CSE471—Computer Design and Organization

Prof. Steven Burns *Autumn 1994*

Lecture Notes: Tenth Day

Processor Design

- Read Sections 5.1 through 5.4 as soon as possible.
- We'll design a processor for a subset of the MIPS instruction set.
- We'll use three different design styles:
  - Single-clock cycle implementation (5.3)
  - Multiple-clock cycle implementation (5.4)
  - Microprogrammed implementation (5.5)
- We'll develop detailed Verilog models of these three implementations.

Sequential Circuits

- We need to review a couple of things from CSE 370 and define our clocking methodology.
- We'll use edge-triggered registers: all storage devices (except possibly the large memories) will be written to on the rising edge of the clock.
- It will be convenient to use registers (D flip-flops) that also have *WriteEnable* inputs (like the '377).

MIPS Subset

- We'll implement these instructions:
  - R-type: `add`, `sub`, `and`, `or`, and `slt`.
  - Load-Store: `lw`, and `sw`.
  - Branch: `beq`.
  - Jump: `j`.

Espresso

- Espresso is a CAD tool from Berkeley for minimizing two-level representations of combinational logic.
- Espresso is the core of MacAbel, used in the CSE 370 lab for programming PLDs.
- We'll use it here to simplify control.
- Let's look at the two control units for the Single-Clock Processor.

Function Code to ALU Code

- Here is a description of the first one in Espresso format:

```
.type fr
.i 8
.o 3
00 ------ 010
01 ------ 110
10 --0000 010
10 --0010 110
10 --0100 000
10 --0101 001
10 --1010 111
.e
```

Syntax

- The `.type fr` command tells espresso how to handle don't cares. In this case, we want all input patterns not explicitly mentioned to produce don't care outputs.
- The `.i 8` and `.o 3` commands state the number of input and output signals. The `.e` commands delimits the end of the file.
- Espresso includes many heuristics to speed up the minimization. This is required if the logic is large.

Running Espresso

- You can turn off the heuristics and do exact (Quine-McCluskey) minimization if you want. I do this for small pieces.

```
espresso -Dexact alufr.pla > alufr.out
```

- The output is:

```
.i 8
.o 3
.p 6
1-----1- 100
1------1 001
1---1--- 001
-1------ 100
-----0-- 010
0------- 010
.e
```

ALU Control (cont.)

- Here is what happens if you don't include the command `.type fr`

```
.i 8
.o 3
.p 6
10--0101 001
10--1010 001
10---010 110
-0--00-0 010
01------ 100
0------- 010
.e
```

- The logic is larger because the outputs for all the unspecified inputs must be set to all zeroes.

The Main Control Unit

- Here is the Espresso file:

```
.type fr
.i 6
.o 9
000000 100100010
100011 011110000
101011 -1-001000
000100 -0-000101
.e
```

- The output is:

```
.i 6
.o 9
.p 4
---0-0 100100010
---1-- 000000101
--0--1 011110000
--1--- 010001000
.e
```

Main Control (cont.)

- This is the output if you don't include `.type fr`

```
.i 6
.o 9
.p 4
000100 000000101
101011 010001000
000000 100100010
100011 011110000
.e
```