CS 152A / EE M116L

Introductory Digital Lab Design

LAB #2

4-Bit Serial Adder Implementation

April 29, 2014

Grade:

Tianheng Tu

Name1: Nathan Tung

SID1: 004-059-195

Name2: Mark Iskandar

SID2: 704-050-889

INTRODUCTION

The purpose of Lab 2 is to design and build a 4-bit serial adder, a device that can add two 4-bit binary numbers, A and B, serially within a total of eight clock cycles. We use chips including ANDs, MUXes, asynchronous counters, shift registers, a full adder, and a D flip-flop. In order to construct our circuit, we first examined the data sheets of the chips used, verified our logic using truth tables, and created a wiring schematic of the entire breadboard to follow.

In this report, we will specify the role of each component in our serial adder and demonstrate how they work collectively as a whole. A basic design of our 4-bit serial adder is later shown.

DESIGN COMPONENTS

Overall

Two 4-bit numbers in binary, A and B, are accepted as input via the eight switches at the bottom of the board. The debounced pushbutton PB1 acts as a clock and, through the counter, allows us to remember which cycle we are currently on. In the first four clock cycles, the 4-bit inputs A and B are loaded into the shift register. In the following four clock cycles, we are adding the last bits of A and B (via a single adder within the full adder chip), accounting for overflow which is saved in the D flip-flop, and placing each resulting bit back into B's shift register. In general, the MUX chips select which bits to store in the shift register. The bits in shift register B are outputted to the 4-bit logic indicator, and the carry-out from the D flip-flop is displayed in the LED. The 4-bit indicator displays the resulting sum, and the LED essentially records the 5th bit in the case of overflow.

SR Latch (Debouncer)

By wiring up an SR Latch (using the NAND gates inside the TTL chip) for both the Clock and Reset pushbuttons, we effectively debounce their input to make the voltage distinction between high and low very clear. Reset clears the D flip-flop and resets the current clock cycle stored inside the counter. We will be using the already-debounced Clock and Reset as inputs.

Asynchronous Counter

The counter takes in Clock as input and decides which clock cycle we are currently on. Although the counter is capable of using 4-bits, we really only need the last 3-bits for memorizing eight clock cycles (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111). Individual output bits can be used as MUX selectors and also to notify the D flip-flop when a clock cycle has passed such that the current flip-flop input can be memorized. We'll assume the counter output is DCBA.

MUX1, MUX2, MUX3

All four 2-bit multiplexers in MUX1 are used in order to select between the switch inputs for A and B. In this case, we use the counter's output A as the selector so that, for each of the first four clock cycles, we are toggling between which MUX input is selected. On the other hand, MUX2 uses counter output B as the selector to narrow down MUX1's output into two 1-bit numbers for A and B to be passed into their respective shift registers.

The selection process is dependent on the clock cycle. At clock cycle 0000, we want S4 (or A0) and S8 (or B0). At 0001, we want S3 and S7, and so on. The table below summarizes this:

B (from counter)	A (from counter)	A (selected 1-bit)	B (selected 1-bit)
0	0	S2, **S4**	S6, **S8**
0	1	S1, **S3**	S5, **S7**
1	0	**S2**, S4	**S6**, S8
1	1	**S1**, S3	**S5**, S7

To be more precise, the 1-bit numbers from the switch are passed straight into shift register A, but not into shift register B. Since shift register B holds 4-bit B for the first four clock cycles and the final sum for the next four clock cycles, an additional MUX3 is used to select between these inputs. When counter output C is 0, MUX3 gives the 1-bits of B to shift register B. When counter output C is 1, MUX3 instead passes the full adder's 1-bit output to shift register B.

Shift Registers (A and B)
Shift register A takes in a single bit at a time from the user's inputted A value, starting with the LSB (least significant bit) and working towards the MSB (most significant bit). At each clock cycle, we shift this input right by one bit and wait for the next higher significant bit until all four bits are in. A similar procedure applies to shift register B with the user's inputted B value. However, for the last four clock cycles, shift register B begins to accept the summation results from the full adder and shift them right until the final, 4-bit summation result is in.

Full Adder
The adder accepts the last bit from both shift registers and uses the D flip-flop's Q output as the carry-in. It then adds these bits together. The summation bit is passed to MUX3 to be potentially stored in Shift Register B when applicable, whereas the carry-out bit is passed to the D flip-flop to be remembered for the next valid clock cycle.

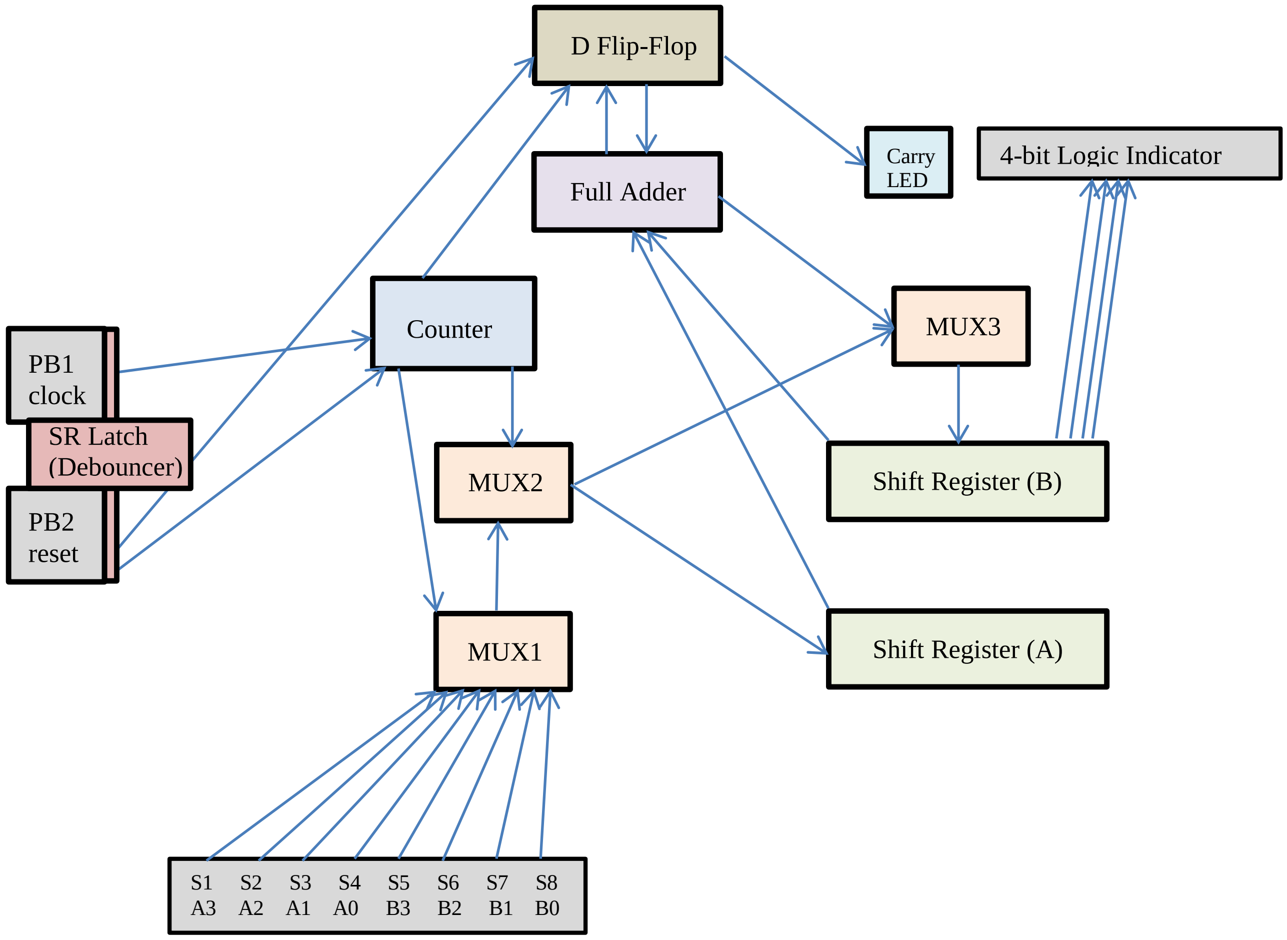
D Flip-Flop
Given the adder's carry-out as input, the D flip-flop stores this bit and produces it as Q for the following clock cycle. As mentioned before, Q is passed to the full adder for use as carry-in. It is also connected to the LED such that the LED will light up when overflow occurs. Ultimately, a bright LED tells us that the 5th bit (not displayed as a logic indicator) is 1.

CONCLUSION

This report addresses the mechanisms – both in the components and the design as a whole – that allow this 4-bit serial adder to work as intended. Our team faced some challenges in terms of avoiding one-off errors in counting the clock cycle, as well as creating a design that is easy to build. Eventually, we overcame these challenges and constructed a system of modules that was both interesting to build and to test.

Lab Contribution:
Nathan Tung: 50%
Mark Iskandar: 50%

DESIGN SCHEMATIC


D Flip-Flop
Carry LED
4-bit Logic Indicator
Full Adder
Counter
MUX3
PB1 clock
SR Latch (Debouncer)
PB2 reset
MUX2
Shift Register (B)
MUX1
Shift Register (A)
S1 A3
S2 A2
S3 A1
S4 A0
S5 B3
S6 B2
S7 B1
S8 B0