**NASIR-AMEEN NASIR**

**17/ENG02/045**

**COMPUTER ENGINEERING**

**Assignment Title:** Summary of Note Given
**Course Title:** Digital Computer Systems

**Course Code:** COE 312

**Summary’s Contents**

* **FLIP-FLOP APPLICATIONS**
* **FLIP-FLOP SYNCHRONIZATION**
* **DETECTING AN INPUT SEQUENCE**
* **DATA STORAGE AND TRANSFER**
* **FREQUENCY DIVISION AND COUNTING**
* **ASYNCHRONOUS (RIPPLE) COUNTERS**
* **PROPAGATION DELAY IN RIPPLE COUNTERS**
* **SYNCHRONOUS (PARALLEL) COUNTERS**
* **COUNTERS WITH MOD NUMBERS < 2**
* **SYNCHRONOUS DOWN AND UP/DOWN COUNTERS**
* **PRESETTABLE COUNTERS**
* **DECODING A COUNTER**
* **ANALYZING SYNCHRONOUS COUNTERS**
* **SYNCHRONOUS COUNTER DESIGN**
* **INTEGRATED CIRCUIT REGISTERS**
* **SHIFT-REGISTER COUNTERS**

**FLIP-FLOP APPLICATIONS**

Clocked flip-flops (Edge-Triggered Flip Flops) are versatile devices that can be used in a wide variety of applications including counting, storing of binary data, transferring binary data from one location to another, and many more. Almost all of these applications utilize the Flip-Flop's clocked operation. Many of them fall into the category of sequential circuits. A sequential circuit is one in which the outputs follow a predetermined sequence of states, with a new state occurring each time a clock pulse occurs.

**FLIP-FLOP SYNCHRONIZATION**

Most digital systems are principally synchronous in their operation because most of the signals will change states in synchronism with the clock transitions. However, in most cases there will be an external signal that is not synchronized to the clock (asynchronous). Asynchronous signals often occur as a result of a human operator's actuating an input switch at some random time relative to the clock signal. This randomness can produce unpredictable and undesirable results.

**DETECTING AN INPUT SEQUENCE**

In many situations, an output is to be activated only when the inputs are activated in a certain sequence. This cannot be accomplished using pure combinational logic but requires the storage characteristic of Flip Flops

**DATA STORAGE AND TRANSFER**

The most common use of flip-flops is for the storage of data or information. The data may represent any of a wide variety of types of data that have been encoded in binary. These data are generally stored in groups of Flip-flops called registers. The operation most often performed on data that are stored in a register is the data transfer operation; this involves the transfer of data from one register to another.

There are two major forms of data transfer which are: Parallel Data Transfer and Serial Data Transfer.

**Parallel Data Transfer:**  In parallel transfer, the output of each Flip Flop in register X is connected to a corresponding Flip Flop input in register Y. It is important to understand that parallel transfer does not change the contents of the register that is the source of data.

**Serial Data Transfer (Shift Registers):** A shift register is a group of Flip Flops arranged so that the binary numbers stored in the Flip Flops are shifted from one to the next for every clock pulse. Devices such as an electronic calculator, where the digits shown on the display shift over each time you key in a new digit have the same action taking place in a shift register.

**Differences Between Parallel and Serial Transfer**

In parallel transfer, all of the information is transferred simultaneously upon the occurrence of a single transfer command pulse, no matter how many bits are being transferred. In serial transfer, the complete transfer of N bits of information requires N clock pulses (i.e three bits require three pulses, four bits requires four pulses, etc.). In other words, parallel transfer then, is obviously much faster than serial transfer using shift registers. The choice of either parallel or serial transmission depends on the particular system application and specifications. Usually, a combination of the two types is used to take advantage of the speed of parallel transfer and the economy and simplicity of serial transfer. More will be said later about information transfer.

**FREQUENCY DIVISION AND COUNTING**

For frequency division, toggle mode flip-flops are used in a chain as a divide by two counter. The final output clock signal will have a frequency value equal to the input clock frequency divided by the MOD number of the counter. Such circuits are known as “divide-by-n” counters. Counters can be formed by connecting individual flip-flops together and are classified according to the way they are clocked.

The basic idea of making a frequency counter is to count the number of cycles of the input in one second. We can keep one counter that keeps counting the rising/falling edges of the input signal. Then we can read the value of this counter once in a second and then clear it. The value we read from the counter is nothing but the frequency of the input signal.

**MOD Number**

An example is a MOD-8 counter, where the MOD number indicates the number of states in the counting sequence. If a fourth Flip Flop were added, the sequence of states would count in binary from 0000 to 1111, a total of 16 states. This would be called a MOD-16 counter. In general, if N flip-flops are connected, the counter will have 2N different states, and so it is a MOD-2N counter. It would be capable of counting up to 2N - 1 before returning to its 0 state. The MOD number of a counter also indicates the frequency division obtained from the last FF.

**ASYNCHRONOUS (RIPPLE) COUNTERS**

It often referred to as a ripple counter because of the way the FFs respond one after another in a kind of rippling effect. In this counter, each Flip Flop output drives the CLK input of the next FF. This is called an asynchronous counter because the Flip Flop’s do not change states in exact synchronism with the applied clock pulses; only flip-flop A responds to the clock pulses. Flip Flop B must wait for Flip Flop A to change states before it can toggle; Flip Flop C must wait for FF B, and so on. Therefore, there is a delay between the responses of successive FFs. This delay is typically 5-20 ns per Flip Flop.

**PROPAGATION DELAY IN RIPPLE COUNTERS**

Ripple counters are the simplest type of binary counters because they require the fewest components to produce a given counting operation. They do, how­ ever have one major drawback, which is caused by their basic principle of operation: each Flip Flop is triggered by the transition at the output of the preceding Flip Flop. Because of the inherent propagation delay time (t,) of each Flip Flop, this means that the second Flip Flop will not respond until a time tpd after the first Flip Flop receives an active clock transition; the third Flip Flop will not respond until a time equal to 2 X tpd after that clock transition; and so on. In other words, the propagation delays of the Flip Flops accumulate so that the Nth Flip Flop cannot change states until a time equal to N X tpd after the clock transition occurs.

**SYNCHRONOUS (PARALLEL) COUNTERS**

The problems encountered with ripple counters are caused by the accumulated Flip Flop propagation delays; stated another way, the Flip Flops do not all change states simultaneously in synchronism with the input pulses. These limitations can be overcome with the use of synchronous or parallel counters in which all of the FFs are triggered simultaneously (in parallel) by the clock input pulses. Because the input pulses are applied to all the Flip Flops, some means must be used to control when an Flip Flop is to toggle and when it is to remain unaffected by a clock pulse, this is accomplished by using the J and K inputs

**Advantage of Synchronous Counters over Asynchronous**

In a parallel counter, all of the Flip Flops will change states simultaneously; they are all synchronized to the NGTs of the input clock pulses. Thus, unlike the asynchronous counters, the propagation delays of the FFs do not add together to produce the overall delay. Instead, the total response time of a synchronous counter is the time it takes one FF to toggle plus the time for the new logic levels to propagate through a single AND gate to reach the J, K inputs. That is, for a synchronous counter, total delay = FF tpd, + AND gate tpd. This total delay is the same no matter how many Flip Flops are in the counter, and it will generally be much lower than with an asynchronous counter with the same number of Flip Flops. Thus, a synchronous counter can operate at a much higher input frequency. Of course, the circuitry of the synchronous counter is more complex than that of the asynchronous counter.

There are many synchronous IC counters in both the TTL and the CMOS logic families. Some of the most commonly used devices are:

 74ALS160/162, 74HC160/162: synchronous decade counters

74ALS161/163, 74HC161/163: synchronous MOD-16 counters

**COUNTERS WITH MOD NUMBERS < 2**

 The basic synchronous counter is limited to MOD numbers that are equal to 2N, where N is the number of FFs. This value is actually the maximum MOD number that can be obtained using N flip-flops. The basic counter can be modified to produce MOD numbers less than 2N by allowing the counter to skip states that are normally part of the counting sequence. However, the presence of the NAND gate will alter this sequence as follows:

1. The NAND output is connected to the asynchronous CLEAR inputs of each FF. As long as the NAND output is HIGH, it will have no effect on the counter. When it goes LOW, however, it will clear all of the FFs so that the counter immediately goes to the 000 state.

2. The inputs to the NAND gate are the outputs of the Band C flip-flops, and so the NAND output will go LOW whenever B = C = 1. This condition will occur when the counter goes from the 101 state to the 110 state on the NGT of input pulse 6. The LOW at the NAND output will immediately (generally within a few nanoseconds) clear the counter to the 000 state. Once the FFs have been cleared, the NAND output goes back HIGH because the B = C = 1 condition no longer exists.

3. The counting sequence is, therefore,

CBA

000

001

010

011

100

101

110 (temporary state needed to clear counter)

Although the counter does go to the 110 state, it remains there for only a few nanoseconds before it recycles to 000. Thus, we can essentially say that this counter counts from 000 (zero) to 101 (five) and then recycles to 000.It essentially skips 110 and 111 so that it goes through only six different

states; thus, it is a MOD-6 counter. The waveform at the output contains a spike or glitch caused

by the momentary occurrence of the 110 state before clearing. This glitch is very narrow and so would not produce any visible indication on indicator LEDs or numerical displays. It could, however, cause a problem if the output were being used to drive other circuitry outside the counter. It should also be noted that the C output has a frequency equal to one-sixth of the input frequency; in other words, this MOD-6 counter has divided the input frequency by six. The waveform at C is not a symmetrical square wave (50 percent duty cycle) because it is HIGH for only two clock cycles, whereas it is LOW for four cycles.

**SYNCHRONOUS DOWN AND UP/DOWN COUNTERS**

A synchronous down counter is constructed using the inverted Flip Flop outputs to control the higher-order J, K inputs. Comparing the synchronous, MOD-16 down counter with the up counter shows that we need only to substitute the corresponding inverted Flip Flop output in place of the A, B, and C outputs. For a down count sequence, the LSB FF (A) still needs to toggle with each NGT of the clock input signal Flip-flop B must change states on the next NGT of the clock when A = 0(A = 1). Flip-flop C changes states when A =B=0( 44..B=1), and flip-flop D changes states when A = B =C = 0 (4.8. C= 1).This circuit configuration will produce the count sequence: 15,14, 13,12,...,3,2, 1, 0, 15, 14, and so on, as shown in the timing diagram.

**PRESETTABLE COUNTERS**

Many synchronous (parallel) counters that are available as ICs are designed to be presettable; in other words, they can be preset to any desired starting count either asynchronously (independent of the clock signal) or synchronously (on the active transition of the clock signal). This presetting operation is also referred to as parallel loading the counter.

**Asynchronous Presetting**

This type of presetting is used by several IC counters, such as the TTL 74ALS190, 74ALS191, 74ALS192.

**Synchronous Presetting**

This type of presetting is used by several IC counters, such as the TTL 74ALS160, 74ALS161, 74ALS162.

**DECODING A COUNTER**

Digital counters are often used in applications where the count represented by the states of the Flip Flops must somehow be determined or displayed. One of the simplest means for displaying the contents of a counter involves just connecting the output of each Flip Flop to a small indicator LED In this way the states of the Flip Flops are visibly represented by the LEDs ( on = 1,off= 0), and the count can be mentally determined by decoding the binary states of the LEDs. The indicator LED method becomes inconvenient as the size (number of bits) of the counter increases because it is much harder to decode the dis played results mentally. For this reason, it is preferable to develop a means for electronically decoding the contents of a counter and displaying the results in a form that is immediately recognizable and requires no mental operations. Another more important reason for electronic decoding of a counter occurs because of the many applications in which counters are used to control the timing or sequencing of operations automatically without human intervention.

**Active-HIGH Decoding:**

A MOD-X counter has X different states; each state is a particular pattern of 0s and 1s stored in the counter FFs. A decoding network is a logic circuit that generates X different outputs, each of which detects (decodes) the presence of one particular state of the counter. The decoder outputs can be designed to pro­ duce either a HIGH or a LOW level when the detection occurs. An active-HIGH decoder produces HIGH outputs to indicate detection. The decoder consists of eight three-input AND gates. Each AND gate produces a HIGH out­ put for one particular state of the counter.

**Active-LOW Decoding:** If NAND gates are used in place of AND gates, the decoder outputs produce a normally HIGH signal, which goes LOW only when the number being decoded occurs. Both types of decoders are used, depending on the type of circuits being driven by the decoder outputs.

**BCD Counter Decoding:** A BCD counter has 10 states that can be decoded using the techniques described previously. BCD decoders provide 10 outputs corresponding to the decimal digits 0-9 and represented by the states of the counter Flip Flops. These 10 outputs can be used to control IO individual indicator LEDs for a visual display. More often, instead of using 10 separate LEDs, a single display device is used to display the decimal numbers 0-9. One class of decimal displays contains seven small segments made of a material (usually LEDs or LCDs) that either emits light or reflects ambient light. The BCD decoder outputs control which segments are illuminated in order to produce a pattern representing one of the decimals

digits.

**ANALYZING SYNCHRONOUS COUNTERS**

Synchronous counter circuits can be custom-designed to generate any desired count sequence. We can use just the synchronous inputs that are applied to the individual flip-flops to produce the counter's sequence. By not using asynchronous FF controls, such as the clears, to change the counter's sequence, we will never have to deal with transient states and possible glitches in output waveforms. In the analysis, A PRESENT state/NEXT state table is a very useful tool in this analysis process. The first step is to write the logic expression for each Flip Flop control input. Next, we assume a PRESENT state for the counter and apply that combination of bits to the control logic expressions. The outputs from the control expressions will allow us to predict the commands to each Flip Flop and the resulting NEXT state for the counter after clocking. Repeat the analysis process until the entire count sequence is determined.

It is important to note that the gating resources for most PLDs actually consist of sets of AND-OR circuit arrangements and the SOP logic expression more accurately describes the internal circuit implementation. However, we can see that the expressions have been greatly simplified by using the XOR function. This leads us to predict correctly that to create a MOD-16 binary counter with D flip-flops, we would need a fourth Flip Flop with:

D0 =D Ꚛ (ABC)

**SYNCHRONOUS COUNTER DESIGN**

Many different counter arrangements are available as ICs - asynchronous, synchronous, and combined asynchronous/synchronous. Most of these count in a normal binary or BCD count sequence, although their counting sequences can be somewhat altered . There are situations, however, where a custom counter is required that follows a sequence that is not a regular binary count pattern. Several methods exist for designing counters that follow arbitrary sequences.

**Design Procedure for J-K Flip Flops**

Step 1. Determine the desired number of bits (FFs) and the desired counting sequence.

Step 2. Draw the state transition diagram showing all possible states, include ing those that are not part of the desired counting sequence.

Step 3. Use the state transition diagram to set up a table that lists all PRESENT states and their NEXT states.

Step 4. Add a column to this table for each] and K input. For each PRESENT state, indicate the levels required at each J and K input in order to produce the transition to the NEXT state.

Step 5. Design the logic circuits needed to generate the levels required at each J and K input.

Step 6. Implement the final expressions.

**INTEGRATED CIRCUIT REGISTERS**

The various types of registers can be classified according to the manner in which data can be entered into the register for storage and the manner in which data are outputted from the register. Each of these types and several variations are available in IC form so that a logic designer can usually find exactly what is required for a given application.

 The various classifications are listed below:

1. **Parallel in/parallel out (PIPO):** These isa group of flip-flops that can store multiple bits simultaneously and in which all bits of the stored binary value are directly available.

2. **Serial in/serial out (SISO):**  This register will have data loaded into it one bit at a time. The data will move one bit at a time with each clock pulse through the set of flip-flops toward the other end of the register. With continued clocking, the data will then exit the register one bit at a time in the same order as it was originally loaded.

3. **Parallel in/serial out (PISO):** This has serial data entry via Ds and asynchronous parallel data entry via P0 through P7. The register contains eight FFs (Q0 through Q7), internally connected as a shift register, but the only accessible FF outputs are Q7 and Q7.

4. **Serial in/parallel out (SIPO):** It is a shift register with each FF output externally accessible. Instead of a single serial input, an AND gate combines inputs A and B to produce the serial input to flip-flop Q0.

**SHIFT-REGISTER COUNTERS**

Shift register counters use feedback, which means that the output of the last Flip Flop in the register is connected back to the first Flip Flop in some way.

**Ring Counter**

The simplest shift-register counter is essentially a circulating shift register connected so that the last FF shifts its value into the first FF. The Flip Flops are connected so that information shifts from left to right and back around from Q0 to Q3. In most instances, only a single 1 is in the register, and it is made to circulate around the register as long as clock pulses are applied. This is why it is called a ring counter.