

EE 2310 Homework #7 – 4-Bit Adder/Data Processor

This mandatory semester design project is the only homework assignment to be turned in. It is recommended that you use LogicWorks™ for your design. If you do not have LogicWorks™, you must use a computer layout tool, and your project design must be printed out. If you use LogicWorks™, simply turn in a print-out of the circuit (make sure to write your name on the circuit drawing). Copying and pasting a LogicWorks design into Word works very well, and you can reduce the size of your drawing to exactly one 8 ½ x 11 page. This homework is triple bonus problem. You get TRIPLE the points earned, although it only counts as one homework. Anyone who turns in this assignment gets a minimum of 100, so long as it is a “good-faith” effort (i.e., a serious attempt to complete the design).

Problem Statement: It is now time to tie together what has been learned about logic design. Design the following circuit using T and D FFs, plus other required simple logic gates.

You are to design a data receiver, timer, and adder circuit that will store two 4-bit numbers in registers made of D flip-flops, and add those two 4-bit binary numbers together (note that you are adding two positive binary numbers, NOT two’s complement). You must then transmit the result on a separate data bus. Because the system into which you are transmitting the data uses inverted logic (i.e., logic 1 = 0, logic 0 = 1), you must output the one’s complement of the sum onto the data bus. Note that such a circuit might be used in a digital system where the calculated sum is used to determine whether a process should be started or ended. There is a reset that starts up your system, and then a clock starts and runs continuously. Events in your circuit are based on an 8-count cycle after the clock starts, so you must provide a modulo-8 counter. The following signal inputs are given:

1. Clock In: A 50/50 pulse input that begins after Reset– and runs continuously.
2. Reset–: A negative-going signal that resets the system at startup before clock start.
3. Data: Four data lines, W-Z (W = MSB), on which data appears as follows: Data Word 1 is valid on the W-Z lines on count 2 of the counter, and Data Word 2 is valid on count 4.

Your job is to design the following:

1. Build a 3-bit counter that counts modulo-8 continuously. It is reset only at startup with Reset–.
2. Build two 4-bit registers to hold the data word that is transmitted on data lines, W-Z (again, W = MSB). Data Word 1 is stored in its register on count 2, and Data Word 2 in its register on count 4.
3. Build a 4-bit adder to add the contents of the two registers.
4. Build a 5-bit register to hold the sum and carry out (overflow) of the adder (outputs V’-Z’)
5. Build a 5-bit inverter circuit that converts to negative logic.
6. Build a timer circuit that detects counts 2, 4, 6, and 7 and generates the appropriate signals. The signals should be ½ clock cycle in length, valid when clock is low.
7. On count 6, the add is completed and you store the sum of Words 1 and 2 in the output register.
8. On count 7, you must send a signal of ½ clock cycle duration (called Data Valid +) that data is valid on the output register lines V’-Z’.

The output signals from your system are:

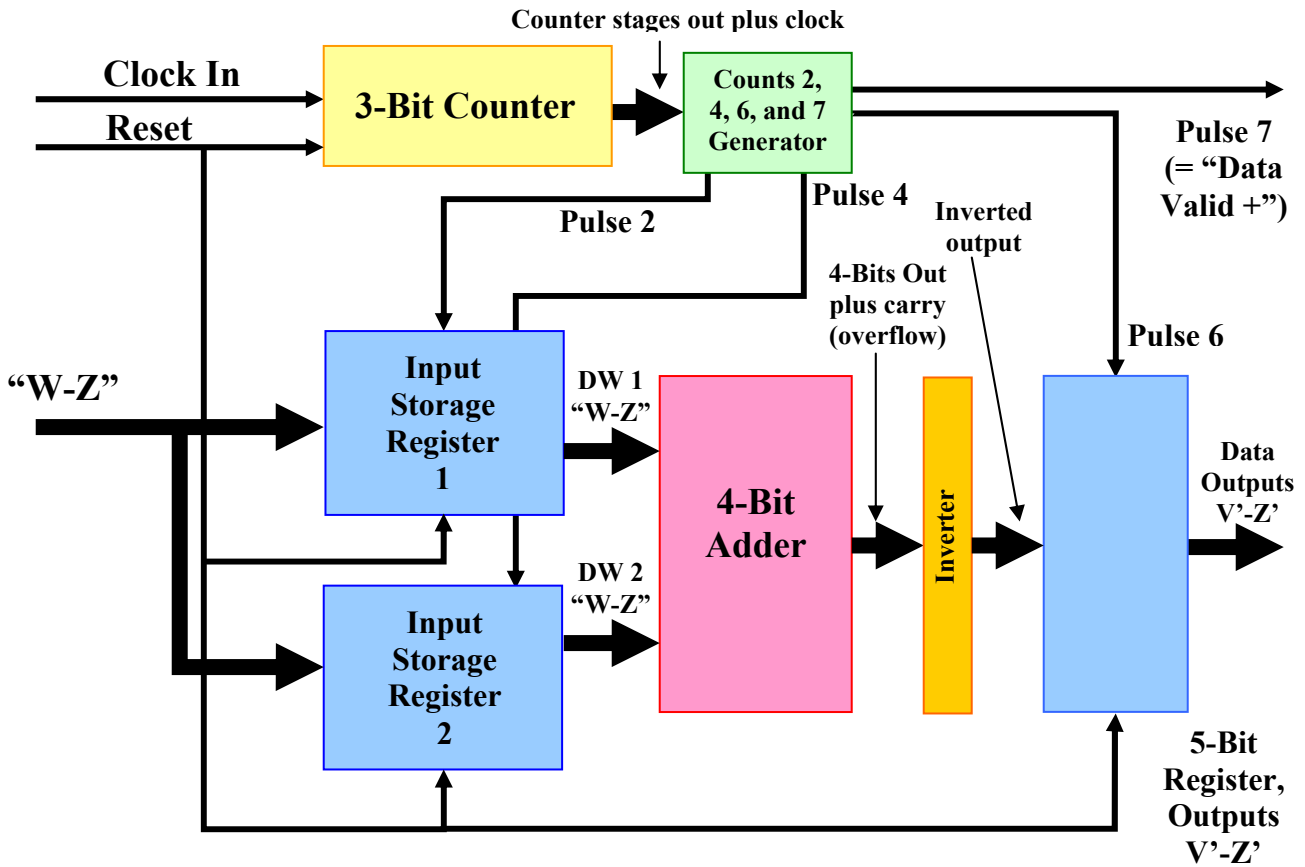
1. Data Valid +: A ½ -clock-cycle wide signal that is sent to indicate that the outputs V’-Z’ are valid.
2. V’-Z’: Output data lines from your output registers.

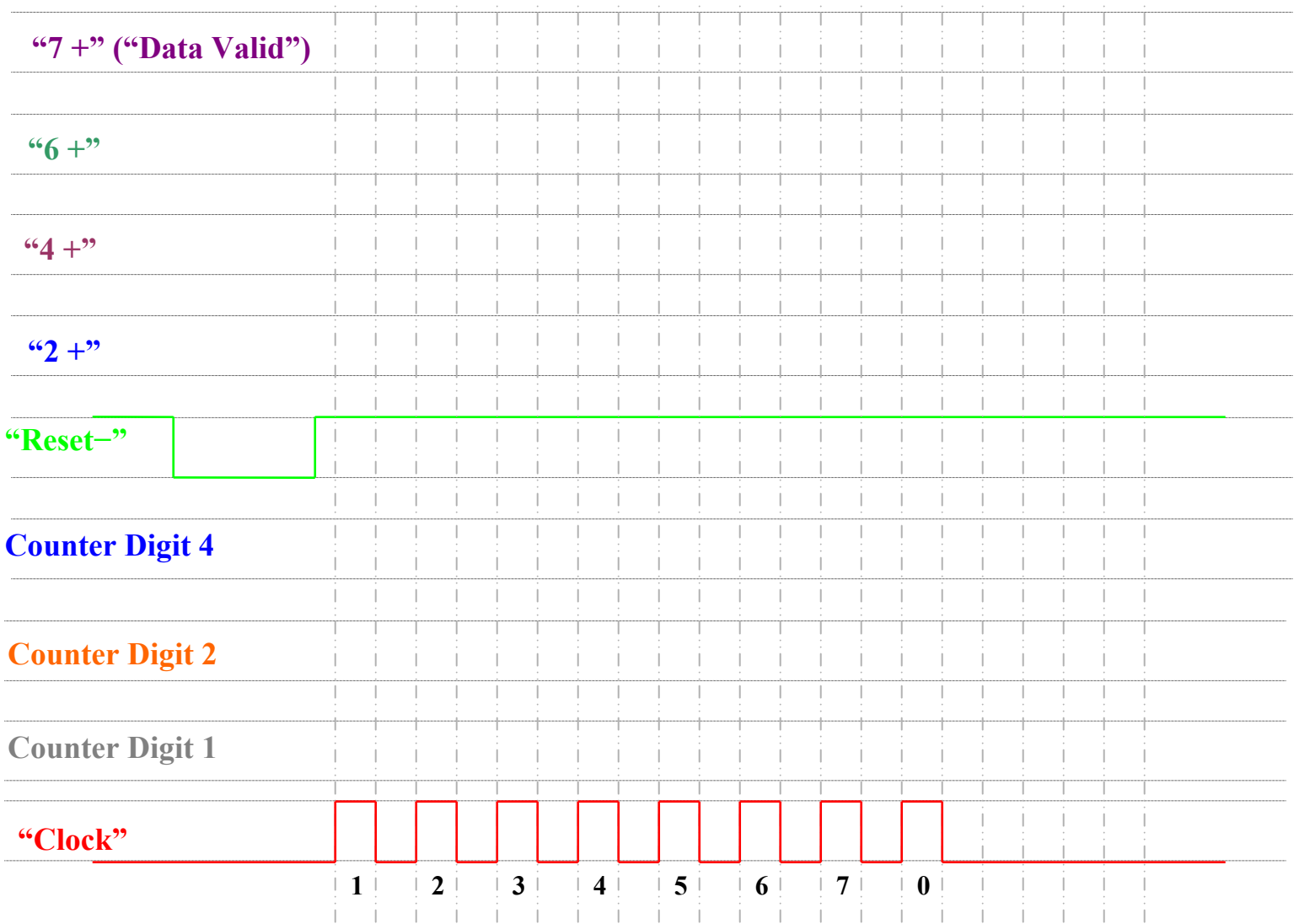
Arrange your design drawing so that input signals enter from the left, and the outputs for your circuit exit on the right. Also, clearly label each input or output element (e.g., “W,” “Clock In,” “Count 2,” etc.).

Show the timing for the circuit for one 8-count cycle. Include all relevant signals except the input data W-Z and output data V'-Z', since they coincide with counts 2, 4, and 7. Use D and T ff's as shown below to make your registers and counters. These ff's are in LogicWorks. You may NOT use any register or counter circuits – you must design them from the component flip-flops. Also, you must design the adder from basic components (you may, however, use XOR gates, which simplifies a 4-bit adder considerably).



The block diagram below should help you in the design, as it shows the key subsystems of the overall design plus how they are interconnected.





Timing Diagram for 4-Bit Adder and Data Processor