Tutorial #3
RTL & Computer Basic Building Blocks

1. Show the block diagram of the hardware that implements the following register transfer statement:

yT_2: R2 ← R1, R1 ← R2

2. The outputs of four registers, R0, R 1, R2, and R3, are connected through 4-to-1-line multiplexers to the inputs of a fifth register, RS. Each register is eight bits long. The required transfers are dictated by four timing variables T_1 through T_3 as follows:

T_0: R5←R0
T_1: R5←R1
T_2: R5←R2
T_3: R5←R3

The timing variables are mutually exclusive, which means that only one variable is equal to 1 at any given time, while the other three are equal to 0. Draw a block diagram showing the hardware implementation of the register transfers. Include the connections necessary from the four timing variables to the selection inputs of the multiplexers and to the load input of register R0.

3. Represent the following conditional control statement by two register transfer statements with control functions.

If *(P* = 1) then (R1 ← R2) else if (Q = 1) then (R1 ← R3)

4. Draw the block diagram for the hardware that implements the following statements:

x + yz: AR ← AR + BR

where AR and BR are two n-bit registers and *x, y,* and z are control variables. Include the logic gates for the control function. (Remember that the symbol + designates an OR operation in a control or Boolean function but that it represents an arithmetic plus in a microoperation).

5. Consider the following register transfer statements for two 4-bit registers R1 and *R2.*

xT: R1 ← R1 + R2

$x'T$: R1 ← R2

Every time that variable T = 1, either the content of R2 is added to the content of R1 if x = 1, or the content of R2 is transferred to R1 if x = 0. Draw a diagram showing the hardware

implementation of the two statements. Use block diagrams for the two 4-bit registers, a 4-bit adder, and a quadruple 2-to-l-line multiplexer that selects the inputs to R1. In the diagram, show how the control variables x and T select the inputs of the multiplexer and the load input of register R1.

6. The following transfer statements specify a memory. Explain the memory operation in each case.

   a) R2← M[AR]

   b) M[AR] ← R3

   c) R5← M[R5]

7. What is wrong with the following register transfer statements?

   a) xT: AR←$\overline{AR}$, AR← 0

   b) yT: R1←R2, R1← R3

   c) zT: PC←AR, PC←PC+1

8. A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

   a) How many selection inputs are there in each multiplexer?

   b) What sizes of multiplexers are needed?

   c) How many multiplexers are there in the bus?

9. Design a 4-bit combinational circuit decrementer using four full-adder circuits.

10. Design an arithmetic circuit with one selection variable S and two n-bit data inputs A and B. The circuit generates the following four arithmetic operations in conjunction with the input carry C_{in} . Draw the logic diagram for the first two stages.

S	$C_{in} = 0$	$C_{in} = 1$
0	D = A+B (add)	D = A+1 (increment)
1	D = A-1 (decrement)	D = A+$\overline{B}$+1 (sub)

11. Design a digital circuit that performs the four logic operations of OR, NOR, AND, and NAND. Use two selection variables. Show the logic diagram of one typical stage.

12. The 8-bit registers AR, BR, CR, and DR initially have the following values:

AR= 11110010
BR= 11111111
CR= 10111001
DR =11101010

Determine the 8-bit values in each register after the execution of the following sequence of microoperations.

AR←AR + BR	add BR to AR
CR ←CR ∧ DR, BR ← BR + 1	AND DR to CR, increment BR
AR←AR – CR	subtract CR from AR

13. An 8-bit register contains the binary value 10011100. What is the register value after arithmetic shift right? Starting from the initial number 10011100, determine the register value after an arithmetic shift left, and state whether there is an overflow.

14. Starting from an initial value of R = 11011101, determine the sequence of binary values in R, after a logical shift-left, followed by a circular shift-right, followed by a logical shift-right and a circular shift-left.