Extra Credit Project

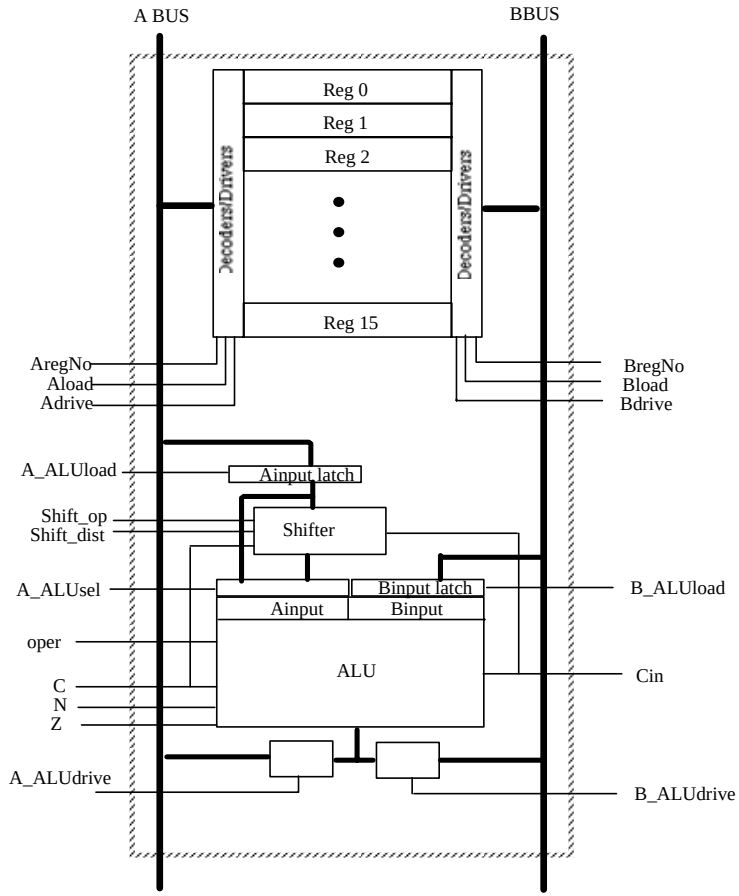
Modification that can be done to the datapath.

Extra Credit

- One writeup – 2 parts for extra credit
- Each part can raise your final grade 1 ½ %
- Part 1 – Add a shifter to the datapath
- Part 2 – Beyond integrating a shifter, use it to do integer multiplication (8-bit by 8-bit for a 16-bit result)

The architecture

- Block diagram
- Dashed line shows interface
- Note added control signals and location of shift unit



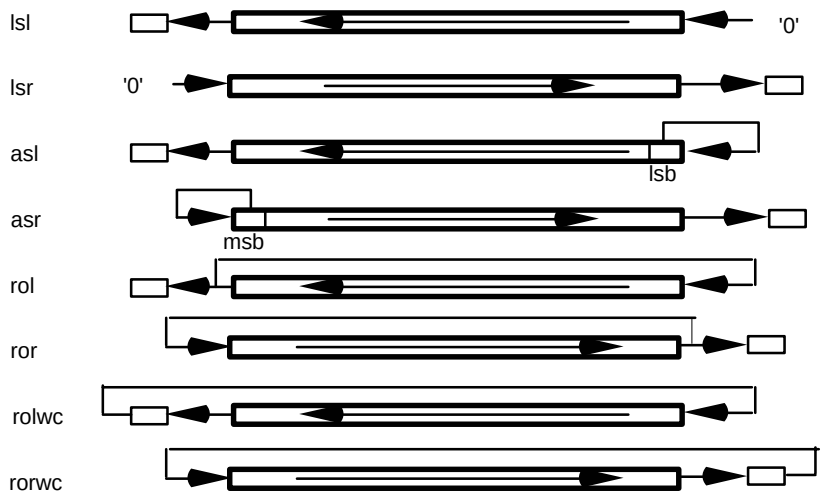
From the writeup

Add a shifter. The shifter is connected to the A input latch so it can only be loaded from the ABUS. The A_ALUsel controlls a multiplexer that can either select the latched input or the shifter output. A value of ‘1’ on A_ALUsel selects the latch input, and a value of ‘0’ selects the shifter output.

The possible shift operations are logical shift left, logical shift right, arithmetic shift left, arithmetic shift right, rotate left, rotate right, rotate left with carry, and rotate right with carry. The mnemonics are (lsl,lsr,asl,asr,rol,ror,rolwc,rorwc). Their effected is illustrated below. In the following diagram the large rectangle is the register and the small square represents the C bit of the Flags

The operations

- Typical shift operations
  - Logical shifts
    - Left, right
  - Arithmetic shifts
    - Left and right
  - Rotates left and right with carry and without carry



The writeup continued

The output of the shifter goes into the A input selector multiplexer. The shfiter will always affect the C bit of the Flags as shown above. If the ALU operation is op_A and the ALU A input is selected to be the shifter, then the carry output of the ALU/shifter is determined by the shift operation. Any other ALU operation will have the effect of overriding the C bit generated by the shifter and causing the C bit to be set accoring to the ALU operation.

The N and Z flags are set accoring to the ALU_operation. Note that the result of a shift operation will pass through the ALU using the ALU operation op_A. This will cause the Z and N flags to be set and the C bit will come from the shifter. During other types of operations, such as a multiply step, the ALU operation of the step will determine the state of the flags.

Continued

- The testbench for this will be on the web and has been modified to include the shift operation.

1) So your first task is to modify your datapath to include a shifter that can perform the above functions and integrate it into the datapath as shown. The testbench for this step produces the appropriate select signal for the A_ALUsel signal. It runs through the same tests that it did in the previous project step and then runs through step to test the correct operation of the shfiter.

You will need a type for the shifter operations in your support package. The testbench assumes that the mnemonic names above are used and the type will be called shift_operations

The second part

- This part is a little more challenging
- You modify the control by writing a procedure like shiftop to do integer multiplicaiton
- The source for the operands are s1reg and s2reg with the result put back in s2reg
- You will need a multiply control register for one of the operands which is already set up.

More info from the assignment

2) Write a microcode procedure to do integer multiplication. By a microcode routine I mean a routine such as the tworegop, oneregop, shiftop procedures. Your routine will be passed the multiplier and multiplicand register numbers, s1reg, s2reg, and put the result back in register s2reg. On the first cycle you will have to load the multiply control register (already set up in the bus_cycle procedure) and then use its values to do the multiply.
Other information:
The operation multiplies the 8 lsb of each resiter and puts the 16 bit result in the 2nd reg.
The operation should take a constant number of cycles.
The output of the ALU can be driven on the B bus and latched back into the B input, acting as an accumulator.

cont

Enter calls to your procedure as indicted to do the following 8 multiplications as indicated in the testbench.

R0 x R1 --> R1 R2 x R3 --> R3 R4 x R5 --> R5
R6 x R7 --> R7 R8 x R9 --> R9 R10 x R11 --> R11
R12 x R13 --> R13 R14 x R15 --> R15

This is to test your modification

The bus cycle procedure

- This procedure runs the busses and internal transfer of data

The call to PROCEDURE bus_cycle is

bus_cycle(
A_regldi - **TBLoad** or **Load** - Load value on ABUS into register Aregno
Drive - Drive Aregno value onto ABUS
Idle or Accum - Register do not drive or load from ABUS
A_regno -
A_aluldi - **Load** - Latch value from ABUS for Ainput of ALU
Drive - Drive value of ALU output onto ABUS
Idle, **TBLoad**, **Accum** - ALU does not latch or drive ABUS
A_aluinsel - '1' - A input to ALU is latched value, '0' shifted value
A_BUS_VAL - Value placed on ABUS by test bench if A_regldi = **TBLoad**
A_BUS_EXP - Expected value on ABUS
B_regldi, B_regno - same as for ABUS side but for BBUS
B_aluldi - same as ABUS side except
Accum causes ALU to drive BBUS and then Latch that value.
B_BUS_VAL, B_BUS_EXP - same as for ABUS but for BBUS
Shf_op - the shift operation to be performed - if the shifter is not used it can be any value
Shf_dist - the distance of the shift - if the shifter is not used it can be any value
MCR_ldi - multiply control register loading - if Load, MCR is loaded from the 8 lsb of the A bus, otherwise no action.
alu_op - the alu_operation to be performed - if the alu is not used can be any value
Cinval - the value of the carry input to the alu and shifter this cycle
exp_cc - the index for the expected condition code
checkbus - if TRUE then the value driven on the bus during Drive signals being low is checked as well as the flags are checked.
if FALSE the actual value on the bus is not checked.
)

Example of use of procedure bus_cycle

For example the two register operation does the following

Cycle	ABUS	BBUS	Aalu insel	AluOP	Cin	Shift OP	Shift Dist
1	R(Aregno)->Aalu	R(Bregno)->Balu	‘1’	oper	Cinval	--	--
2	idle	aluout->R(Bregno)	‘1’	oper	Cinval	--	--

and has calles to the bus_cycle procedure as seen in the testbench.

To do a shift operation, the following is needed.

Cycle	ABUS	BBUS	Aalu insel	AluOP	Cin	Shift OP	Shift Dist
1	R(regno)->Aalu	Idle	‘0’	op_A	Cinval	sop	dist
2	idle	aluout->R(Bregno)	‘1’	op_A	Cinval	sop	dist

Set up a similar table for the multiply operation which will make coding the procedure much easier. Do not check the contents on the bus during operation of your multiply routine, i.e., call bus_cycle with the checkbus parameter as FALSE. Check of the correct opertion is done on the subsequent dump of the registers.