface_i:get_prefix 예시

```
-- interface 예제
inst    = interface:new("inst")
inst:set_signal("EN")
inst:set_signal("INST", 32)
inst:set_modport("s", {["input" ]={"EN", "INST"}})
inst:set_modport("m", {["output"]={"EN", "INST"}})

top         = module:new("top")

i_int = top:add_interface(inst)
i_int:set_port("m")     -- inst interface 를 modport 'm' 으로 top 출력 지정

-- 기본 설정된 prefix 출력
LOGI("PREFIX : " .. i_int:get_prefix())
```

## 4.3 module

verilog module 선언과 매칭되는 객체입니다. module:new 함수로 생성되며, module:make_code 함수를 통해 최종 결과 소스를 출력합니다. 이때 하위에 포함된 sub module 들과 한번이라도 사용된 interface 들의 선언도 함께 이루워집니다.

top module 의 port 는 systemverilog 의 interface 문법이 아닌 input/output 형태의 단일 signal들로 전환되어 출력되며, 내부 sub module 들은 interface 문법에 따라 기술됩니다.

module:add_module함수로 추가된 sub module 객체는 module_i 인터페이스로 사용하게 됩니다.

**Table 4-4. module 객체 요약**

| Member | Type | Description |
|---|---|---|
| .name | string | module 이름 |
| :new | function | module 생성 |
| :set_inception | function | code inception 파일을 지정합니다. |
| :get_inception | function | code inception을 반환합니다. |
| :set_title | function | code inception의 title 을 지정합니다. |
| :set_author | function | code inception의 author 를 지정합니다. |
| :set_param | function | parameter를 지정합니다. |
| :get_param | function | parameter를 검색합니다. |
| :add_interface | function | interface를 추가합니다. |
| :get_interface | function | 추가된 interface를 검색합니다. |
| :get_port | function | 추가된 interface 중 port를 검색합니다. |
| :add_module | function | sub module을 추가합니다. |
| :get_module | function | sub module을 검색합니다. |
| :add_code | function | 사용자 코드 문장을 추가합니다. |
| .find | function | module 객체를 찾습니다. |
| .is_valid | function | module 여부를 확인합니다. |
| .apply_code | function | code 파일을 module 에 적용합니다. |
| .code | String | module 의 추가된 코드 문자열 객체 |

**Table 4-5. module_i(sub module) 객체 요약**

| .name | string | sub module 이름을 반환합니다. |
|---|---|---|
| :set_param | function | paramter의 값을 지정합니다. |
| :get_param | function | paramter의 값을 반환합니다. |
| :set_port | function | port의 값을 지정합니다. |
| :get_port | function | port의 값을 반환합니다. |
| .is_valid | function | module_i 객체 여부를 판별합니다. |

### 4.3.1 module:new

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module:new(name) |
| 반환값 | module |
| 설명 | module을 생성합니다. |
| name | module 이름 |

ex) module 생성 예시

```
top         = module:new("top")      -- 모듈 생성 예
```

### 4.3.2 module:make_code

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module:make_code() |
| 반환값 | - |
| 설명 | 모듈을 최종 결과 파일로 생성합니다. 생성 목록은 다음과 같습니다.<br>[top_name]_defines.vh (define 과 interface 선언들)<br>[사용된 모든 module 명].sv (결과 systemverilog 소스 파일들)<br>[top_name].f (모든 참조 소스명)<br>[top_name]_constraint.xdc (constraint 선언)<br>[top_name]_hierarchy.svg (design hierarchy diagram) |

### 4.3.3 module:set_inception

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module:set_inception(filename) |
| 반환값 | - |
| 설명 | code inception 이 기술된 파일을 지정합니다.<br>code inception 기술에 아래와 같은 메타 문장을 사용할 수 있습니다.<br>__YEAR__ : 현재 연도. 예) 2023<br>__DATE__ : 현재 날짜. 예) May/08/2023 Mon<br>__TITME__ : 현재 시간. 예) 19:34:21<br>__AUTHOR__ : module:set_author 함수로 지정한 author. 기본값 : "testdrive profiling master - verigen"<br>__TITLE__ : module:set_title 함수로 지정한 타이틀. 기본값 : "no_title" |
| filename | code inception 이 기술된 파일명 |
| bit_width | signal 의 bit width. 만약 설정하지 않으면 1로 간주합니다. 또한 명시적으로 0으로 설정할 경우 해당 시그널은 사용하지 않습니다. (상수 이외에 parameter 값이나 수식을 사용할 수 있습니다.) |

이 inception 문구는 각 .sv 소스의 상단에 위치하게 됩니다. 각 module 마다 별도로 :set_inception, :set_title, :set_author 함수로 지정하여 별도로 license 와 같은 문장을 삽입할 수 있습니다.

만약 module:set_inception 으로 호출하면, 모든 생성 모듈에서 별도로 지정하지 않는 한 기본 모듈의 code inception을 사용하게 됩니다.

ex) set_inception 예시

```
-- code inception 설정
module:set_inception("code_inception.txt")
module:set_title("some title")
module:set_author("me")

top         = module:new("top")

top:make_code()
```

**[code_inception.txt]**

```
//=================================================================================
// Copyright (c) 2013 ~ __YEAR__. HyungKi Jeong(clonextop@gmail.com)
// Freely available under the terms of the 3-Clause BSD License
// (https://opensource.org/licenses/BSD-3-Clause)
```

```
//
// Title : __TITLE__
// Rev.  : __DATE__ __TIME__ (__AUTHOR__)
//================================================================================
```

실행 결과

**[결과 파일 : top.sv]**

```
//================================================================================
// Copyright (c) 2013 ~ 2023. HyungKi Jeong(clonextop@gmail.com)
// Freely available under the terms of the 3-Clause BSD License
// (https://opensource.org/licenses/BSD-3-Clause)
//
// Title : some title
// Rev.  : May/09/2023 Tue 14:10:16 (me)
//================================================================================
`include "top_defines.vh"

module top ();

endmodule
```

### 4.3.4 module:get_inception

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module:get_inception() |
| 반환값 | - |
| 설명 | module:set_inception 으로 설정한 code inception 문구에 모든 메타 문장을 적용한 결과를 반환합니다. |

#### 4.3.5 module:set_title

| 함수 원형 | function module:set_title(title) |
|---|---|
| 반환값 | - |
| 설명 | code inception의 title을 지정합니다.<br>code inception 내 __TITLE__ 메타 문장이 지정된 title로 변환됩니다. |
| title | title 문자열 |

### 4.3.6 module:set_author

| 함수 원형 | function module:set_author(name) |
|---|---|
| 반환값 | - |
| 설명 | code inception의 author 를 지정합니다.<br>code inception 내 __AUTHOR__ 메타 문장이 지정된 author로 변환됩니다. |
| name | author 문자열 |

### 4.3.7 module:set_param

| 함수 원형 | function module:set_param(name, value, [is_local]) |
|---|---|
| 반환값 | - |
| 설명 | module의 parameter를 추가합니다. |
| name | parameter 이름 |
| value | parameter 기본값 |
| is_local | true 일 경우 localparam으로 구현되고, 아니면 port의 parameter로 구현됩니다. 생략할 경우 기본값 false 입니다. |

ex) module:set_param 예시

```
top = module:new("top")      -- 모듈 생성 예

-- port parameter 설정
top:set_param("DATA_WIDTH", 32)

-- local parameter 설정
top:set_param("BYTE_WIDTH", "DATA_WIDTH/8", true)
```

### 4.3.8 module:get_param

| 함수 원형 | function module:get_param(name) |
|---|---|
| 반환값 | integer 또는 string |
| 설명 | module의 parameter의 기본값을 반환합니다. |
| name | parameter 이름 |

### 4.3.9 module:add_interface

| 함수 원형 | function module:add_interface(i, [name], [modport]) |
|---|---|
| 반환값 | interface_i |
| 설명 | 모듈에 interface instance 객체를 추가합니다. |
| i | interface 객체 |
| name | interface instance 명<br>이름을 지정하지 않을 경우 인터페이스 명을 따른다. |
| modport | modport 이름, port가 아닌 내부 선언용으로 사용할 경우.<br>이 값을 지정하지 않는다. |

### 4.3.10 module:get_interface

| 함수 원형 | function module:get_interface(name) |
|---|---|
| 반환값 | interface_i |
| 설명 | 모듈에 추가된 interface instance 객체를 검색하여 반환합니다. |
| name | interface instance 객체 이름 |

#### 4.3.11 module:get_port

| 함수 원형 | function module:get_port(name) |
|---|---|
| 반환값 | interface_i |
| 설명 | 모듈에 추가된 interface instance 객체 중, port로 설정된 객체를 검색하여 반환합니다. |
| name | interface instance 객체 이름 |

#### 4.3.12 module:add_module

| 함수 원형 | function module:add_module(m, [name]) |
|---|---|
| 반환값 | - |
| 설명 | sub module을 추가합니다. |
| m | sub module이 될 module 원본 |
| name | sub module 이름.<br>생략할 경우, 원본 모듈 이름과 같게 지어집니다.<br>생략한 여러 개의 같은 이름의 sub module 이 존재할 경우, 각 이름 뒤에 "_#" 형태로 번호를 붙여 중복하지 않도록 합니다. |

#### 4.3.13 module:get_module

| 함수 원형 | function module:get_module(name) |
|---|---|
| 반환값 | module |
| 설명 | sub module을 검색하여 반환합니다. |
| name | sub module 이름 |

#### 4.3.14 module:add_code

| 함수 원형 | function module:add_code(s) |
|---|---|
| 반환값 | - |
| 설명 | 사용자 코드 추가합니다.<br>이 코드들은 각 모듈 소스의 맨 마지막에 삽입되는 문장으로 포함됩니다.<br>module.code(String) 객체에 추가되며, _V() 메크로 함수와 함께 유용하게 사용될 수 있습니다. 추가된 코드의 마지막 문자가 ';' 로 끝날 경우 자동으로 enter code가 삽입됩니다. |
| s | 사용자 추가 코드 |

### 4.3.15 module.find

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module.find(name) |
| 반환값 | module |
| 설명 | 생성된 module 를 찾습니다. |
| name | 찾을 module 이름 |

### 4.3.16 module.is_valid

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module.is_valid(obj) |
| 반환값 | boolean |
| 설명 | module 객체가 맞는지 확인합니다. |
| obj | 확인할 모듈 객체. |

### 4.3.17 module.apply_code

| 구분 | 표현식 |
|---|---|
| 함수 원형 | function module.apply_code(filename) |
| 반환값 | - |
| 설명 | 코드 기술 파일로부터 코드를 읽어 각 module 에 코드로 삽입합니다.<br>코드 기술 파일에서 ":모듈명 (옵션)" 으로 시작한 뒤, 다음 줄부터 해당 모듈에 코드가 옵션의 결과가 true 일 때, 삽입됩니다.<br>옵션은 하위 기술된 코드들이 삽입 여부를 나타내는 Boolean 결과의 Lua 스크립트입니다.<br>이 옵션은 생략될 수 있습니다. (기본 값 : true) |
| filename | 코드 기술 파일명 |

ex) module.apply_code 예시 (Core, ALU 모듈에 코드를 추가하고자 할 때.)

```
module.apply_code("__core.sv")
```

[__core.sv]

```
:Core
assign  A = B;      // Core's code
assign  C = D;      // Core's code

:ALU (config.core_size > 4)
assign  E = F;      // ALU's code
assign  G = H;      // ALU's code
wire    [15:0]  CORE_SIZE   = $(config.core_size);
```

**NOTE:** verilog 코드 중간에 '$(*)' 또는 '${*}'로 기술하여, lua 코드를 실행할 수 있습니다. '$(*)'는 string 또는 number 반환되는 코드이며, '${*}'는 반환없는 lua 코드 실행을 기술할 수 있습니다.

#### 4.3.18 module_i:set_param

| 함수 원형 | function module_i:set_param(name, val) |
|---|---|
| 반환값 | - |
| 설명 | sub module의 parameter 값을 지정합니다. |
| name | parameter 이름 |
| val | parameter 설정 값 |

#### 4.3.19 module_i:get_param

| 함수 원형 | function module_i:get_param(name) |
|---|---|
| 반환값 | integer 또는 string |
| 설명 | sub module의 parameter로 지정된 값을 반환합니다. |
| name | parameter 이름 |

### 4.3.20 module_i:set_port

| 함수 원형 | function module_i:module_i:set_port(name, val) |
|---|---|
| 반환값 | - |
| 설명 | sub module의 port 값을 지정합니다. |
| name | port 이름 |
| val | port 설정 값 |

### 4.3.21 module_i:get_port

| 함수 원형 | function module_i:get_port(name) |
|---|---|
| 반환값 | interger 또는 string |
| 설명 | sub module의 port 값을 반환합니다. |
| name | port 이름 |

### 4.3.22 module_i.is_valid

| 함수 원형 | function module_i.is_valid(obj) |
|---|---|
| 반환값 | boolean |
| 설명 | module_i(sub module) 객체 여부를 반환합니다. |
| obj | module_i 객체 |

# 5. Appendix

## 5.1 Appendix : test_definition.lua

```
-------------------------------------------------------
-- clock definition
-------------------------------------------------------
clk     = {}

clk.MCLK    = clock:new("CLK", "main clock")    -- for core
clk.MCLK:set_speed(1000)

clk.PCLK    = clock:new("PCLK", "APB clock")
clk.PCLK:set_speed(100)
clk.PCLK:set_reset("PRESETn", "low")

clk.BCLK    = clock:new("ICLK", "interconnection clock")
clk.BCLK:set_speed(1500)

clk.ACLK    = clock:new("ACLK", "AXI clock")
clk.ACLK:set_speed(1000)

-------------------------------------------------------
-- bus interface
-------------------------------------------------------
bus     = {}

-- APB bus
bus.apb     = interface:new("apb")
bus.apb:set_clock(clk.PCLK)
bus.apb:set_param("ADDR_WIDTH", 16)
bus.apb:set_param("DATA_WIDTH", 32)
bus.apb:set_param("SEL_WIDTH", 2)
bus.apb:set_signal("PADDR", "ADDR_WIDTH")
bus.apb:set_signal("PSEL", "SEL_WIDTH")
bus.apb:set_signal("PENABLE")
bus.apb:set_signal("PWRITE")
bus.apb:set_signal("PWDATA", "DATA_WIDTH")
bus.apb:set_signal("PREADY")
bus.apb:set_signal("PRDATA", "DATA_WIDTH")
bus.apb:set_signal("PSLVERR")

bus.apb:set_modport("s", {["input" ]={"PSEL", "PENABLE", "PWRITE", "PADDR", "PWDATA"},
["output"]={"PREADY", "PRDATA", "PSLVERR"}})
bus.apb:set_modport("m", {["output"]={"PSEL", "PENABLE", "PWRITE", "PADDR", "PWDATA"},
["input" ]={"PREADY", "PRDATA", "PSLVERR"}})

bus.apb:set_prefix("S#")

-- AXI3 master bus
```

```
bus.maxi3   = interface:new("maxi3")
bus.maxi3:set_clock(clk.ACLK)
bus.maxi3:set_param("DATA_WIDTH", 128)
bus.maxi3:set_param("ADDR_WIDTH", 32)
bus.maxi3:set_param("ID_WIDTH", 4)
-- write address
bus.maxi3:set_signal("AWVALID")
bus.maxi3:set_signal("AWREADY")
bus.maxi3:set_signal("AWADDR", "ADDR_WIDTH")
bus.maxi3:set_signal("AWSIZE", 3)
bus.maxi3:set_signal("AWBURST", 2)
bus.maxi3:set_signal("AWCACHE", 4)
bus.maxi3:set_signal("AWPROT", 3)
bus.maxi3:set_signal("AWID", "ID_WIDTH")
bus.maxi3:set_signal("AWLEN", 4)
bus.maxi3:set_signal("AWLOCK", 2)
-- write data
bus.maxi3:set_signal("WVALID")
bus.maxi3:set_signal("WREADY")
bus.maxi3:set_signal("WLAST")
bus.maxi3:set_signal("WDATA", "DATA_WIDTH")
bus.maxi3:set_signal("WSTRB", "DATA_WIDTH/8")
bus.maxi3:set_signal("WID", "ID_WIDTH")
-- write response
bus.maxi3:set_signal("BVALID")
bus.maxi3:set_signal("BREADY")
bus.maxi3:set_signal("BRESP", 2)
bus.maxi3:set_signal("BID", "ID_WIDTH")
-- read address
bus.maxi3:set_signal("ARVALID")
bus.maxi3:set_signal("ARREADY")
bus.maxi3:set_signal("ARADDR", "ADDR_WIDTH")
bus.maxi3:set_signal("ARSIZE", 3)
bus.maxi3:set_signal("ARBURST", 2)
bus.maxi3:set_signal("ARCACHE", 4)
bus.maxi3:set_signal("ARPROT", 3)
bus.maxi3:set_signal("ARID", "ID_WIDTH")
bus.maxi3:set_signal("ARLEN", 4)
bus.maxi3:set_signal("ARLOCK", 2)
-- read data
bus.maxi3:set_signal("RVALID")
bus.maxi3:set_signal("RREADY")
bus.maxi3:set_signal("RLAST")
bus.maxi3:set_signal("RDATA", "DATA_WIDTH")
bus.maxi3:set_signal("RRESP", 2)
bus.maxi3:set_signal("RID", "ID_WIDTH")

bus.maxi3:set_modport("s", {
    ["input"]={
            "AWVALID", "AWADDR", "AWSIZE", "AWBURST", "AWCACHE", "AWPROT", "AWID", "AWLEN",
"AWLOCK",
            "WVALID", "WLAST", "WDATA", "WSTRB", "WID",
```

```
            "BREADY",
            "ARVALID", "ARADDR", "ARSIZE", "ARBURST", "ARCACHE", "ARPROT", "ARID", "ARLEN",
"ARLOCK",
            "RREADY", "RLAST"
            },
    ["output"]={
            "AWREADY", "WREADY", "BVALID", "BRESP", "BID", "ARREADY", "RVALID", "RDATA", "RRESP",
"RID"
            }
})
bus.maxi3:set_modport("m", {
    ["output"]={
            "AWVALID", "AWADDR", "AWSIZE", "AWBURST", "AWCACHE", "AWPROT", "AWID", "AWLEN",
"AWLOCK",
            "WVALID", "WLAST", "WDATA", "WSTRB", "WID",
            "BREADY",
            "ARVALID", "ARADDR", "ARSIZE", "ARBURST", "ARCACHE", "ARPROT", "ARID", "ARLEN",
"ARLOCK",
            "RREADY", "RLAST"
            },
    ["input"]={
            "AWREADY", "WREADY", "BVALID", "BRESP", "BID", "ARREADY", "RVALID", "RDATA", "RRESP",
"RID"
            }
})

bus.maxi3:set_prefix("M#")

-- AXI4 master bus
bus.maxi4   = bus.maxi3:new("maxi4")
bus.maxi4:set_signal("AWLOCK")          -- modified 2bit to 1bit
bus.maxi4:set_signal("ARLOCK")          -- modified 2bit to 1bit
bus.maxi4:set_signal("AWLEN", 8)        -- modified 4bit to 8bit
bus.maxi4:set_signal("ARLEN", 8)        -- modified 4bit to 8bit
bus.maxi4:set_signal("AWQOS", 4)        -- new on AXI4
bus.maxi4:set_signal("AWREGION", 4)     -- new on AXI4
bus.maxi4:set_signal("ARQOS", 4)        -- new on AXI4
bus.maxi4:set_signal("ARREGION", 4)     -- new on AXI4

bus.maxi4:add_modport("s", {
    ["output"]={"ARQOS", "ARREGION"},
    ["input" ]={"AWQOS", "AWREGION"}
})
bus.maxi4:add_modport("m", {
    ["input" ]={"ARQOS", "ARREGION"},
    ["output"]={"AWQOS", "AWREGION"}
})

bus.maxi4:set_prefix("M#")

--------------------------------------------------------
-- core interface
--------------------------------------------------------
```

```
core_i  = {}
core_i.inst     = interface:new("inst")
core_i.inst:set_signal("EN")
core_i.inst:set_signal("INST", 32)
core_i.inst:set_signal("READY")
core_i.inst:set_modport("m", {
    ["output"]={"EN", "INST"},
    ["input"] ={"READY"}
})
core_i.inst:set_modport("s", {
    ["input"]={"EN", "INST"},
    ["output"] ={"READY"}
})

------------------------------------------------------------
-- configuration
------------------------------------------------------------
config              = {}
config.core_size    = 4
```