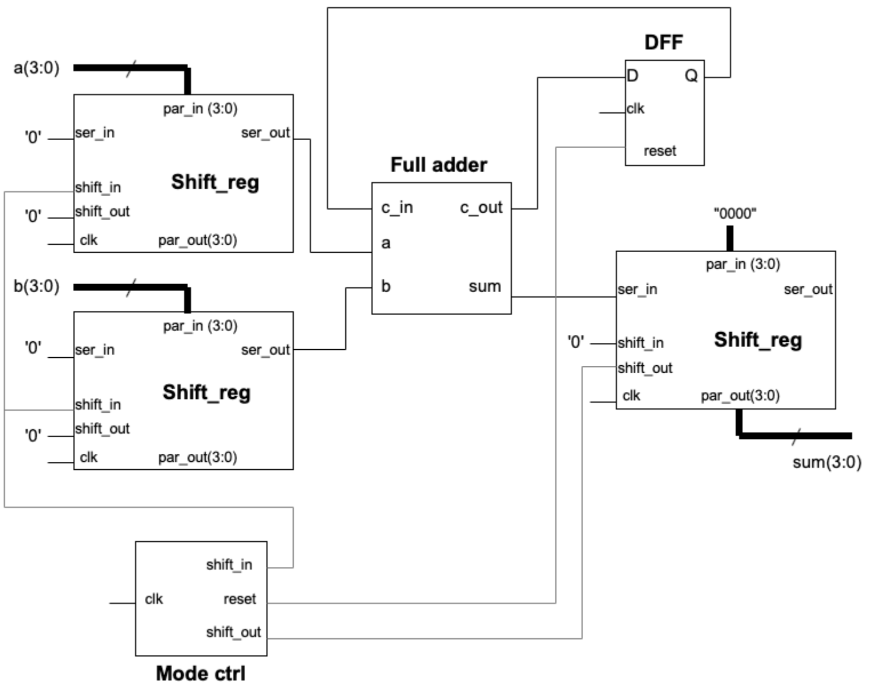
Lab 2: Serial adder

Introduction

This lab exercise is an introduction to sequential logic and state machines. Like Lab 1, you will create a design that adds two 4-bit numbers, but this circuit will use state machines to convert the two parallel inputs into serial streams, add them with a single full adder, and then convert the sum back into a parallel output. A D flip-flop saves the carry bit from each single-bit addition to the next cycle.



For this lab, you should design and simulate the different components separately. Once the individual component functionalities have been verified, integrate them in the top-level design, debug the design (especially relative timing) in simulation, and finally implement and test the design in hardware.

Component descriptions

Full Adder

For this component, you may re-use your code from Lab 1. Alternatively, you may write a new version if you prefer.

Shift Register (shift_reg)

This component is a multi-purpose shift register that can be used to:

- At the two inputs: Receive a 4-bit parallel word (`par_in`) and convert it to a serial bitstream (`ser_out`) for output to the full-adder.
- At the output: Receive a serial bitstream (`ser_in`) from the full-adder output and convert it to a 4-bit parallel word (`par_out`).

The input and output of the shift register is controlled by two signals: `shift_in` and `shift_out`.

When `shift_in` is set to 1, the four bits of parallel input data are loaded from `par_in` into the 4-bit register. When `shift_in` is set to 0, the input bit `ser_in` is written to the top (most significant) bit of the register and the other three bits are shifted down by one position. The least significant bit of the register is connected to output port `ser_out`.

When `shift_out` is set to 1, the contents of the shift register are written to `par_out`. Otherwise `par_out` does not change.

Flip flop (DFF)

This is a 1-bit register (flip-flop) that holds the `c_out` bit from the full adder and sends it to `c_in` for the next clock cycle. The DFF also includes a `reset` signal that clears the output of the DFF, setting it to zero. This is used to prevent the final (most significant) carry bit of a 4-bit addition from being carried into the next 4-bit addition.

You may write and instantiate the DFF as a separate component, or simply implement it in the top-level design as a process.

Mode control

The mode control component is a simple state machine that controls and synchronizes the different components of the serial adder. The three outputs (`shift_in, shift_out` and `reset`) control the input shift registers, output shift register, and DFF reset, respectively. The mode_control state machine should have four states, one for each pair of input bits being added.

In the input shift registers, the 4-bit parallel inputs are read in when `shift_in` is equal to 1. For the next three cycles `shift_in` is set to 0 as the contents of the register are shifted out serially.

In the output shift register, the `shift_out` signal is set to 1 when the serial output is correctly aligned just after a four bit summation is completed, writing the register contents to `par_out`. `shift_out` is then set to 0 for the next three cycles until the next aligned sum arrives.

Finally, the `reset` signal is set to 1 when the most-significant bits are being summed in the full-adder, in order to prevent the carry-out of the top-most bits in the summation from being carried in to the next calculation.

The relative timing of the three `mode_control` outputs must be correct in order for the serial adder to function properly. Stand-alone simulation of this component will only confirm a repeating sequence of the three control signals, so when you simulate the final integrated design you may find that you need to adjust the relative timing to get the correct results.

Implementation

After you have written and simulated each of the individual components, instantiate them in a top-level architecture resembling the diagram on Page 1. As mentioned above, it is important to simulate the serial adder in a VHDL test bench to verify the correct timing throughout the design. The relative timing of the `mode_control` output signals is especially important to check and adjust; pay special attention to correct behavior of the carry chain, and the alignment of the parallel output.

After verifying in simulation, implement the full design on your developer board. Use eight switches to provide the two four-bit inputs, and four LEDs to display the output. For timing, use the 100 MHz oscillator clock supplied to pin W5 of the FPGA.

Ideas for further development

If you wish to, you may choose to display the output as hexadecimal numbers using the seven-segment LED display. This can be done by instantiating the VHDL module disp4.vhd downloadable from the course site.

Note that the LED display driver port map includes a clock, 16 input bits (corresponding to four 4-bit digits) and eight output bits corresponding to the anode and seven cathode lines to the display.

Your adder produces only a one digit output, so the parallel output should be connected to the four least significant bits of the driver component. The other driver input bits should be set to '0'. You can also use the same clock as the rest of your design.

The FPGA I/O pins connected to the anode and cathode lines of the 7-segment LED display are documented in the Basys3 reference manual, also available on the course site.