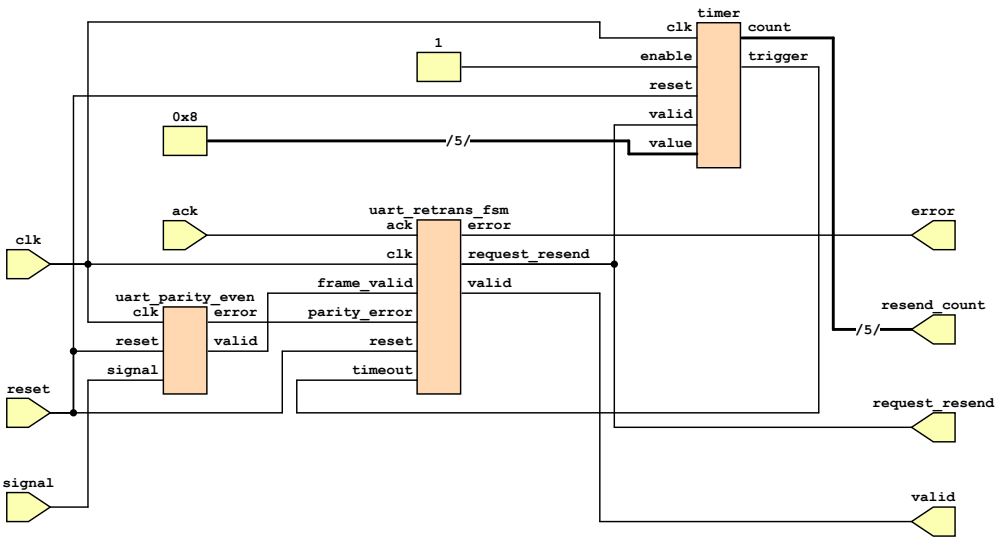
Computation II: embedded system design (5EIB0)

Available from: Friday, 12 April 2024, 1:30 PM
Due date: Friday, 12 April 2024, 4:30 PM
Requested files: uart_retrans.v, uart_retrans_fsm.v (Download)
Type of work: Individual work

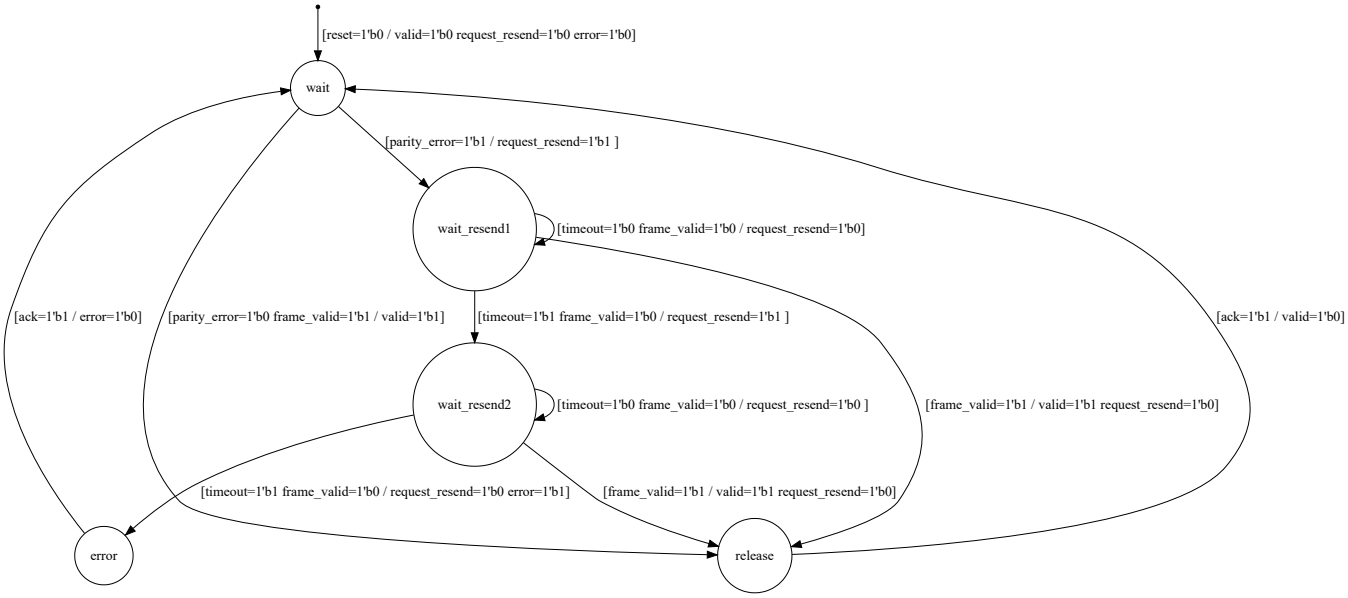
Universal Asynchronous Receiver/Transmitter (UART) is a simple serial protocol commonly used to communicate between two devices that do not share the same synchronous clock. Data is communicated as frames from one device to the other device along a single wire. The UART protocol also accounts for the use of a parity check to test if the data being communicated has been corrupted between the sender and the receiver.

In this assignment, you will create a Verilog implementation of a module that receives the UART signal and performs an even parity check. If the parity check is good the module will signal that it is valid. If the parity check is bad the module will signal that the frame should be resent. If the next frame arrives within the specified time frame and the parity check is good then the module will signal that it is valid. If the timeout is reached without a valid frame being signalled then an error signal is produced. The module will have the following structure:



Input signals are illustrated towards the left of the diagram with an arrow to the right. Output signals are illustrated towards the right of the diagram with an arrow to the left. Wires are shown as lines. Bold lines represent multiple wires with the bit width specified on the line between forward slashes, e.g. a bit width of 5 is represented as a /5/ on the line. Modules to be instantiated are shown as vertical rectangular boxes with input ports on the left and output ports on the right. The names of the modules are shown above the rectangle. For this question, the **`uart_parity_even`** and **`timer`** modules have been provided for you and can be instantiated. The **`uart_retrans_fsm`** module needs to be created as follows.

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 6 input signals and 3 output signals.



The clk input signal is 1-bit wide, the reset input signal is 1-bit wide, the parity_error input signal is 1-bit wide, the frame_valid input signal is 1-bit wide, the timeout input signal is 1-bit wide and the ack 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 transtion to the initial state.

The valid output signal is 1-bit wide, the request_resend 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 in correcting your implementation.

VPL