Discussion Session 3

CS/ECE 552

Ramkumar Ravi

13 Feb 2012

GENERAL HW DELIVERABLES

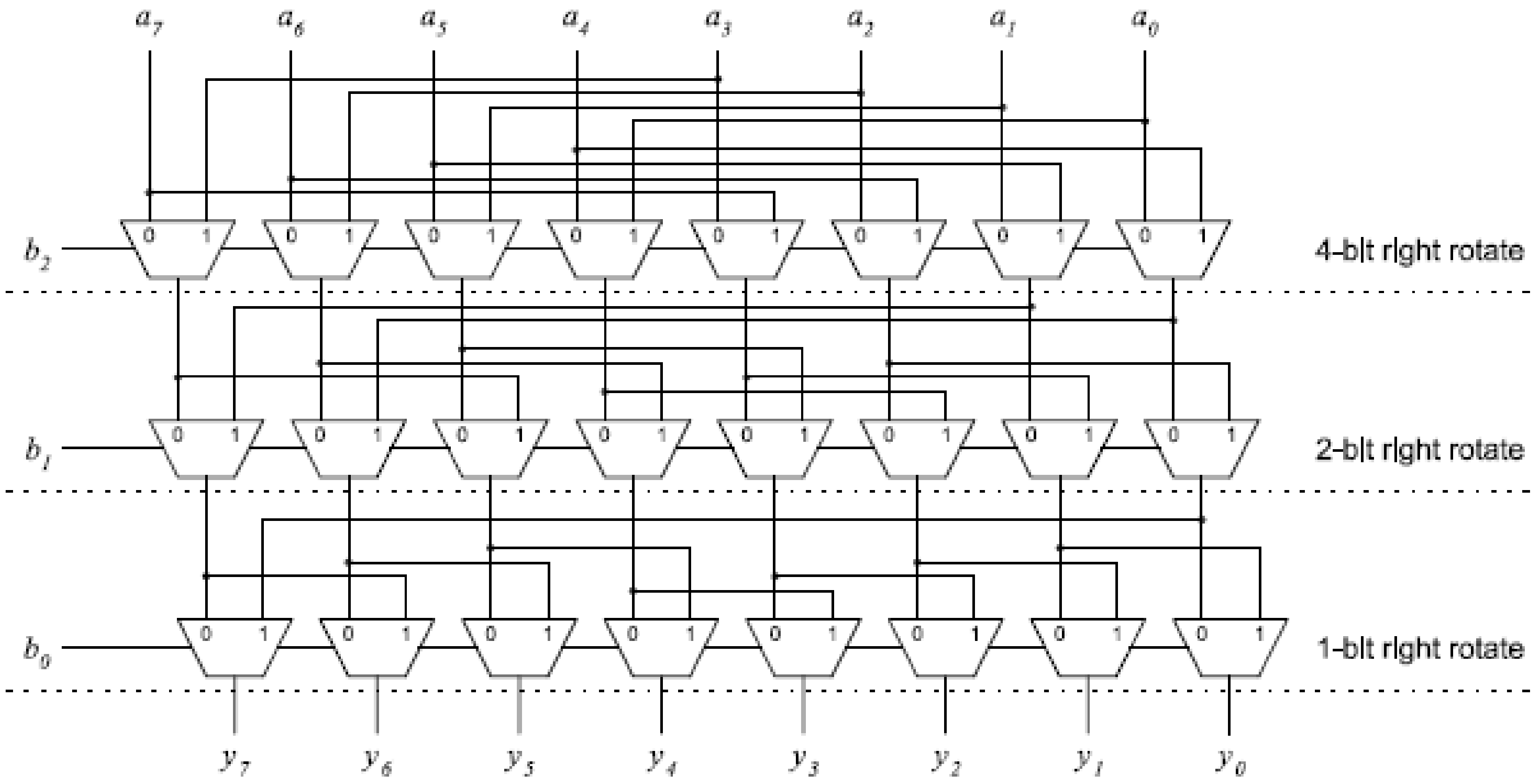
- ELECTRONIC
  - All verilog files including testbench
  - Vcheck.out files
  - Anything else that is mentioned in the assignment
- MANUAL
  - Annotated waveforms
  - Schematic of the design
  - Anything else that is mentioned in the assignment

Problem 1 – Barrel shifter

- Barrel shifters are often utilized by embedded digital signal processors and general-purpose processors to manipulate data
- In this table, the bit vector for A is denoted as $a_7a_6a_5a_4a_3a_2a_1a_0$ and the shift/rotate amount is 3 bits. As illustrated in this table

Operation	Y
3-bit shift right logical	0 0 0 $a_7\ a_6\ a_5\ a_4\ a_3$
3-bit shift right arithmetic	$a_7\ a_7\ a_7\ a_7\ a_6\ a_5\ a_4\ a_3$
3-bit rotate right	$a_2\ a_1\ a_0\ a_7\ a_6\ a_5\ a_4\ a_3$
3-bit shift left logical	$a_4\ a_3\ a_2\ a_1\ a_0$ 0 0 0
3-bit shift left arithmetic	$a_7a_3\ a_2\ a_1\ a_0$ 0 0 0
3-bit rotate left	$a_4\ a_3\ a_2\ a_1\ a_0\ a_7\ a_6\ a_5$

8-bit Right Rotate example implementation



Points to note

- EXPLORE – Design can be simplified by using a combination of 2:1 MUX and the 4:1 MUX you designed in the last HW (<= 50 lines of code)

- Hint: What do you think about this code below ?

```
wire [15:0] S0;
wire [15:0] L0;
// level 1
mux2_1 inst [13:0] (.InA(In[13:0]), .InB(In[15:2]), .Out(lev0[14:1]), .S(Op[1]));
mux4_1 inst1 (.out(lev0[0]), .InA(In[15]), .InB(1’d0), .InC(In[1]), .InD(In[1]), .S(Op));
mux4_1 inst2 (.out(lev0[15]), .InA(In[14]), .InB(In[14]), .InC(In[0]), .InD(In[15]), .S(Op));
mux2_1 inst [15:0] (.InA(In), .InB(L0), .S(cnt[0]), .out(out));

/levels 2, 4 and 8
??
```

From previous slide – All 1 bit operations

- For rotate left (Op = 00)
  - lev0 [14:1] = In [13:0]
  - lev0 [0] = In [15]
  - lev0 [15] = In [14]
- For shift left (Op = 01)
  - lev0 [14:1] = In [13:0]
  - lev0 [0] = 0
  - lev0 [15] = In [14]
- For rotate right (Op = 10)
  - lev0 [14:1] = In [15:2]
  - lev0 [0] = In [1]
  - lev0 [15] = In [0]
- For shift right arithmetic (Op = 11)
  - lev0 [14:1] = In [15:2]
  - lev0 [0] = In [1]
  - lev0 [15] = In [15]

WHAT TO SUBMIT

- ELECTRONIC
  - Verilog code of all modules and testbench
- MANUAL
  - Neat Schematic of the design (hand-drawn is fine)
  - Annotated waveforms
  - Explain why you chose a set of inputs for your simulation (in 3 or 4 sentences)

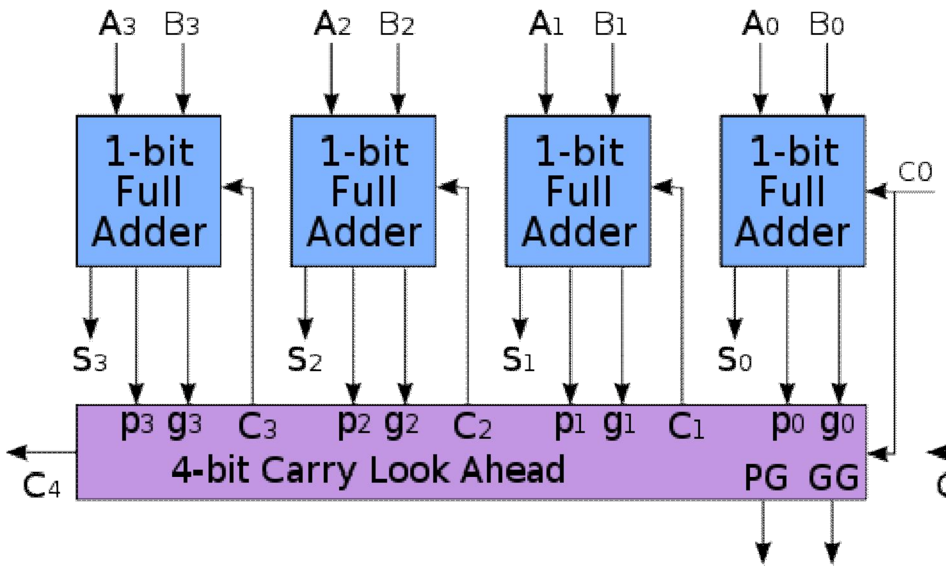
Problem 2 - ALU

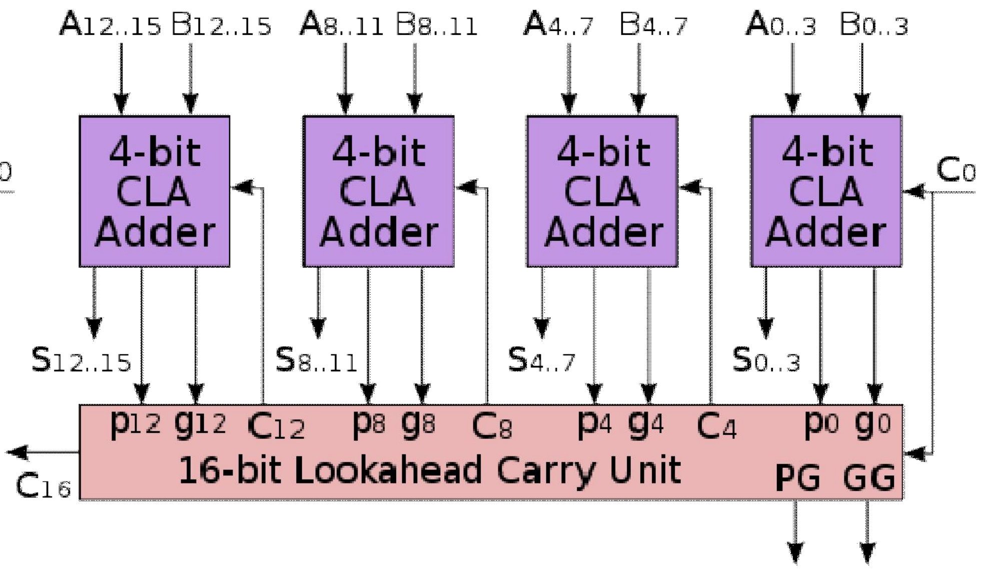
- Carry look ahead adder example (works on carry-generate and carry-propogate)

$$G(A,B) = A \cdot B$$
$$P'(A,B) = A \oplus B$$
$$C_{i+1} = G_i + (P_i \cdot C_i)$$

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1$$
$$C_3 = G_2 + P_2 \cdot C_2$$
$$C_4 = G_3 + P_3 \cdot C_3$$

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$
$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$
$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$





2’s complement representation

- Represent positive 2’s complement numbers as simple binary
- Represent negative 2’s complement as a binary that when added to a positive number of same magnitude equals 0

Signed	Unsigned	2’s complement
3	3	0000 0011
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101

2’s complement examples

- Addition

```
5 + (-3)
     0000 0101 = +5
+    1111 1101 = -3
------------------------
      0000 0010 = +2
```

SIGN EXTENSION

Signed -1 in 16-bits is 1111 1111 1111 1111

Signed +1 in 16-bits is 0000 0000 0000 0001

- Subtraction

```
7 – 12 = 7 + (-12)
     0000 0111 = +7
+    1111 0100 = -12
---------------------------
     1111 1011 = -5
```

Other things to note

- Use shifter designed in problem 1

- Shift amount is represented by lower 4 bits of input B

  Input to be shifted is A

  ```
  a1 = invA ? (~InA) : InA;
  b1 = invB ? (~InB) : InB;
  ```

- Take care of OFL -> Keep track of sign bit and ‘cout’ bit

- Underflow is don’t care

WHAT TO SUBMIT

- ELECTRONIC
  - Verilog code of all modules and testbench
- MANUAL
  - Neat Schematic of the design (hand-drawn is fine)
  - Annotated waveforms
  - Explain why you chose a set of inputs for your simulation (in 3 or 4 sentences)

Problem 3 and 4

- Translate to MIPS

- For problem 4, after translating to MIPS, you might have a structure similar to this:

```
XXX
YYY
Loop: PPP
      QQQ
      RRR
Total = 2 + 3*(number of times loop is executed)
```

- Memory references for problem 4?? -> Think ☺

Problem 5

- Total number of instructions is given
- Also mentioned is the % distribution of each instruction along with the number of cycles an instruction takes to execute
  - Overall CPI = (40*2 + .. + .. + ..) / total

  - Old number of multiplies = 8% of 200 = x
  - 50% of the old number is replaced by shift-add that takes 3.5 cycles each
  - New number of multiplies = x * 50% = y
  - Additional ALU instructions = x * 50% * (length)

  - Calculate new total number of instructions and the new CPI

Problem 6

- Find the maximum IPC
  - IPS = IPC * clock speed

- Average CPI

  = (2A+B+C+D+E)/(2+1+1+1+1)

  Find results for both P1 and P2

  Speedup (P2) / Speedup (P1) = Avg. time per instruction on P1 / Avg. time per instruction on P2

  For P1, average CPI / 4GHz and for P2, average CPI / 6GHz

- Find out that frequency where ratio above is 1

Problem 7

- CPI = (CPU time × clock rate)/No. instr.

  CPU time and clock rate is given

  Number of instructions = 0.85 * (Data in problem 1.12)

- Clock rate ratio = New clock / old clock

  $(\text{CPI})_{4\text{ GHz}}$ / $(\text{CPI})_{X\text{ GHz}}$ for a and b

  Why do you think they are dissimilar ? CPU time ?

- New execution time / old execution time

  CPU time reduction = [1 – (New exec time / old exec time)] * 100 = ?? %