

# MOD\_CLOCK\_ENABLE\_GENERATOR



January 27, 2025

Jay Convertino

# Contents

<b>1 Usage</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Dependencies . . . . .	2
1.2.1 fusesoc_info Depenecies . . . . .	2
1.3 In a Project . . . . .	2
<b>2 Architecture</b>	<b>2</b>
<b>3 Building</b>	<b>3</b>
3.1 fusesoc . . . . .	3
3.2 Source Files . . . . .	3
3.2.1 fusesoc_info File List . . . . .	3
3.3 Targets . . . . .	4
3.3.1 fusesoc_info Targets . . . . .	4
3.4 Directory Guide . . . . .	4
<b>4 Simulation</b>	<b>5</b>
4.1 iverilog . . . . .	5
4.2 cocotb . . . . .	5
<b>5 Code Documentation</b>	<b>6</b>
5.1 mod_clock_ena_gen . . . . .	7
5.2 tb_mod_ena . . . . .	9
5.3 tb_cocotb verilog . . . . .	11
5.4 tb_cocotb python . . . . .	13

# 1 Usage

## 1.1 Introduction

This core will generate an enable that is some divisible rate of the clock. The pulse will last for one clock cycle. This core is meant to be used in situations where the enable on a register is used to control the clock rate.

## 1.2 Dependencies

The following are the dependencies of the cores.

- fusesoc 2.X
- iverilog (simulation)
- cocotb (simulation)

### 1.2.1 fusesoc\_info Dependencies

- dep
  - AFRL:utility:helper:1.0.0
- dep\_tb
  - AFRL:simulation:clock\_stimulator
  - AFRL:utility:sim\_helper

## 1.3 In a Project

Simply use this core to generate a slow enable for a clocked device.

# 2 Architecture

The only module is the `mod_clock_ena_gen` module. It is listed below.

- **mod\_clock\_ena\_gen** Implement an algorithm to generate a slow enable based on a faster clock (see core for documentation).

This method of generating a slow enable allows for a design to save clocks and prevent timing issues. Since the timing is still based on the original clock and the enable signal is synchronous to it. It can suffer from jitter and deviation, but for devices such as a UART this

is well within its tolerance. In testing this seems to be 5 percent at worst down to 0 percent at best.

Method is based on a mod divide of the clock frequency and the requested output enable rate.

1. Add the requested enable rate to a counter
2. Set enable to 0
3. If the counter is equal to or greater than the clock frequency
  - (a) Set the counter to counter minus clock frequency. This will result in the overflow being set to the counter for the next build up.
  - (b) Set enable to 1

Please see ?? for more information.

## **3 Building**

The mod clock enable core is written in Verilog 2001. They should synthesize in any modern FPGA software. The core comes as a fusesoc packaged core and can be included in any other core. Be sure to make sure you have met the dependencies listed in the previous section.

### **3.1 fusesoc**

Fusesoc is a system for building FPGA software without relying on the internal project management of the tool. Avoiding vendor lock in to Vivado or Quartus. These cores, when included in a project, can be easily integrated and targets created based upon the end developer needs. The core by itself is not a part of a system and should be integrated into a fusesoc based system. Simulations are setup to use fusesoc and are a part of its targets.

### **3.2 Source Files**

#### **3.2.1 fusesoc\_info File List**

- src
  - src/mod\_clock\_ena\_gen.v
- tb
  - 'tb/tb\_mod\_ena.v': 'file\_type': 'verilogSource'

- tb\_cocotb
  - 'tb/tb\_cocotb.py': 'file\_type': 'user', 'copyto': '.'
  - 'tb/tb\_cocotb.v': 'file\_type': 'verilogSource'

### 3.3 Targets

#### 3.3.1 fusesoc\_info Targets

- default
 

Info: Default for IP intergration.
- sim
 

Info: Test
- sim\_cocotb
 

Info: Cocotb unit tests

### 3.4 Directory Guide

Below highlights important folders from the root of the directory.

1. **docs** Contains all documentation related to this project.
  - **manual** Contains user manual and github page that are generated from the latex sources.
2. **src** Contains source files for the core
3. **tb** Contains test bench files for iverilog and cocotb

## 4 Simulation

There are a few different simulations that can be run for this core. All currently use iVerilog (icarus) to run. The first is iverilog, which uses verilog only for the simulations. The other is cocotb. This does a unit test approach to the testing and gives a list of tests that pass or fail.

### 4.1 iverilog

All simulation targets that do NOT have cocotb in the name use a verilog test bench with verilog stimulus components. All of these tests provide fst output files for viewing the waveform in the there target build folder.

### 4.2 cocotb

To use the cocotb tests you must install the following python libraries.

```
$ pip install cocotb
```

Then you must use the cocotb sim target. In this case it is sim\_cocotb. This target can be run with various bus and fifo parameters.

```
$ fusesoc run --target sim_cocotb AFRL:clock:  
    ↪ mod_clock_ena_gen:1.0.0
```

## 5 Code Documentation

Natural docs is used to generate documentation for this project. The next lists the following sections.

- **mod\_clock\_ena\_gen** Generate a low rate enable clock.
- **tb\_mod\_ena** Verilog test bench.
- **tb\_cocotb verilog** Verilog test bench base for cocotb.
- **tb\_cocotb python** cocotb unit test functions.

# mod\_clock\_ena\_gen.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/01/27

---

## INFORMATION

---

### Brief

---

Generate a enable signal at some rate that divides the clock. This can be any rate. This enable will not be a 50% clock cycle or as stable as a pll. This uses the mod algorithm. Essentially it adds the number of ticks till it reaches the clock rate and then saves the remainder and generates a 1 cycle high pulse.

### License MIT

---

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## mod\_clock\_ena\_gen

---

```
module mod_clock_ena_gen #(
  parameter
    CLOCK_SPEED
    =
    2000000,
  parameter
    START_AT_ZERO
    =
    0,
  parameter
    DELAY
    =
```



```
0
) ( input clk, input rstn, input hold, input [31:0] rate, output ena )
```

Mod rate enable generator

## Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz
<b>START_AT_ZERO</b> parameter	Start counter at 0 if set. Otherwise start at CLOCK_SPEED/2.
<b>DELAY</b> parameter	Delay the enable by a number of clock ticks

## Ports

<b>clk</b>	Clock used for enable generation
<b>rstn</b>	Negative reset for anything clocked on clk
<b>hold</b>	hold enable low and pause + reset count till hold removed (low).
<b>rate</b>	rate that enable pulse will be generated, must be less then the clock rate.
<b>ena</b>	positive enable that is pulsed high at enable rate.

## tb\_mod\_ena.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2025/01/27

---

### INFORMATION

---

#### Brief

---

Test bench for mod clock divide enable generator

#### License MIT

---

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## tb\_mod\_ena

---

```
module tb_mod_ena #(
  parameter
    CLOCK_SPEED
    =
    20000000,
  parameter
    START_AT_ZERO
    =
    0,
  parameter
    DELAY
    =
    0
)()
```

---

mod clock enable test bench

### Parameters

<b>CLOCK_SPEED</b> parameter	Clock speed
<b>START_AT_ZERO</b> parameter	Set to 1 to enable start at zero.
<b>DELAY</b> parameter	Set to the number of clock cycles to delay the enable output signal.

## INSTANTIATED MODULES

---

### clk\_stim

```
clk_stimulus #(
    CLOCKS(1),
    CLOCK_BASE(CLOCK_SPEED),
    CLOCK_INC(1000),
    RESETS(1),
    RESET_BASE(2000),
    RESET_INC(100)
) clk_stim ( .clkv(tb_dut_clk), .rstnv(tb_dut_rstn), .rstv() )
```

Generate a 50/50 duty cycle set of clocks and reset.

### dut

```
mod_clock_ena_gen #(
    CLOCK_SPEED(CLOCK_SPEED),
    START_AT_ZERO(START_AT_ZERO),
    DELAY(DELAY)
) dut ( .clk(tb_dut_clk), .rstn(tb_dut_rstn), .hold(1'b0), .rate(115200), .e
```

Device under test, mod\_clock\_ena\_gen

## tb\_cocotb.v

---

### AUTHORS

---

JAY CONVERTINO

---

### DATES

---

2025/01/27

---

### INFORMATION

---

#### Brief

---

Test bench wrapper for cocotb

#### License MIT

---

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## tb\_cocotb

---

```
module tb_cocotb #(
  parameter
    CLOCK_SPEED
    =
    20000000,
  parameter
    START_AT_ZERO
    =
    0,
  parameter
    DELAY
    =
    0
) ( input clk, input rstn, input hold, input [31:0] rate, output ena )
```

---

Mod rate enable generator test bench

### Parameters

<b>CLOCK_SPEED</b> parameter	This is the aclk frequency in Hz
<b>START_AT_ZERO</b> parameter	Start counter at 0 if set. Otherwise start at CLOCK_SPEED/2.
<b>DELAY</b> parameter	Delay the enable by a number of clock ticks

### Ports

<b>clk</b>	Clock used for enable generation
<b>rstn</b>	Negative reset for anything clocked on clk
<b>hold</b>	hold enable low and pause + reset count till hold removed (low).
<b>rate</b>	rate that enable pulse will be generated, must be less then the clock rate.
<b>ena</b>	positive enable that is pulsed high at enable rate.

## INSTANTIATED MODULES

---

### dut

```
mod_clock_ena_gen #(
    CLOCK_SPEED(CLOCK_SPEED),
    START_AT_ZERO(START_AT_ZERO),
    DELAY(DELAY)
) dut ( .clk(clk), .rstn(rstn), .hold(hold), .rate(rate), .ena(ena) )
```

Device under test, mod\_clock\_ena\_gen

# tb\_cocotb.py

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2025/01/27

---

## INFORMATION

---

### Brief

---

Cocotb test bench

### License MIT

---

Copyright 2025 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## FUNCTIONS

---

### random\_bool

---

```
def random_bool()
```

Return a infinite cycle of random bools

Returns: List

### start\_clock

---

```
def start_clock(  
    dut  
)
```

Start the simulation clock generator.

### Parameters

**dut** Device under test passed from cocotb test function

## reset\_dut

---

```
async def reset_dut(  
    dut  
)
```

Cocotb coroutine for resets, used with await to make sure system is reset.

### Parameters

**dut** Device under test passed from cocotb.

## count\_pulses

---

```
async def count_pulses(  
    dut  
)
```

Cocotb task to count pulses from a output.

## speed\_test

---

```
@cocotb.test()  
async def speed_test(  
    dut  
)
```

Test various speeds of the output enable

### Parameters

**dut** Device under test passed from cocotb.

## in\_reset

---

```
@cocotb.test()  
async def in_reset(  
    dut  
)
```

Coroutine that is identified as a test routine. This routine tests if device stays in unready state when in reset.

### Parameters

**dut** Device under test passed from cocotb.

### no\_clock

---

```
@cocotb.test()
async def no_clock(
    dut
)
```

Coroutine that is identified as a test routine. This routine tests if no ready when clock is lost and device is left in reset.

### Parameters

**dut** Device under test passed from cocotb.