CE1901 LABORATORY PROJECT

SUMMARY

Engineers use hardware description languages to describe logic circuits using text rather than pictures. Two hardware description languages in widespread use are VHDL (IEEE Standard 1076) and Verilog (IEEE Standard 1364). Hardware description languages describe digital logic using assignment and conditional statements that fully describe the truth table behavior of a component. These descriptions give engineers great flexibility because, as a design changes, the changes can be reflected textually rather than requiring extensive schematic re-wiring. Computer-aided design tools convert hardware descriptions into circuit designs for implementation as custom integrated circuit chips or as designs configured in field programmable gate arrays (FPGAs).

All electrical and computer engineering students at MSOE learn the VHDL hardware description language. These laboratory exercises focus on the use of VHDL to describe a four bit carry look ahead adder.

PRELIMINARY READING

Addition is the fundamental arithmetic operation. Addition of binary numbers A and B relies on a basic component called a full-adder. The full-adder adds one column of A and B to produce a sum bit and a carry bit. An n-bit adder is implemented using n full-adder components.

Addition algorithms differ by how they generate the carry bits. The simplest scheme directly matches the paper and pencil approach. This scheme ripples the carry from one column to the next. Unfortunately, this ripple-carry adder (RCA) is slow because carry energy must ripple through all n stages of the number. Thus, the time complexity of this algorithm is linear. We say that the time complexity is “order n” or “big-O n” and write the complexity in mathematical symbols as **O(n)**.

Carry look-ahead addition, on the other hand, uses algebra to expand the iterative carry equations to equations that only depend on C0, the carry that arrives with A and B at the adder inputs. These expanded logic equations use AND and OR gates to calculate all carries simultaneously and remove the ripple dependency. Thus, the time complexity of a carry look-ahead adder (CLA) is constant ***provided gates can have any number of inputs***. This is written in big-O notation as O(1). This constant time algorithm is not dependent on the number of bits. It sacrifices semiconductor space for speed.

Carry-select adders compromise by using ripple carry-adders to save on gate space. Consider an 8-bit carry-select adder. The lower nibble is added using one ripple-carry. Due to the gate delays of the full adder, the carry into column four (C4) will be stable after four full-adder delays. The upper nibble does not wait for this carry to arrive. Rather, two full adders add both possible results. One full adder adds the upper nibble assuming that C4 will be 0 and the other adds the upper nibble assuming that C4 will be 1. Both results wire to a bus multiplexer. When C4 arrives, C4 energizes the select signal of the bus multiplexer to pass the correct upper nibble as the output. This technique breaks the linearity of ripple carry addition because the upper nibble does not wait. Complexity analysis for n-bit carry-select adders gives O($\sqrt{n}$).

Many other addition algorithms exist. All algorithms support subtraction by using XOR gates to pass $\bar{B}$ to each full adder when C0 = 1. This results in –B presented as the second input.

This laboratory is design of carry look-ahead addition. Figure 1 documents the project hierarchy by showing the name of each VHDL entity and its type of architecture.

**These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.**

CE1901 LABORATORY PROJECT

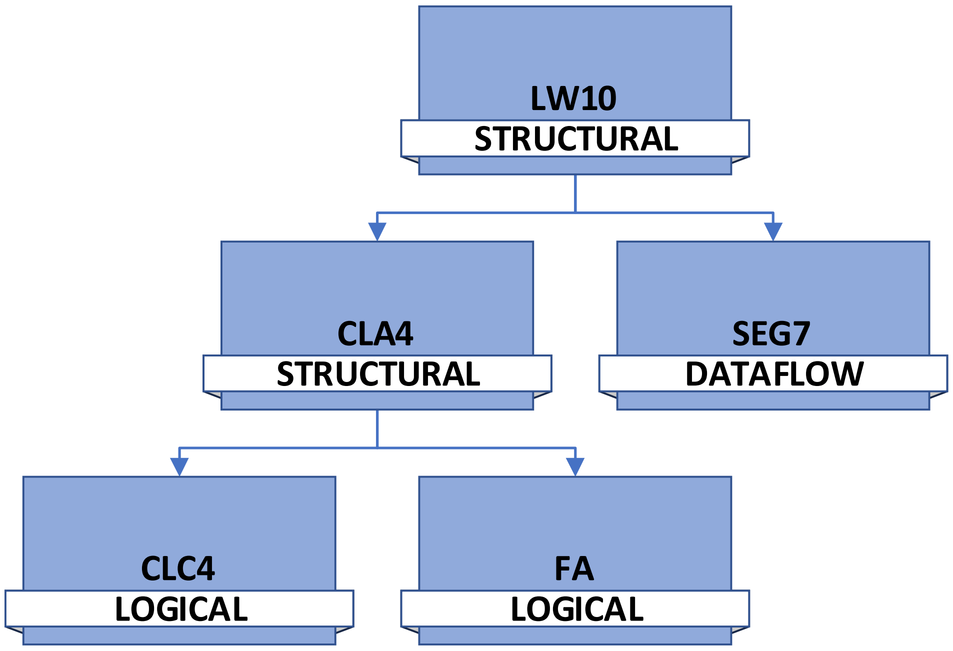

Figure 1: Project Hierarchy Diagram

The diagram shows a top-level entity named LW10 that relies on a carry lookahead adder to complete addition and a seven-segment display decoder to display the result to the user. The diagram also shows that the carry lookahead adder relies on a carry lookahead circuit and full adders to implement addition.

VHDL descriptions can be done in a variety of ways. There are four standard types of architectures that are written.

- Logical architectures describe equations using gates such as NOT, AND, OR, and XOR.
- Dataflow architectures describe truth tables using with-select statements (multiplexer implementations) or when-else statements (comparators).
- Structural architectures interconnect components using port map statements.
- Behavioral architectures include memory behavior to provide computation and storage through time.

This class does not use memory behavior in circuits and thus the first three types of architecture best map to problems in CE1901. As Figure 1 shows, this laboratory requires practice of all three types.

Figure 2 shows the interconnection of components in the top level LW10 entity.

**These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.**

CE1901 LABORATORY PROJECT

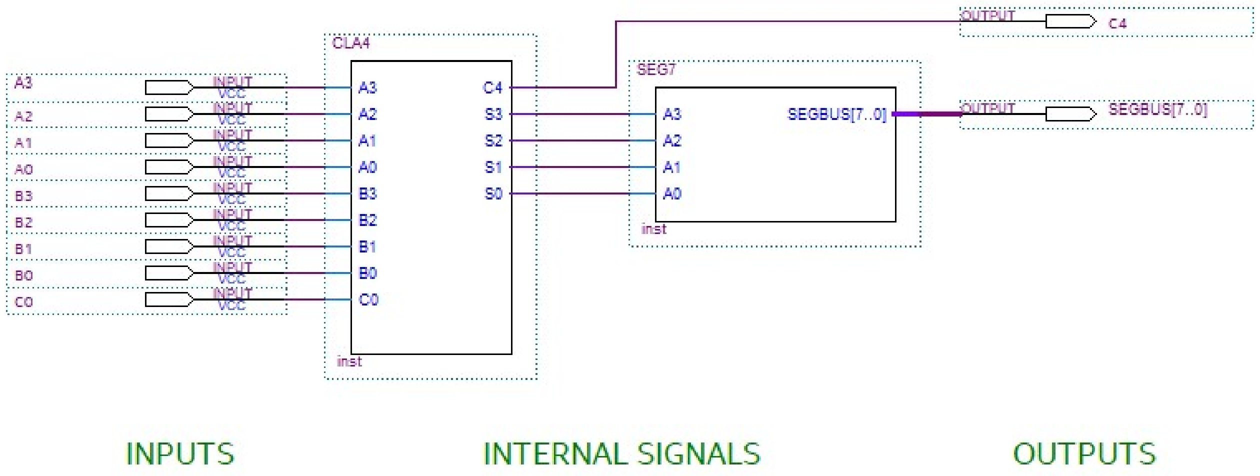

Figure 2: Top Level Structural Architecture LW10

The four internal signals in Figure 2 are declared in structural VHDL using **signal statements**. Signal statements are placed between the architecture statement and the begin delimiter.

```
architecture STRUCTURAL of LW10 is
      signal S3, S2, S1, S0 : std_logic;
begin
U1: CLA4 port map( … );
```

The CLA4 component also contains internal signals between the CLC4 and the associated full-adders. These internal signals will need to be similarly declared and port mapped. Another interesting challenge in the CLA4 structural architecture is the absence of connections to the carry outputs of the full adders. The VHDL port map statement provides the ability to state that this connection should be left as an **open pin** not connected to any other device.

```
-- using positional association to the FA component statement for port maps
architecture STRUCTURAL of CLA4 is
      signal C3, C2, C1: std_logic;
begin
U0: FA port map(A0,B0,C0,open,S0);
```

```
-- using name-to-name association for port maps
architecture STRUCTURAL of CLA4 is
      signal C3, C2, C1: std_logic;
begin
U0: FA port map(A0=>A0,B0=>B0,C0=>C0,COUT=>open,S0=>S0);
```

**These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.**

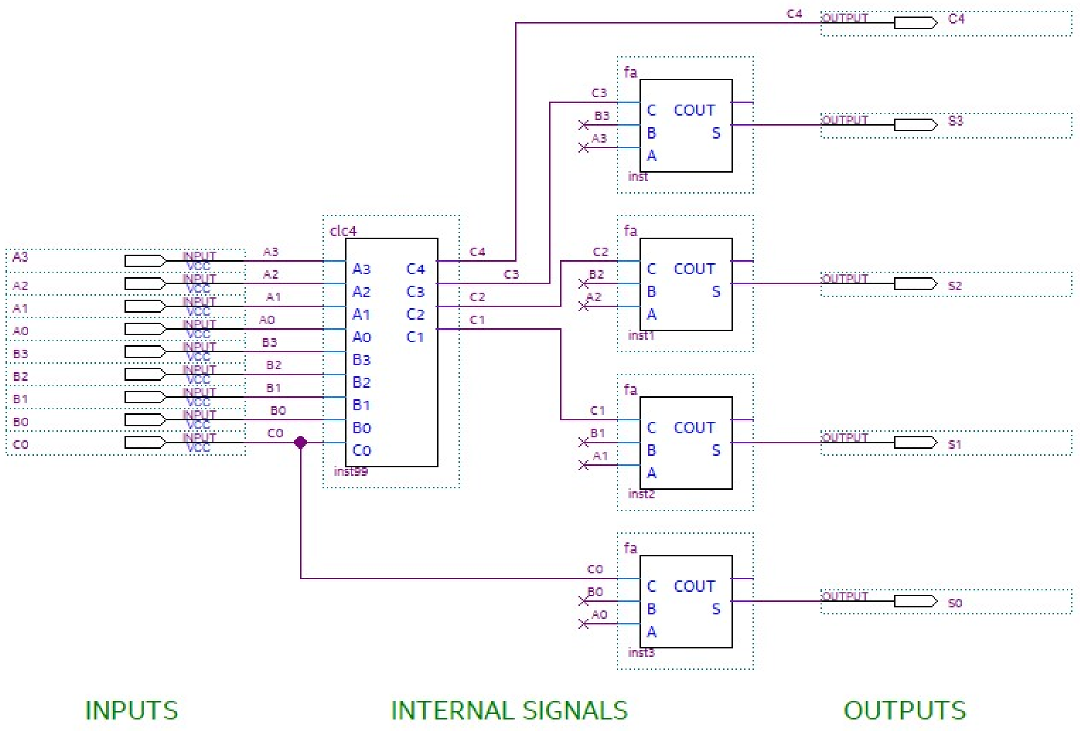

Figure 3: CLA4 Structural Architecture

Figure 3 shows the structural architecture of the CLC4 component. The figure uses physical wires to show that least-significant full adder will port map directly to input C0 while all other full adders will port map to internal signals generated by the carry lookahead circuit. The figure also demonstrates that the carry lookahead circuit will also directly port map C4 to the CLA4 output pin. Finally, the **open pin** connections on all full adders is visible.

The project requires a seven-segment display decoder. Seven-segment displays are standard output devices used to display data in human readable form. Figure 4 illustrates the hexadecimal characters in one standard font called DSEG7.

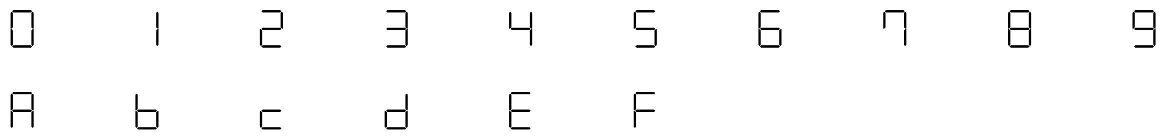

Figure 4: The DSEG7 Font Hexadecimal Characters

These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.

CE1901 LABORATORY PROJECT

Seven-segment displays differ by the logic level that lights segment LEDs. **Common-anode displays** use logic-0 to light LEDs while **common-cathode displays** use logic-1 to light LEDs. Engineers must read the data sheet for the display and control its LEDs using the right logic-level. The DE10-Lite user manual notes that its six seven-segment displays are **common-anode** (Section 3.4, page 28).

Each LED in the seven-segment display corresponds to one bit of the binary data that controls the light behavior. Figure 5 shows the mapping for the DE10-Lite seven-segment displays.

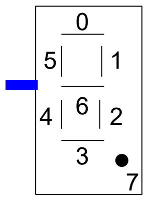

Figure 5: Mapping Data Bits to DE10-Lite Seven-segment Displays

Consider implementing the DSEG7 font number 7 as a binary number. The common-anode behavior of the DE10-Lite seven-segment display means that the control pattern would be created as shown in Table 1.

Table 1: Creating DSEG font 7 on the DE10-Lite Common-Anode Seven-segment Display

BIT 7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
OFF	OFF	ON	OFF	OFF	ON	ON	ON
1	1	0	1	1	0	0	0

PRELABORATORY EXERCISES

1. **Create** a new Quartus project.
   a. **Use** File → New Project Wizard
   b. **Name** the project **lw10**.
   c. This will be the only project you create.
   d. All design files will be stored in the same project.
2. **Create** a full adder entity as a logical equation architecture in the same project.
   a. **Use** File → New → VHDL File
   b. **Include** and **use** the IEEE multi-valued logic standard (**std_logic_1164**).
   c. **Name** the entity **FA.**
   d. **Describe** the functional block symbol (**entity**) and gate-level circuit (**architecture).** This circuit implements the full adder using AND, OR, and XOR gates.
   e. **Save** your file as **fa.vhd.**
   f. **Use** Processing → Analyze Current File. This will check your VHDL syntax for initial errors. **Make** corrections if needed before proceeding.
3. **Create** a four bit carry look-ahead circuit entity as a logical equation architecture in the same project.
   a. **Use** File → New → VHDL File.
   b. **Include** and **use** the IEEE multi-valued logic standard (**std_logic_1164**).
   c. **Name** the entity **CLC4**.
   d. **Describe** the functional block symbol (**entity**) and gate-level circuit (**architecture**). This circuit implements the carry look-ahead equations C1, C2, C3, and C4 using AND and OR gates.
   e. **Save** your file as **clc4.vhd**.
   f. **Use** Processing → Analyze Current File. **Make** corrections if needed before proceeding.

**These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.**

4. **Create** a four bit carry look-ahead adder entity as a structural architecture in the same project.
   a. **Use** Figure 3 as your guide.
   b. **Use** File → New → VHDL File.
   c. **Include** and **use** the IEEE multi-valued logic standard (**std_logic_1164**).
   d. **Name** the entity **CLA4**.
   e. **Describe** the functional block symbol (**entity**) and structural circuit (**architecture)**. This circuit interconnects the CLC4 with four FA components using VHDL port map statements. **Declare** internal signals using the VHDL signal statement.
   f. **Save** your file as **cla4.vhd**.
   g. **Use** Processing → Analyze Current File. **Make** corrections if needed before proceeding.
5. **Complete** this truth table for a **common-anode seven-segment decoder** using the DSEG7 font.

INPUTS				**CHARACTER**	**OUTPUTS**							
A3	A2	A1	A0		SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
0	0	0	0	0								
0	0	0	1	1								
0	0	1	0	2								
0	0	1	1	3								
0	1	0	0	4								
0	1	0	1	5								
0	1	1	0	6								
0	1	1	1	7								
1	0	0	0	8								
1	0	0	1	9								
1	0	1	0	A								
1	0	1	1	b								
1	1	0	0	c								
1	1	0	1	d								
1	1	1	0	E								
1	1	1	1	F								

6. **Create** a seven-segment decoder entity using a with-select dataflow architecture in the same project.
   a. **Use** File → New → VHDL File.
   b. **Include** and **use** the IEEE multi-valued logic standard (**std_logic_1164**).
   c. **Name** the entity **seg7**.
   d. **Describe** the functional block symbol (**entity**) and multiplexer with-select dataflow circuit (**architecture)**. This circuit implements the truth table created in step five of this laboratory. The input standard logic signals must be converted to a bus for easy use in the with-select.

```
with To_StdLogicVector(A&B&C) select
SEGBUS <= X"C0" when X"0",
…
```

   e. **Save** your file as **seg7.vhd**.
   f. **Use** Processing → Analyze Current File. **Make** corrections if needed before proceeding.
7. **Create** the top-level entity as a structural architecture in the same project.
   a. **Use** Figure 2 as your guide.
   b. **Use** File → New → VHDL File.

**These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.**

   c. **Include** and **use** the IEEE multi-valued logic standard (**std_logic_1164**).
   d. **Name** the entity **lw10**.
   e. **Describe** the functional block symbol (**entity**) and structural circuit (**architecture)**. This circuit interconnects the CLA4 and SEG7 components using VHDL port map statements. **Declare** internal signals using the VHDL signal statement.
   f. **Save** your file as **lw10.vhd**.
   g. **Use** Processing → Analyze Current File. **Make** corrections if needed before proceeding.

8. **Prepare** the design for simulation.
   a. **Use** Processing → Start → Start Analysis and Synthesis.
   b. Quartus will begin working through the hierarchy of the project attempting to synthesize the final circuit. During this process, it will check each file for logical errors in the VHDL descriptions, incorrect connections, and other design problems. **Make corrections** if needed.

9. **Simulate** your design in Quartus.
   g. **Use** File → New → University Program VWF.
   h. **Insert** all inputs and outputs. **Be sure to choose the cable named SEGBUS**.
   i. **Group** A3, A2, A1, A0, B3, B2, B1, and B0 into unsigned decimal busses called A and B.
   j. **Place** 16 random values on A and B using the Random Values toolbar icon.
   k. **Select** a region of time on C0 and then use the logic-1 toolbar icon to force carry into FA0.
   l. **Run** functional simulation to verify operation.
   m. **Use** Figure 6 as a guide for what a good simulation would look like.
   n. **Verify** that your design works by checking the segment bus output based on the addition result.

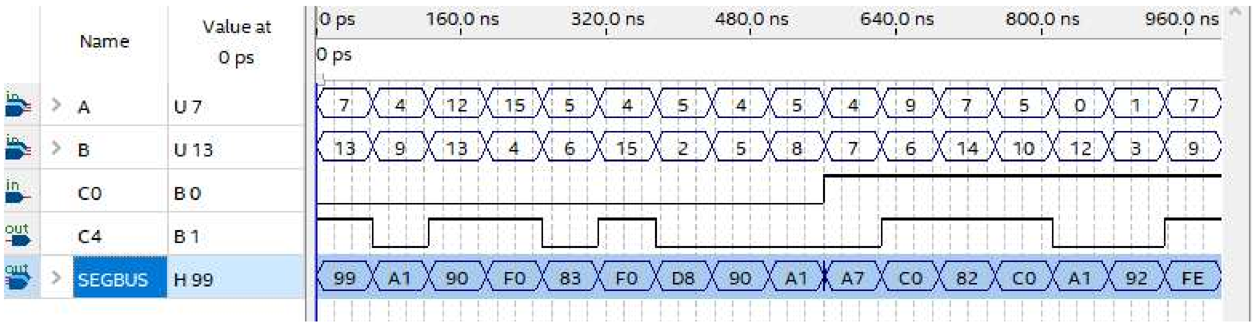
Name
Value at 0 ps
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0 ns
A U 7: 7 4 12 15 5 4 5 4 5 4 9 7 5 0 1 7
B U 13: 13 9 13 4 6 15 2 5 8 7 6 14 10 12 3 9
C0 B 0
C4 B 1
SEGBUS H 99: 99 A1 90 F0 83 F0 D8 90 A1 A7 C0 82 C0 A1 92 FE

Figure 6: A CLA4 simulation

10. **Assign** DE10-Lite I/O slider switches and LEDs to design inputs and outputs as shown in Table 1.

Table 1: Pin Assignments

INPUT	DE10	INPUT	DE10	OUTPUT	DE10	OUTPUT	DE10
A3	SW7	B3	SW3	SEGBUS[7]	HEX07	SEGBUS[2]	HEX02
A2	SW6	B2	SW2	SEGBUS[6]	HEX06	SEGBUS[1]	HEX01
A1	SW5	B1	SW1	SEGBUS[5]	HEX05	SEGBUS[0]	HEX00
A0	SW4	B0	SW0	SEGBUS[4]	HEX04	C4	LEDR9
		C0	SW9	SEGBUS[3]	HEX03		

11. **Compile** and **download** to the DE10.
12. **Test** your design.

These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.

CE1901 LABORATORY PROJECT

DEMONSTRATION AND SUBMISSION

1. **Demonstrate** completed work to the instructor during the lab period.
2. **Submit** laboratory documentation through your instructor's preferred submission method.

**These homework and laboratory exercises are © Dr. Russ Meier, Milwaukee School of Engineering.
All Rights Reserved. Unauthorized reproduction in print or electronic form is prohibited.**