SIPO

April 30, 2025

Jay Convertino

Contents

1 Usage

1.1 Introduction

This core converts the serial input data to parallel output data. This is done on a positive clock edge in relation to a active high enable. The enable sets the rate data is sampled. This core is useful for any serial to parallel operation such as UARTS, SPI, i2c, 1553, etc. This is a fusesoc 2.X compatible core and its VLNV is AFRL:simple:sipo:X.X.X.

1.2 Dependencies

The following are the dependencies of the cores.

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

1.2.1 fusesoc_info Depenecies

- dep
  - AFRL:utility:helper:1.0.0

1.3 In a Project

This core is made to be used on the same clock domain as the input parallel data. The serial input rate is set by the enable. The enable should only be pulsed for one clock cycle. If the clock is the rate the enable should be tied high. Load has to be used to get the output data and reset the input count. The data count provers other cores a bit count to know what bit currently being input to the core. On load data is cleared from the input register and the count is reset to 0. Once enable is toggled the count will increment and indicates how many bits of the word are contained in the register. The register is full once the max number of bits is reached on the counter output.

1. Set ena to 1 at rising clock edge for one clock cycle
2. Continue till the counter hits the number of bits needed.
3. Read the parallel data output to capture data.
4. Assert load to 1 to restart counter and clear parallel register.

Adding the core to a fusesoc project, outside of adding the verilog module to your code, requires it is added as dependency. Example:

```
dep:
  depend:
    - ">=AFRL:simple:sipo:1.0.0"
```

Module instantiation:

```
sipo #(
  .BUS_WIDTH(4)
) inst_sipo (
  .clk(clk),
  .rstn(rstn),
  .ena(ena),
  .load(load),
  .pdata(pdata),
  .sdata(sdata),
  .dcount(dcount)
);
```

2 Architecture

This core is made to interface for serial to parallel data. On each enable pulse on a positive clock edge the input serial line will be sampled. The counter starts at 0 and the shift is complete once it hits BUS_WIDTH * 8. For a 32 bit word you will need 32 enable pulses. These will correspond to counter output 0 to 31. 32 will signal the shift is done and the parallel output data is valid. Once it is valid data can be read and load asserted. Load will reset the core and clear it for the next shift operation. All data is shifted in MSb to LSb. Meaning Bit 31 is input first and 0 is last. The only module is the sipo module. It is listed below.

- **sipo** Convert serial data to parallel by shifting in data on each enable pulse on a rising clock. (see core for documentation).

Please see 5 for more information.

2.1 Waveform

Below is a waveform in a typical application of the core. This shows the count up and how the enable is pulsed to increment it.

3 Building

The SIPO 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 as a dependency. Be sure you have meet 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 into 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/sipo.v
- tb
  - tb/tb_sipo.v
- 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: Base simulation using icarus as default.

- 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
    - **cocotb** testbench files

4 Simulation

There are a few different simulations that can be run for this core. The backend used for testing is iverilog for verilog or cocotb simulations. Usually GTKWave is used to view the fst waveform output. Cocotb are the unit tests that attempt to give a pass/fail verification to the core operation.

4.1 iverilog

iverilog is used for simple test benches for quick visual verification of the core. This will autofinish after it has run up to a certain number of words have been output.

4.2 cocotb

This method allows for quick writing of test benches that actually assert and check the state of the core. These tests are much more conclusive since it will run all test vectors and generate a report if they pass or fail. All tests generate to a single fst wave file. The method of launching the tests is to use fusesoc. These have not been written to use the python runner method or makefiles. To use the cocotb tests you must install the following python libraries.

```
$ pip install cocotb
```

The targets available are listed below.

- **sim_cocotb** Standard simulation PISO conversion using cocotb.

The targets above can be run with various parameters.

```
$ fusesoc run --target sim_cocotb AFRL:simple:sipo
    ↪ :1.0.0
```

5 Module Documentation

- **sipo** SIPO converter
- **tb_sipo-v** Verilog test bench
- **tb_cocotb-py** Cocotb python test routines
- **tb_cocotb-v** Cocotb verilog test bench

The next sections document the module in detail.

sipo.v

AUTHORS

JAY CONVERTINO

DATES

2025/04/15

INFORMATION

Brief

SIPO (serial in parallel out) The idea is to keep this core simple, and let the control logic be handled outside of this core.

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.

sipo

```
module sipo #(
parameter
BUS_WIDTH
 =
1
) ( input clk, input rstn, input ena, input load, output [BUS_WIDTH*8-1:0]
```

serial in parallel out

Parametes

BUS_WIDTH width of the parallel data input in bytes.

Ports

clk	global clock for the core.
rstn	negative syncronus reset to clk.
ena	enable for core, use to change input rate. Enable serial shift input.
load	load parallel data from core, and reset counters for next incoming serial data.
pdata	parallel data output, valid when dcount is BUS_WIDTH*8.
sdata	serialized data input.
dcount	Number of bits to shift out. BUS_WIDTH*8 means all bits have been sampled and put into the register.

tb_sipo.v

AUTHORS

JAY CONVERTINO

DATES

2025/04/16

INFORMATION

Brief

Test bench for Serial in Parallel out core.

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 [width=0.3\textwidth]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_sipo

```
module tb_sipo ()
```

Test bench for serial in parallel out

INSTANTIATED MODULES

sipo

```
sipo #(
```

```
                                                                        .
BUS_WIDTH(1)
) dut ( .clk(tb_clk), .rstn(tb_rstn), .ena(tb_ena), .load(tb_load), .pdata(
```

Device under test, serial to parallel data conversion.

tb_cocotb.py

AUTHORS

JAY CONVERTINO

DATES

2025/04/16

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.

piso

piso

Convert parallel data to serial

VARIABLES

wqueue

```
self.wqueue
```

Queue to store write requests

_idle_write

```
self._idle_write
```

Event trigger for cocotb read

FUNCTIONS

_restart

```
def _restart(
self
)
```

kill and restart _run thread.

random_bool

```
def random_bool()
```

Return a infinte 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.

increment test

Coroutine that is identified as a test routine. Write data, on one clock edge, read on the next.

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.

tb_cocotb.v

AUTHORS

JAY CONVERTINO

DATES

2025/04/16

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.BUS_WIDTH

tb_cocotb

```
module tb_cocotb #(
parameter
BUS_WIDTH
  =
4
) ( input clk, input rstn, input ena, input load, output [BUS_WIDTH*8-1:0]
```

SIPO interface DUT

Parameters

BUS_WIDTH Width of the APB3 bus data port in bytes.
parameter

Ports

clk	Clock
rstn	negative reset
ena	enable for core, use to change input rate. Enable serial shift input.
load	load parallel data from core, and reset counters for next incoming serial data.
pdata	parallel data output, valid when dcount is BUS_WIDTH*8.
sdata	serialized data input.
dcount	Number of bits to shift out. BUS_WIDTH*8 means all bits have been sampled and put into the register.

INSTANTIATED MODULES

dut

```
sipo #(
                                                                                .
BUS_WIDTH(BUS_WIDTH)
) dut ( .clk(clk), .rstn(rstn), .ena(ena), .load(load), .pdata(pdata), .sda
```

Device under test, sipo