1. Chapter Outline
   1) Arithmetic Circuits

      Arithmetic circuits are fundamental circuits in digital building blocks which are used in computers. In these arithmetic circuits, there are addition, subtraction, comparison, shifts, multiplication, and division. These circuits make the block perform much more complex tasks.

      In the addition arithmetic circuit, there are half adder and full adder on the top level. Half adder adds two inputs and then outputs the result of sum. However, half adder can only output the single digit so that it cannot represent the value 2. To solve this problem, full adder is appeared. Full adder uses carry in and carry output to represent more digits of the output. In full adder, there are ripple-carry adder, carry-lookahead adder, and prefix adder. The ripple-carry adder is the basic adder which can add multiple bits of inputs. However, when the bit of inputs is going larger, the ripple-carry adder performs slow. To solve this problem, carry-lookahead adder uses generate and propagate inputs to reduce the delay. Moreover, prefix adder extends generate and propagate to perform much faster than carry-lookahead adder.

      In the subtraction arithmetic circuit, the inputs are subtracted, and the result of subtraction is an output of the circuit. Most of the parts of the subtracter is similar with the adder. Subtracter uses Boolean equation to change the adder into subtracter so that it connects NOT gate to one of the inputs of the subtractor.

      The comparator compares the input values. It checks the relation of the inputs including greater, less, or equal, and then outputs 0 or 1. Usually, the comparator is implemented by using HDLs including System Verilog.

      The shifters and rotators move the bit left or right and multiply or divide the bit by power of 2. There are logical shifter, arithmetic shifter, and rotator. Logical shifter shifts the number to left or right and fill the empty place of the number with 0, and arithmetic shifter is similar to logical shifter, but it only shifts the number to right and fills the most significant bit by copying the past most significant bit. Rotator rotates the number in circle so that empty places are filled with bits which are shifted off the other end.

      Multiplication and division can be easily implemented with System Verilog and VHDL. In the hardware implementation of multiplication and division, these are implemented by using AND gate, adders, and multiplexer. In case of multiplication, it is implemented by using the combination of AND gates and adders. Moreover, in case of division, it is implemented by using the block which is consisted of adder and multiplexer.

2) Number Systems

Computer uses both integer and fractions. In order to make this possible, number systems are necessary. In number systems, there are fixed-point number systems which represent the decimal numbers, and floating-point number systems which represent scientific notation.

Fixed-number systems imply the binary point between the integer and fraction bit. It uses two’s complement and sign/magnitude notation. Moreover, it depends on the interpretation.

Floating-point number systems avoid the limitation of fixed-number systems, and it allows large range of numbers, very small to very large numbers. The system is divided in three parts: sign, base, and exponent. By using these parts, it is able to represent scientific notation, and use rounding and adding.

3) Sequential Building Blocks

In the digital building blocks, there are sequential building blocks. In sequential building blocks, there are counters and shift registers.

The counter uses clock and reset which resets N-bit of inputs and outputs. As the clock cycles, the counter adds by the N-bit inputs to the counting. The reason to add the inputs to the counting is that the counter uses adder to add the input to the counting. To be specific, the input of the counter is connected to one of the inputs of the adder, and then the output of the adder is connected to the input of Flip-flop. The output of Flip-flop is also connected to the other input of the adder so that it is possible to add the input of the counter to the counting.

The shift register uses clock, serial input, serial output, and N-bit of the parallel output. The shift register shifts the inputs when it is rising edge of the clock, and can be implemented by using N flip-flops. To shift the inputs when it is rising edge of the clock, the shift register uses scan chains technique which adds a test mode where then contents of all flip-flops can be read or loaded in desired values because the flip-flops are connected, scan chain, so that it is able to check the content of all flip-flops.

4) Memory Arrays

By using arithmetic circuits and sequential circuits, it is able to create digital systems which needs memories. In those digital systems, memory array is consisted of register made with flip-flops in order to store small amount of data.

In memory array, there are dynamic random access memory (DRAM), static random access memory (SRAM), and read only memory (ROM). The memory array starts from bit cells. Bit cell stores 1 bit of data. By using this bit cells, it can be extended to the large amount of memories. Moreover, by organizing the bit cells with decoder, it creates memory ports which is necessary to all memories, and memory types make the type of memory array RAM which loses data when power is off and ROM which doesn't lose data when power is off.

In RAM, there are DRAM and SRAM. DRAM stores the data on capacitor so that it is dynamic because the value of data is required to be rewritten periodically after the reading. SRAM uses cross-coupled inverters. Unlike DRAM, SRAM is static because it doesn't have to rewrite the value of data periodically after the reading.

ROM uses transistor to store a bit as it is absence or presence. The contents of ROM are represented by using dot notation. Because of this feature, ROM don't lose data when power is off. By using these features of ROM, it is able to make ROM programmable. In programmable ROM, there are PROM. EEPROM.

5) Logic Arrays

As the memory can be allocated in form of array, logics can be allocated in form of array. There are two types of logic array: programmable logic arrays(PLAs), and field programmable gate arrays (FPGAs). PLAs perform only combinational logics because it is older technology. FPGAs perform both sequential and combinational logic.

PLAs are implemented by using two-level combinational logic in sum-of-products from. In two-level combinational logic, there are AND array followed by an OR array. By the combination of NOT gate, AND array, and OR array, it is possible to perform various combinational logic. Moreover, ROM can be a special case of PLAs because ROM array behaves as OR plane, and decoder behaves as an AND plane.

FPGAs is consisted of an array of reconfigurable gates. By connecting software programming tool, it is able to implement the design on FPGA. FPGA functions as an array of configurable logic elements. Each of logic elements is covered by input/output elements which is for meeting the outside world. This feature makes logic elements connect to other logic elements and input/output elements by using programmable routing channels.

2. Figures
   1) Figure 5.17

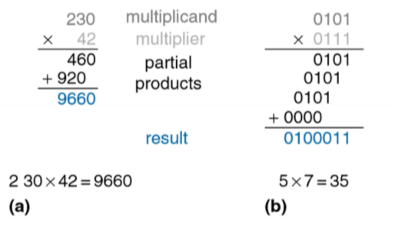

Figure 5.17 Multiplication: (a) decimal, (b) binary

The reason why this figure is selected is that this figure shows each step of multiplication in binary. By understanding how the multiplication works in binary, it can be extended to understanding the division in binary. Thus, figure 5.17 is chosen because it is a starting point to understand and applicate much more complicated concepts and materials.

   2) Figure 5.39

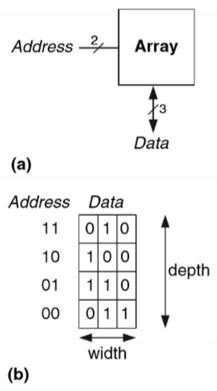

Figure 5.39 4 × 3 memory array: (a) symbol, (b) function

This figure shows symbol and function of 4x3 memory array, and one of the memory arrays. However, starting from the small size of memory array, it can be enlarged to much more complicated type of memory array such as DRAM and SRAM and give help to understand the logic array which appears after this figure. Therefore, figure 5.39 is selected because this is a starting point to understand and applicate much more complicated concept and materials after this figure.

3. Example Problems

1) EX#1 (Example 5.1 Variation)

Compare the delays of a 128-bit ripple-carry adder and a 64-bit carry-lookahead adder with 4-bit blocks. Assume that each two-input gate delay is 150ps and that a full adder delay is 250ps.

⟨Delay of Ripple - carry adder⟩

: $128 \times 250ps = 32ns.$

⟨Delay of CLA⟩

$t_{pg} = 150ps$

$t_{pg\text{-}block} = 6 \times 150ps = 900ps$

$t_{AND\_OR} = 2 \times 150ps = 300ps$

$t_{CLA} = 150ps + 900ps + \left(\frac{64}{4} - 1\right) \cdot 300ps + (4 \times 250ps)$

$= 6.55ns.$

$\frac{32}{6.55} = 4.89$

∴ The CLA is almost 5 times faster than the ripple - carry adder.

2) EX#2 (Example 5.4 Variation)

Compute 0.875 + -0.625 using fixed-point numbers.

i) $0.875 \geq 0.5 \Rightarrow 1$ in the 2^{-1} column

$0.875 - 0.5 = 0.375$

$0.375 \geq 0.25 \Rightarrow 1$ in the 2^{-2} column

$0.375 - 0.25 = 0.125 \Rightarrow 1$ in the 2^{-3} column.

$0.875_{10} = 0000.1110_2$

ii) $0.625 \geq 0.5 \Rightarrow 1$ in the 2^{-1} column

$0.125 < 0.25 \Rightarrow 0$ in the 2^{-2} column

$0.125 \Rightarrow 1$ in the 2^{-3} column

$0.625_{10} = 0000.0110_2$

iii) Use two's complement conversion to 0.375

```
  0000.1010
  1111.1001
+         1
-----------
  1111.0110
```

iv) Add

```
  0000.1110
+ 1111.0110
-----------
 1,0000.0100
```

```
   0.875
 - 0.625
 -------
   0.250
```

∴ 0000.0100_2 (0.250_{10})

3) EX#3 (Example 5.7 Variation)

Alyssa P. Hacker is building a finite state machine that must run at 400MHz. She uses a Cyclone IV GX FPGA with the following specifications: t_LE = 395 ps per LE, t_setup = 85ps, and t_pcq = 150ps for all flip-flops. The wiring delay between LEs is 236ps. Assume the hold time for the flip-flop is 0. What is the maximum number of Les her design can use?

i) Use $t_{pd} \leq T_c - (t_{pcq} + t_{setup})$.

$t_{pd} \leq 2.5ns - (0.150ns + 0.085ns)$

$\Rightarrow t_{pd} \leq 2.265ns$.

ii) Use $N t_{LE+wrie} \leq 2.265ns$.

$t_{LE+wire} = 395ps + 236ps = 631ps$

$= 0.631ns$

$N \cdot 0.631ns \leq 2.265ns$

$\Rightarrow N \leq 3.590$

$\therefore$ The maximum number of LEs is 3.

4. Glossary

1) Trade-off

- Noun:

  The exchange of one thing for another of more or less equal value, especially to effect a compromise.

2) Modularity

- Noun:

  The use of individually distinct functional units, as in assembling an electronic or mechanical system.

3) Regularity (Regular)

- Adjective:

  I. Usual; normal; customary
  II. Evenly or uniformly arranged; symmetrical
  III. Characterized by fixed principle, uniform procedure, etc.
  IV. Recurring at fixed times; periodic

- Noun:

  I. A long-standing or habitual customer or client
  II. Ecclesiastical. A member of a duly constituted religious order under a rule.

4) Prepend

- Verb:

  I. Add (something) to the beginning of something else.
  II. [Computing] Attach (a piece of data) to the beginning of another.

5) Mantissa

- Noun

  I. [Mathematics] the part of a logarithm that follows the decimal point.
  II. [Computing] the part of a floating-point number that represents the significant digits of that number, and that is multiplied by the base raised to the exponent to give the actual value of the number.

5. Reflection

   The content of this chapter was familiar to me when I read this chapter. The reason is that in the previous sections of the lab (ECE272), the materials in this chapter which are adders and counter were used to make counters and clock. Because of these experiences, the contents which were not handled in the lab section were intriguing so that it was fun to obtain knowledges about digital building blocks. I think that the content of this chapter will be fundamentally applicated in the future chapters. Therefore, I am going to read and study more about the materials in this chapter, chapter 5.

6. Questions
   1) By using memory array and logic array, can we able to create CPU?

   2) Is power consumption control of memory devices handled by memory array?

   3) If there is read-only memory, then is there a write-only memory? Because in our computer, there is a function to write something in the stored file. If not, then how do make the writing function with those memory and logic array?