FACULTY OF ELECTRONIC ENGINEERING & TECHNOLOGY

NMJ20404/EKT221

DIGITAL ELECTRONICS II

LAB 3

REGISTER TRANSFER LEVEL

LAB 3: REGISTER TRANSFER LEVEL

INTRODUCTION

Register Transfer Level (RTL)

Digital systems that consist of a set of registers may execute operations such as *shift*, *count*, *clear* and *load*. The information flow and operations on the data stored inside the registers are often referred to as ***Register Transfer Level (RTL)***. Three main components are specified when dealing with digital systems at the register transfer level:

- The set of registers in the system
  - The hardware that will have the ability to perform the elementary operations from the stored data such as *shift*, *count*, *clear* and *load*.

- The operations that are performed on the data stored in the registers.
  - Elementary operations or better known as micro-operations such as: *R1 ← R1+R2*

- The control that supervises the sequence of operations in the systems
  - The control that initiates the sequence of events to perform the micro-operations in a prescribed manner.

Table 1 shows the types of register micro-operations.

Table 1: Types of micro-operations

Micro-operation	Example
Transfer	R0 ← R1
Arithmetic	R0 ← R1 + 1
Logic	R0 ← R1 ^ R2
Shift	R0 ← *sl* R0

Meanwhile, the Verilog symbols that can be used for register transfers are listed in Table 2.

Table 2: Verilog symbols for register transfers

Operation	Text RTL	Verilog
Addition	+	+
Subtraction	-	-
Bitwise AND	∧	&
Bitwise OR	∨	|
Bitwise XOR	⊕	^
Bitwise NOT	‾ (overline)	~
Shift left (logical)	Sl	<<
Shift right (logical)	Sr	>>
Vectors/registers	*A*(3:0)	*A*[3:0]
Concatenation	‖	{ , }

Multiplications and divisions are not listed as basic sets of micro-operations even though they are represented by the symbols '*' for multiplications and '/' for divisions. They are assumed to be implemented by the *shift* and *add* micro-operations.

The register is usually represented by the variable 'R' and may include multiple 'R's which signifies the usage of more than one set of registers to meet the desired specifications of a digital system. The controller section of a digital system is usually represented by the variable 'K' to show the conditional statements of the system (control signals). A system may have more than one 'K' controller to meet the purpose of multiple transfer operations.

An important note is that all registers have a load enable controller (LE) to allow data to be stored into the destination register. The following example shows a particular RTL micro-operation that implements the rule.

Example

An RTL for an *n*-bit adder/subtractor system can be represented as the following RTL micro-operations:

$$\overline{K_2}K_1 : R_1 \leftarrow R_1 + R_2 \; ;$$
$$K_2K_1 : R_1 \leftarrow R_1 + \overline{R_2} + 1$$

The block diagram for the two statements above is shown in Figure 1.

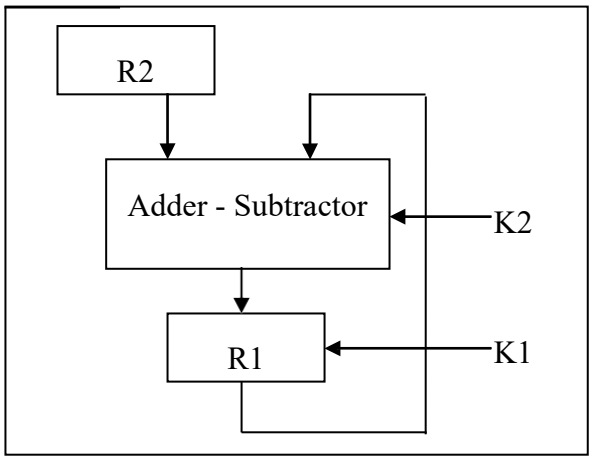

Figure 1: Block Diagram of an *n*-bit adder/subtractor

The diagram shows that K_2 is the selector for the add/subtract operation and K1 is the load enable for register R1. When K2=0 and K1 = 1, then R2 will be added to the contents of R1 and stored back into R1. When K2=1 and K1=1, then the R2 will be subtracted from the contents of R1 and stored back into R1.

For this example, the carry bit and overflow operations are not represented and left out but can still be constructed if desired.

Note that RTL operations are always based on the edge transitions of clock pulses as registers originate from flip-flops.

```
module add_subtract (clk, K1, K2, R1, R2, R1_out);

input clk, K1, K2;
input [3:0] R1;
input [3:0] R2;
output reg [3:0] R1_out;

always @(posedge clk) begin
    if (K1) begin
        if (K2) begin
            R1_out = R1 - R2;
        end
        else begin
            R1_out = R1 + R2;
        end
    end
    else begin
        R1_out = R1;
    end
end

endmodule
```

Figure 2: Verilog code of block diagram in Figure 1

PRE-LAB ASSIGNMENT

Students are advised to write, compile, and simulate the Verilog code in Figure 2 by using Quartus before coming to the lab. Next, students should be able to analyse the output waveform based on their understanding of the simulated code.

TASK / ASSIGNMENT

1. Write a Verilog code for a digital system that uses a 4:1 MUX with the following RTL function:

   C0 : R4 ← R0,
   C1 : R4 ← R1,
   C2 : R4 ← R2,
   C3 : R4 ← R3

   Compile and simulate the code to verify the output of the digital system.

2. A digital circuit consists of two 4-bit registers named R1 and R2 that implements the following statements:

   C0 : R2 ← 0;
   C1 : R2 ← $\overline{R2}$;
   C2 : R2 ← R1;

   Write, compile and simulate a Verilog code for the circuit.

3. A digital system has an *n*-bit register with S1 and S0 control inputs. Write a Verilog code for the following function table:

a.

S1	S0	Register Operation
0	0	Load parallel data
0	1	Clears register to 0
1	0	Complement output
1	1	No change

b.

S1	S0	Register Operation
0	0	Load parallel data
0	1	Shift left
1	0	Shift right
1	1	No change

Next, compile and simulate the code.