Stream Syncer

A small SystemVerilog project by Naftali Kizner © 2021

Challenge Definition

Imagine an endless incoming bit stream, divided into frames of fixed length (e.g., 128 bits) with a fixed 8-bit beginning pattern (e.g., 11101000 (‘he8)). 3 consecutive appearances of this pattern (one per frame) should be found to declare in-frame. The data is output as an 8-bit bus.

Task Breakdown

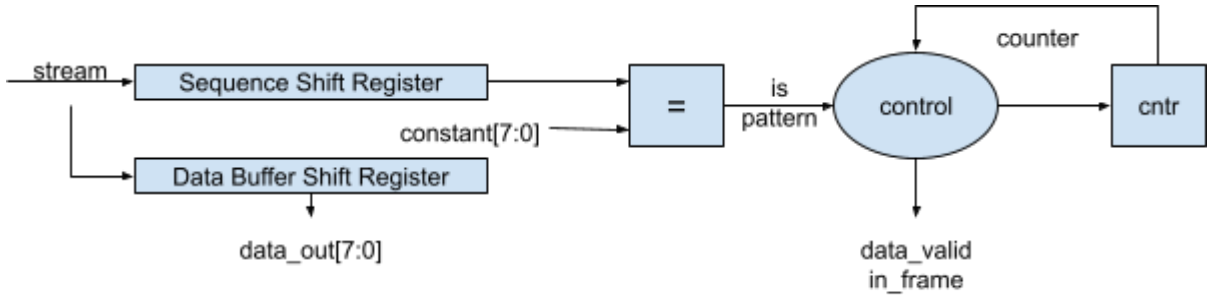
Interface will be defined as follows, with a few assumptions taken:

Name	I/O	Duty
`stream`	`Input`	`Endless bit stream`
`clock`	`Input`	`Clock`
`reset`	`Input`	`Reset the machine`
`data_out[7:0]`	`Output`	`Filtered buffered data (w/o pattern)`
`data_valid`	`Output`	`data_out is valid for reading`
`in_frame`	`Output`	`data_out is within a legal frame`

This system is looking for 3 legal frames, meaning frames that begin with a specific sequence, to declare a legal data output. Any discovery of this sequence out-of-sync, meaning not every frame length, will be handled by resyncing.

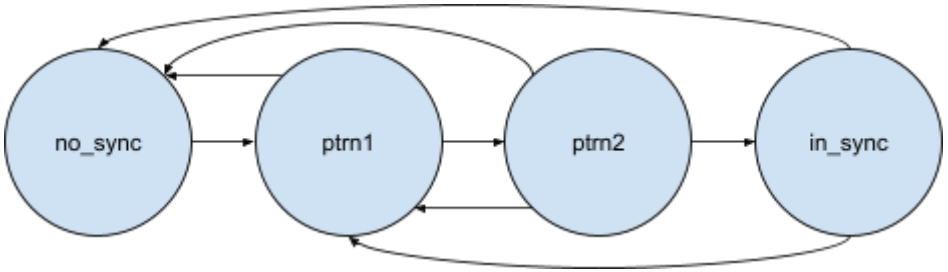
Solution

The system consists of two shift registers, a controller (FSM), a comparator, a frame-size counter, and other combinational logic. Top-level diagram is shown below:



The usage of two shift registers allows flexibility of the output bus size, meaning the system is capable of dealing with sequence sizes which are different from the output bus size.

The FSM is described by the state diagram below:



Controller's pseudo state table is given below:

Current State	Next state				data_valid				in_frame			
	pattern on time	early pattern	late/no pattern	data frame	pattern on time	early pattern	late/no pattern	data frame	pattern on time	early pattern	late/no pattern	data frame
no_sync	ptrn1	ptrn1	no_sync	no_sync	0	0	0	0	0	0	0	0
ptrn1	ptrn2	ptrn1	no_sync	ptrn1	0	0	0	0	0	0	0	0
ptrn2	in_sync	ptrn1	no_sync	ptrn2	0	0	0	0	0	0	0	0
in_sync	in_sync	ptrn1	no_sync	in_sync	0	0	0	1 every 8 cycles	1	0	0	1

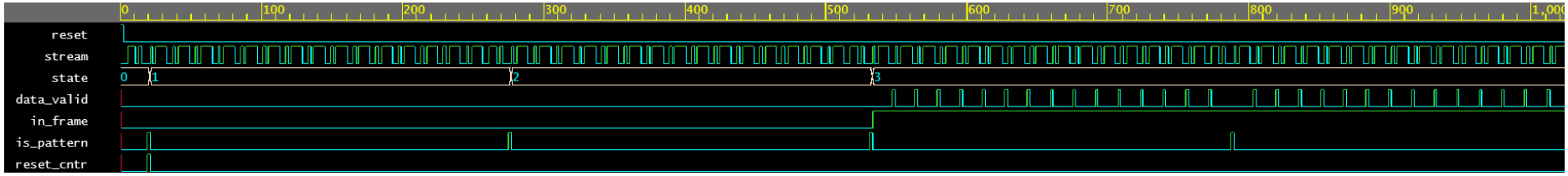
Where inputs to the FSM are defined as:

- Pattern on time - `is_pattren == 1 && counter == 128`
- Early pattern - `is_pattren == 1 && counter  < 128`
- Late/no pattern - `is_pattren != 1 && counter == 128`
- Data frame - `is_pattern != 1 && counter  < 128`

Results

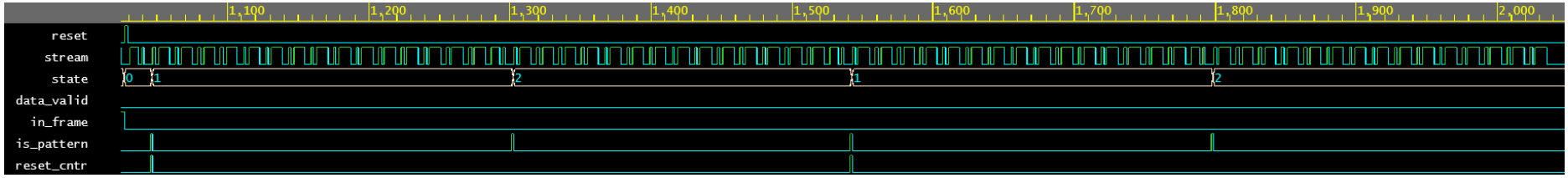
Three tests comprise the testbench:

1. Streaming 4 proper frames



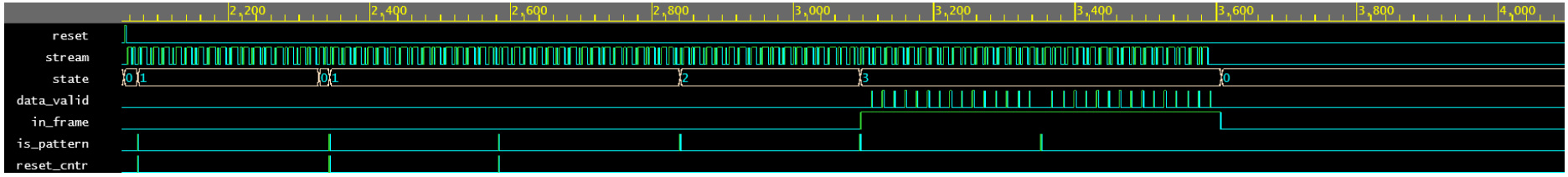
Results appear as expected: `in_frame` rises after the 3rd pattern, and `data_valid` strobes every 8 bits.

2. Streaming one proper frame, then one short (by one byte), and then another 2 proper frames



Results appear as expected: `in_frame` nor `data_valid` ever rise.

3. Streaming a frame that has an extra byte, then one short (by one byte), and then 4 proper frames



Results appear as expected: the system managed to sync and raise `in_frame` and `data_valid` properly.

Further Development

Only 3 test scenarios were run on the system. More tests should be run, while altering the following parameters:

- Stream frames length
- Stream frames content
- Size of data_out
- Sync pattern

Code

Project's code is available at:
https://github.com/naftali10/Samll-SV-Projects/tree/main/Stream%20Syncer
It's also available for running at:
https://www.edaplayground.com/x/rxwh
Instructions for running:
Enter the link and press the Run button on the top-left corner. You will be asked to log in to EDA playground. Do so by clicking the orange "Log In (no save)" button. Choose your preferred login method - Google/Facebook account is good enough. Once logged in, re-enter the link above and click Run again. The log file at the bottom will display results of the testbench on the left.