

University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

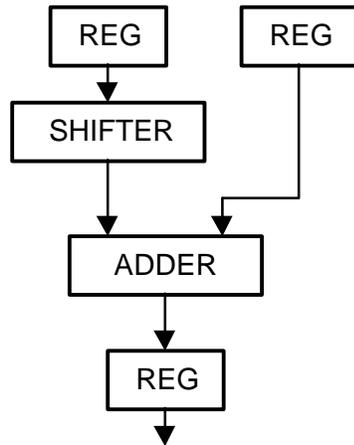
EECS150
 Spring 2002

J. Wawrzynek

Homework #11
 Revised Version 5/3

This homework is due on **Tuesday May 7th by 1pm**. Homework will be accepted in the EECS150 box on the door to room 218 Cory Hall. Late homework will be penalized by 50%. No late homework will be accepted after the solution is posted.

- For the given circuit, assume that $T_{\text{SHIFTER}}=10\text{ns}$, $T_{\text{ADDER}}=12\text{ns}$, T_{FF} (setup and clock to Q)=2ns.



- What is the *unpipelined* throughput for the circuit?
 - Draw a new version with two pipeline stages.
 - What is the new *throughput*? What is the new *latency*?
- The following RTL describes one iteration of a looped computation performed on a micro-programmed data-path. The computation takes three inputs A, B, and C, and produces one output per iteration on Z. All other “variables” in the RTL are registers:

```

X1←A, X2←Y2;
Y←X1 * X2, X1←B, X2←Y1;
T←Y, Y←X1*X2;
X1←C, X2←Y;
Y←X1+X2;
X1←T, X2←Y;
Y←X1+X2;
Y2←Y1, Y1←Y, Z←Y;
    
```

- a) Recall that an *abstract computation graph* is a graph linking operators that minimizes registers and exposes parallel operations. Operators that are shared in the circuit should be separate in the graph. Furthermore, the only registers in the graph are those needed for holding feedback values. Draw the abstract computation graph for this computation that produces one Z value per cycle.
 - b) Based on your answer for part a) draw a graph for a computation that completes 4 loop iterations in parallel, producing 4 Z values per cycle.
3. For the 8-bit processor presented in class, write out the detailed micro-code for the `add` and `ldb` instructions. Do this by filling in a table with one column for each control bit needed in by the data-path, plus one to control instruction dispatch. Fill in the table with 1s and 0s, one row per cycle.
4. The 8-bit processor presented in class has a data-path and RTL designed to minimize the *cycle time* at the expense of many cycles per instruction.
- a) Modify the data-path and RTL to minimize the number of *cycles per instruction* at the expense of increased cycle time. Redraw the data-path and rewrite the RTL to reflect your new design. Assume the memory elements (register file, and memory) have asynchronous read and synchronous write.
 - b) Making reasonable assumptions about delays through the data-path elements, compare the *total time* per instruction for your solution versus the one presented in class.