


Computation II: embedded system design (5EIB0)


[Dashbo...](#) / [My cou...](#) / [5E...](#) / [Final Exam 2024-...](#) / [Part 2d - 2024-04-12 Finite State Machine \(Implement FSM Moore UART Frame Odd Par...](#)

Description

[Submission view](#)

 **Available from:** Friday, 12 April 2024, 1:30 PM

 **Due date:** Friday, 12 April 2024, 4:30 PM

 **Requested files:** uart_parity_odd.v ( [Download](#))

Type of work:  Individual work

Universal Asynchronous Receiver/Transmitter (UART) is a simple serial protocol that is commonly used to communicate between two devices that do not share the same synchronous clock. Data is communicated as frames from one device to the other device along a single wire. The line is idle when it is high. The start of a frame is signalled by pulling the line low for a single data bit duration. This is followed by 4 data bits (this number can vary between implementations, but we will consider 4 here) plus a parity bit that can be high or low, representing half a byte of data plus a parity check bit in binary. This must be immediately followed by the stop bit where the line is held high for at least a single data bit duration. A break signal can be communicated by pulling the line low, such that the stop bit is not signalled.

The parity bit is used as a simple error detection mechanism. For odd parity detection the check bit is set such that the number of binary ones from the data including the check bit are odd, e.g.

data	parity-bit	ones
0000	1	1 - odd
0010	0	1 - odd
1010	1	3 - odd
1110	0	3 - odd

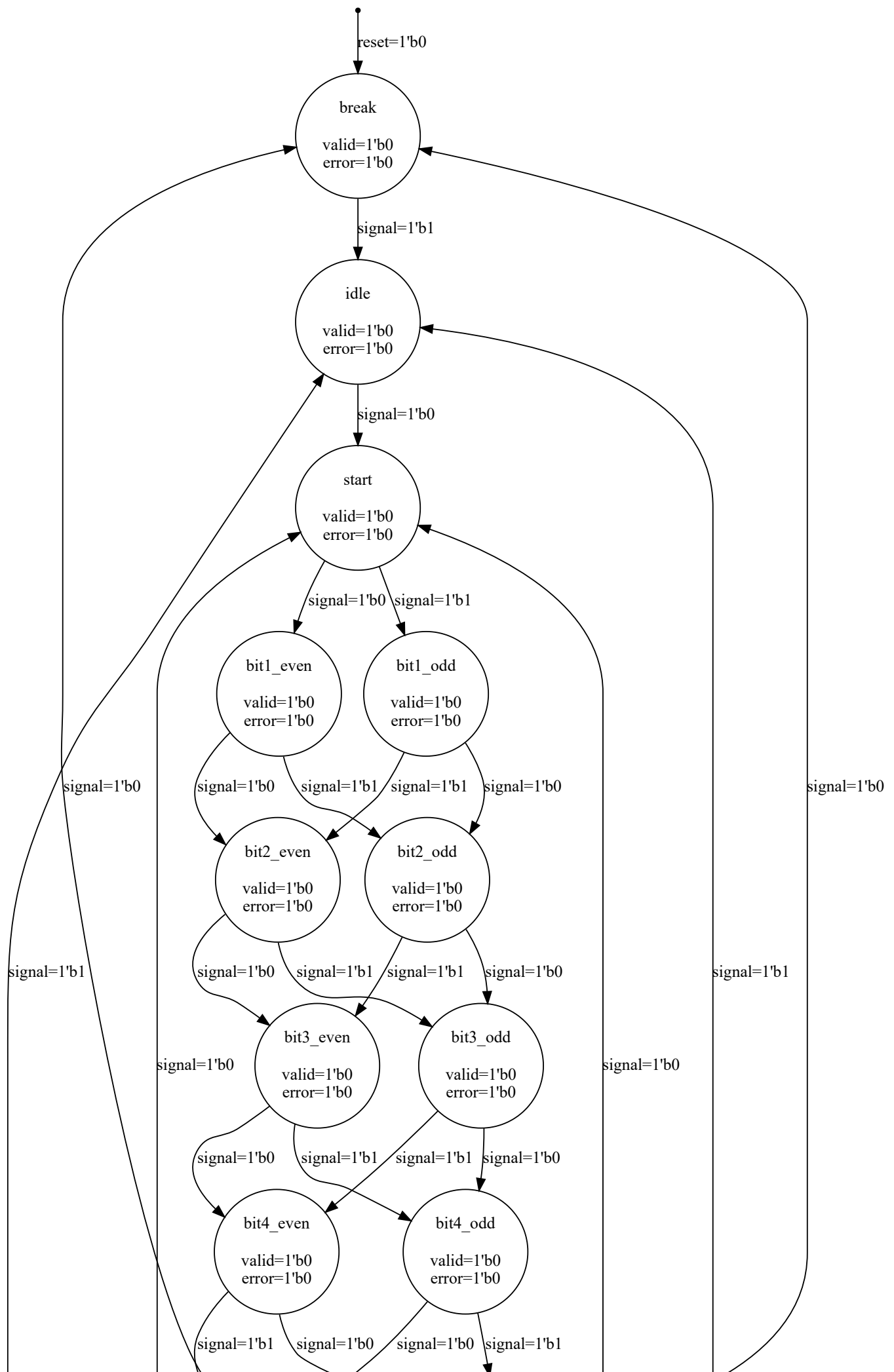
The receiving side checks to see whether the number of ones is odd and if it is not then the data has become corrupted during transmission, e.g.

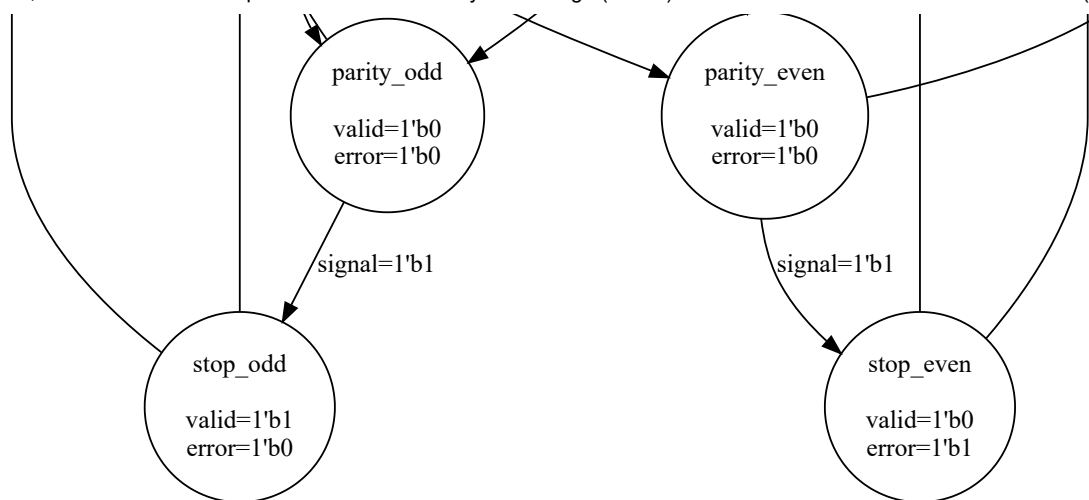
data	parity-bit	ones
0000	0	0 - even
0010	0	1 - odd
1011	0	3 - odd
1110	1	4 - even

The parity check can only detect an odd number of bit flips, but it is a commonly used low overhead check.

In this assignment you will create a Verilog implementation of a Finite State Machine (FSM) with the following specifications. Including the clock (clk) and reset signals, the implemented FSM will have 3 input signals and 2 output signals.







The clk input signal is 1-bit wide, the reset input signal is 1-bit wide and the signal input signal is 1-bit wide. The state elements of the FSM are to capture their inputs on the positive clock (clk) edge and reset their state when the synchronous reset signal is high. A reset can be asserted (synchronously) at any time causing all state elements to reset and hence a transition to the initial state.

The valid output signal is 1-bit wide and the error output signal is 1-bit wide. Initial values of the output signals upon reset are shown in the FSM diagram.

Your solution will be tested using bounded model checking against a golden reference implementation. The model checking software will stop at the earliest moment that your implementation diverges from the behaviour of the golden reference implementation. If a divergence takes place, a waveform will be created showing a trace of the module signals resulting in the divergent state. This waveform also shows the behaviour of the golden reference (REF) so that you can see what your implementation (DUT) should have done differently.

Please note that the golden reference implementation supersedes all other implementation descriptions. If the output from your solution implementation does not diverge from the golden reference implementation then your solution is marked as correct. Otherwise, your solution is incorrect and you should use the waveform that is produced to assist you to correct your implementation.

[VPL](#)

You are logged in as Thomas Stirling Valdez (Log out)
5EIB0

Data retention summary