ECE2330 – Digital Logic Design

Control Signals Logic

Learning Objective

After this activity, you will be able to

- Design circuits to generate control signals using analytical digital design techniques, and
- Verify that the implementation you specify satisfies the requirements

Problem Statement

For this learning activity, you are **required** to design circuits to generate the control signals for the Central Processing Unit (CPU) architecture shown in Figure 1. The signals are listed in Table 1, and the arithmetic operations defined by F0 and F1 are listed in Table 2.

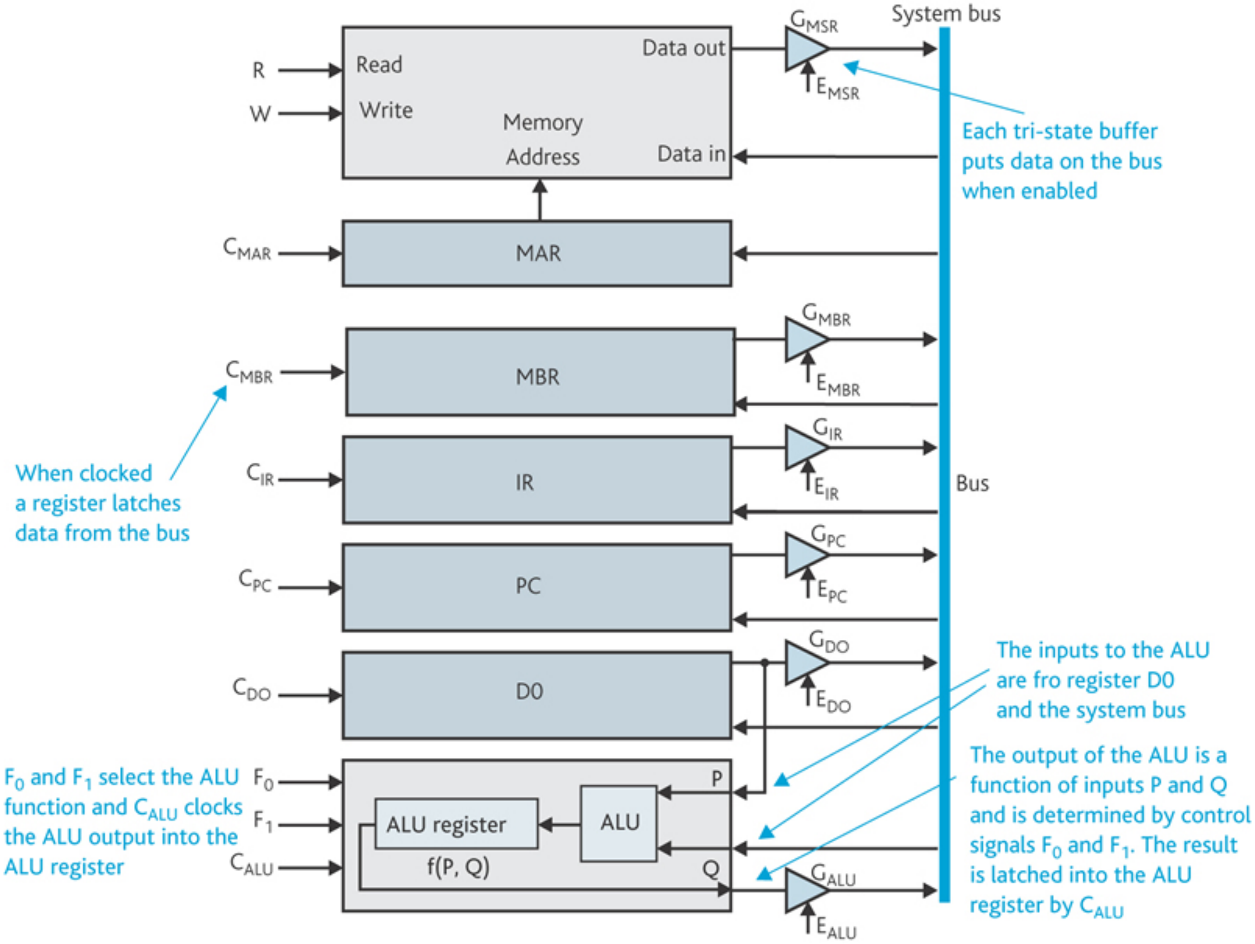


Figure 1: Simple CPU Architecture

The architecture contains the Memory Address Register (MAR), the Memory Buffer Register (MBR), the Instruction Register (IR), the Program Counter (PC), Data register 0 (D0), and an Arithmetic and Logic Unit (ALU).

Table 1: Simple CPU Control Signals

Signal	Type	Operation
R	Memory Control	Read from memory
W	Memory Control	Write to memory
C_{MAR}	Register Clock	Clock data into MAR
C_{MBR}	Register Clock	Clock data into MBR
C_{PC}	Register Clock	Clock data into PC
C_{IR}	Register Clock	Clock data into IR
C_{D0}	Register Clock	Clock data into D0
C_{ALU}	Register Clock	Clock data into ALU register
E_{MSR}	Bus Control	Enable data from memory onto system bus
E_{MBR}	Bus Control	Enable data from MBR onto system bus
E_{PC}	Bus Control	Enable data from PC onto system bus
E_{IR}	Bus Control	Enable data from IR onto system bus
E_{D0}	Bus Control	Enable data from D0 onto system bus
E_{ALU}	Bus Control	Enable data from ALU register onto system bus
F_0	ALU Control	Select ALU function, bit 0
F_1	ALU Control	Select ALU function, bit 1

Table 2: Arithmetic Functions Define by F0 and F1

F_1	F_0	Operation
0	0	Add P to Q, P + Q
0	1	Subtract Q from P, P – Q
1	0	Increment Q, Q + 1
1	1	Decrement Q, Q – 1

The table in Figure 3 provides the Register Transfer Level (RTL) operations, referred to as *microinstructions*, which indicate the sequence of operations that must occur for a given instruction to be executed. The sequencing of these microinstructions occurs with the timing signals, T0–T7 (the circuit to generate these will be designed in a subsequent learning activity). For example, the 4 microinstructions associated with the fetch operation (which occur for every instruction execution) are:

- MAR ← PC: The content of the PC is loaded into the MAR
- IR ← [MAR]: The memory content at the address contained in MAR is loaded into the IR
- ALU (Q) ← PC: The content of the PC is provided as the Q input to the ALU
- PC ← ALU: The output of the ALU is then loaded into the PC

Likewise, the subsequent microinstructions required to complete each instruction execution are also listed in the table in Figure 3.

To verify your design, you will create a component like the one shown in Figure 2, with simulation results shown in Figure 4.

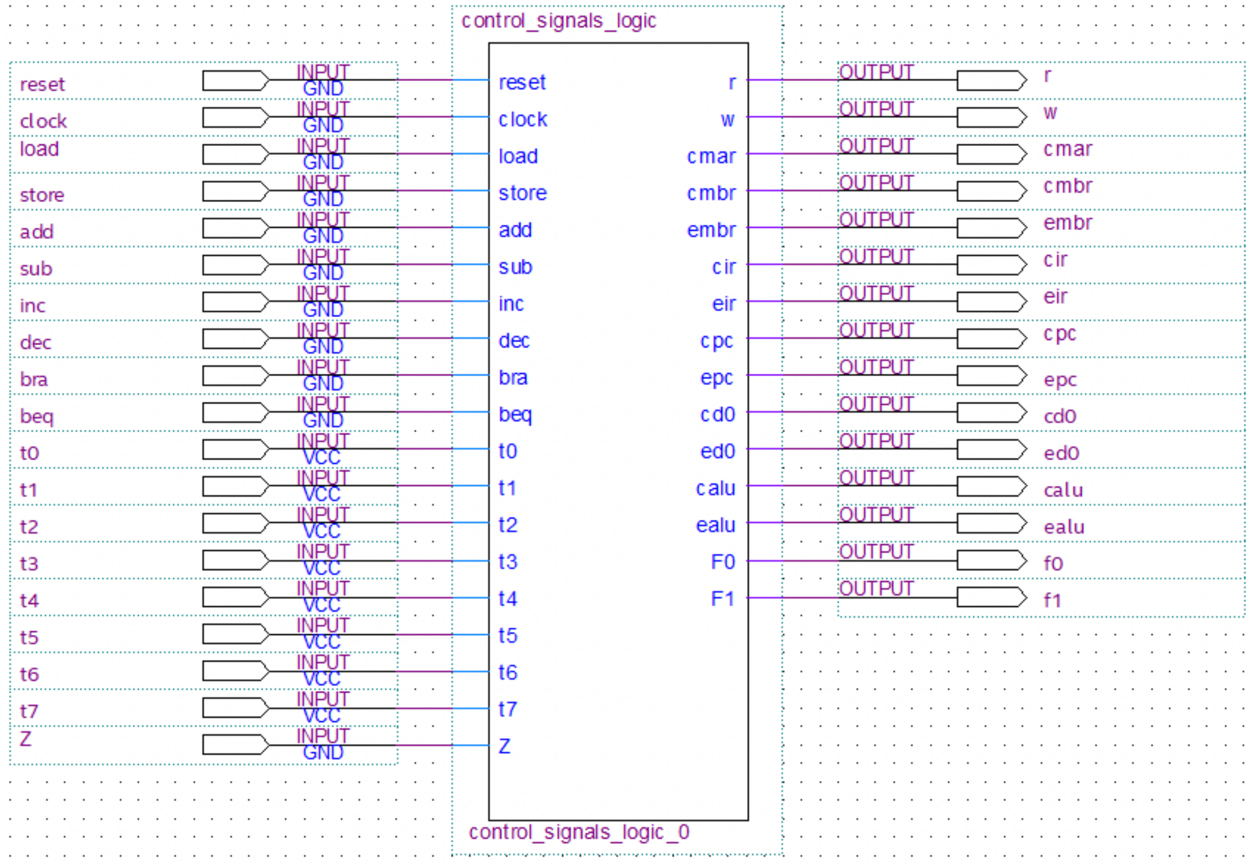


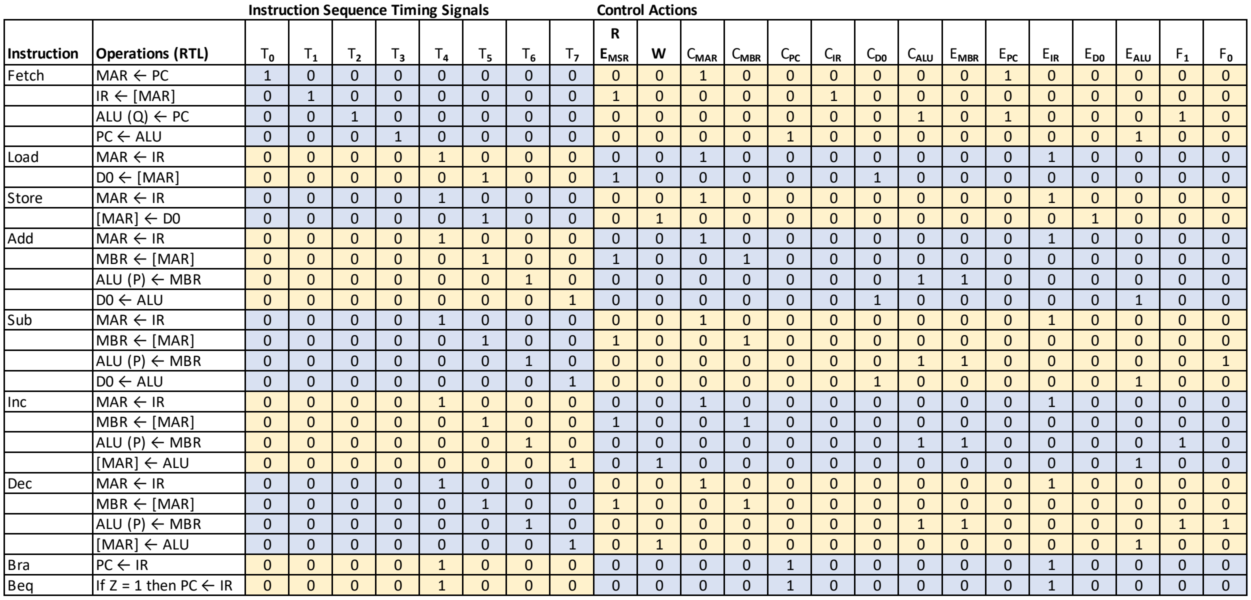
Figure 2: Controls Signals Logic Component

Note that the Read signal and the enable signal, EMSR, essentially represent the same signal, as shown in Figure 3. Thus, only one of them is represented in the component circuit shown in Figure 2.

Also note the input signal, *Z*, is an output from the ALU and indicates whether or not the ALU operation produced a result equal to zero. *Z* = 1 indicates that the ALU operation result equals zero; *Z* = 0 indicates otherwise. This signal is used by the **Beq** (Branch Equal to Zero) instruction to determine if the contents of the IR (where the lower 5 bits are an address) are loaded into the PC, thus executing what is known as a *conditional branch*. Conversely, the **Bra**nch instruction is known as an *unconditional branch*.

Because *Z* is used by a subsequent instruction, your controls signal logic design should include a simple sequential logic circuit to store the value of Z after any ALU operation during the T6 instruction sequence time period (that is, when T6 = 1).

ECE2330 – Digital Logic Design



Instruction Sequence Timing Signals — Control Actions

Instruction	Operations (RTL)	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	R E_{MSR}	W	C_{MAR}	C_{MBR}	C_{PC}	C_{IR}	C_{D0}	C_{ALU}	E_{MBR}	E_{PC}	E_{IR}	E_{D0}	E_{ALU}	F_1	F_0
Fetch	MAR ← PC	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
	IR ← [MAR]	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	ALU (Q) ← PC	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
	PC ← ALU	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
Load	MAR ← IR	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	D0 ← [MAR]	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Store	MAR ← IR	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	[MAR] ← D0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
Add	MAR ← IR	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	D0 ← ALU	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
Sub	MAR ← IR	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
	D0 ← ALU	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
Inc	MAR ← IR	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
	[MAR] ← ALU	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Dec	MAR ← IR	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	MBR ← [MAR]	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	ALU (P) ← MBR	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1
	[MAR] ← ALU	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Bra	PC ← IR	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
Beq	If Z = 1 then PC ← IR	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0

Figure 3: Instruction Sequencing and Control Signals

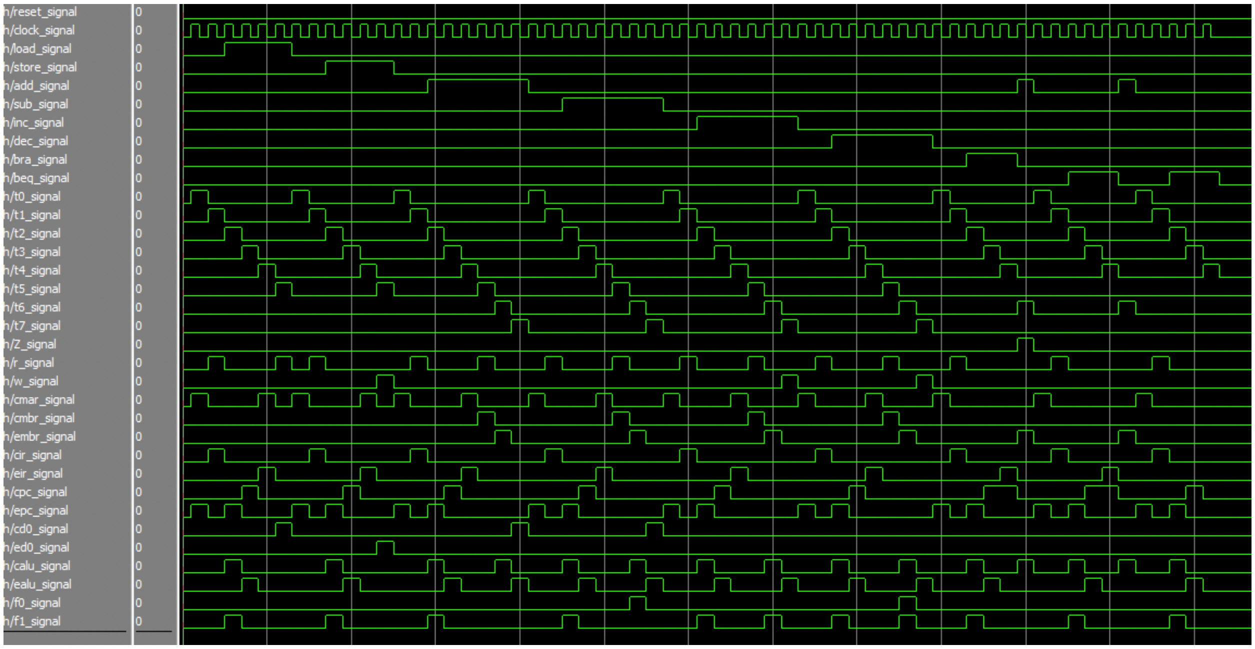


Figure 4: Simulation Results

What To Submit

You will submit the VHDL file you create as part of this assignment, along with simulation results for verification (as a PDF file, annotated appropriately where necessary).

Grading Rubric

This assignment is worth a total of 10 points:

- VHDL file of controls signals logic and testbench
- Numerical Verification