Revision

Dr. Paul D. Franzon

Outline

1. Revision points
2. Digital system timing
Revision Points

I do expect you to be very familiar with at least the following concepts:

- Different combinational logic structures, including gates, adders, multipliers, coders, decoders, etc.
- Combinational logic optimization
- Flip-flops and latches, and their operation.
- Timing diagrams. How to produce one.
- Finite State Machines – purpose, operation, types, state vector encoding
- Counters – basic operation
- MOST IMPORTANTLY, I expect you to be able to design a logic function to a specification (like Q7 with relative ease)

If any of these topics are NOT familiar to you, I suggest reviewing your undergraduate logic course or logic course text. If that is not available to you, there are many suitable texts in the library. Authors include Katz, Wakerley, Mano, but there are many other.s
General Principles of Digital System Design

Most Digital Systems are Synchronous

- I.e. All signals are derived off a single Master Clock fed to all registers
- In most logic implementation families, events are synchronized using edge-triggered flip-flops
  - e.g. positive-edge triggered D- or Data- flip-flop
- Groups of flip-flops are referred to as registers
Timing Diagram

The core tool for analyzing synchronous digital designs is a timing diagram.

don't know (don't care)

Glitches at input do not appear at output.

F/F only samples 'D' at positive clock edge.
Multiple Logic Stages

Generate Timing Diagram

Clock

In

Out
Animation

Track the logic…
Logic With Feedback

Generate Timing Diagram

Q. If we don’t know the initial value of “Out” can we answer this question?

Clock
In
Out

© Dr. Paul D. Franzon, 2011, www.ece.ncsu.edu/erl/faculty/paulf.html
Snapshot

Note: Original notes have Out=3 at t=0. This is a correction.
What is Wrong Here?

Combinational Logic Feedback does not work in synchronous design

- Must Feedback through register

- What happens if the logic gets into the state shown?
Digital sub-systems are built as collections of registers and combinational logic.

- Registers store data from the end of one clock period to be available at the start of the next clock period.
- Combinational logic cannot store data. It only operates on it during each clock period.
What is wrong here?

Any logic block only operates on data coming out of the LH clock. After start of clock period to present to input of RH flip-flop before end of clock period. Cannot operate over 2 periods. Does NOT have internal storage.
Sample Design Problem

Accumulator:

- Design an 8-bit adder accumulator with the following properties:
- While ‘accumulate’ is high, adds the input, ‘in1’ to the current accumulated total and add the result to the contents of register with output ‘accum_out’.
  - use absolute (not 2’s complement) numbers
- When ‘clear’ is high (‘accumulate’ will be low) clear the contents of the register with output ‘accum_out’
- The ‘overflow’ flag is high if the adder overflows

*Hint:*

8-bit adder produces a 9-bit result:
{carry_out, sum} = A+B;
Sketch Design

1. Determine and name registers.
2. Determine combinational logic

\[
\text{Clear} \quad \overset{\overset{\text{accumulate}}{}}{.}
\]