CHIDUBEM ONOCHIE-ONYETENU BRANDON

17/ENG02/072

COMPUTER ENGINEERING

300L

COE 312 ASSIGNMENT

SUMMARY OF FLIP-FLOP APPLICATIONS AND COUNTERS AND REGISTERS

FLIP FLOP APPLICATIONS

Edge-triggered (clocked) 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 FF's clocked operation.

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. In many cases, however, there will be an external signal that is not synchronized to the clock; in other words, it is asynchronous. Asynchronous signals often occur as a result of human operators actuating an input switch at some random time relative to the clock signal. This randomness can produce unpredictable and undesirable results.

DATA STORAGE AND TRANSFER

The most common use of flip-flops is for the storage of data or information. The data may represent numerical values (e.g. binary numbers, BCD-coded decimal numbers) or any of a wide variety of types of data that have been encoded in binary. These data are generally stored in groups of FF's called registers. The operation most often performed on data that are stored in FF or a register is the data transfer operation. This involves the transfer of data from one FF or register to another.

Parallel Data Transfer- is data transfer from one register to another using D-type FF'S. The 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 FF's arranged so that the binary numbers stored in the FF's are shifted from one FF to the next for every clock pulse. On each NGT of the shift pulses, each FF output takes on the level that was present at the output of the FF on its left just prior to the NGT. In serial data transfer the contents of a register is transferred to another register one bit at a time.

Hold Time Requirement- In this shift-register arrangement, it is necessary that the FF's have a very smack hold time requirement because there are times when the J, K inputs are changing at about the same time as the CLK transition.

Serial Transfer between Registers- on the NGT of each pulse, each FF takes on the value that was stored in the FF on its left prior to the occurrence of the pulse. The complete transfer of the three bits of data requires three shift pulses.

Shift-left Operation- there is no general advantage of shifting in one direction over another; the direction chosen by a logic designee will often be dictated by the nature of the application.

Parallel Versus 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 been transferred. In serial transfer the complete transfer of N bits of information requires N clock pulses (three bits require three pulses, four bits require four pulses, etc.). Parallel transfer is obviously much faster than serial transfer using shift registers. In parallel transfer, the output of each FF in register X is connected to register Y, in serial transfer, only the last FF in register X is connected to register Y. In general, parallel transfers require more interconnections between the sending register(X) and the receiving register(Y) than those serial transfer.

MOD NUMBER

The MOD number indicates the number of states in the counting sequence. The MOD number of a counter also indicates the frequency division obtained from the last FF. it is also equal to the number of states that the counter goes through in each complete cycle before it recycles back to its starting state. It can be increased simply by adding FF's to the counter.

ASYNCHRONOUS RIPPLE COUNTERS

In asynchronous ripple counters the FF's do not change states in exact synchronism with the applied clock pulses. Thus there is a delay between the responses of successive FF's. This type of counter is also referred to as a ripple counter because of the way the FF's respond one after another in a kind of rippling effect.

Signal Flow

It is conventional in circuit schematics to draw circuits (wherever possible) so that the signal flow is from left to right, with inputs on the left and outputs on the right. The CLK inputs of each FF are on the right, the outputs are on the left, we use this operation because it makes the counter operation easier to understand and follow (because the order of the FF's is the same as the order of the bits in the binary number that the counter represents).

Frequency Division

In the basic counter, each FF provides an output waveform that is exactly half the frequency of the wave form at its CLK input. In any counter, the signal at the output of the last FF (i.e., the MSB) will have a frequency equal to the input clock frequency divided by the MOD number of the counter. The first step in building a digital clock is to take the 60Hz signal and feed it into a Schmitt-trigger pulse-shaping circuit to produce a square wave.

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 however, have one major drawback, which is caused by their basic principle of operation: each FF is triggered by the transition at the output of the preceding FF.

Synchronous (Parallel) Counters

The problems encountered with ripple counters are caused by the accumulated FF propagation delays; stated another way, the FF's do not change states simultaneously in synchronism with the input pulses. These limitations can be overcome by with the use of synchronous or parallel counters in which all of the FF's are triggered simultaneously (in parallel) by the clock input pulses.

Because the input pulses are applied to all the FF's, some means must be used to control when an FF is to toggle and when it is to remain unaffected by a clock pulse. This is accomplished by using the J and K inputs.

Differences between the circuit arrangements of synchronous counter with asynchronous counter

1. The CLK inputs of all the FF's are connected together so that the input clock signal is applied to each FF simultaneously.

2. Only flip-flop A, the LSB< has its J and K inputs permanently at the HIGH level. The J, K inputs of the other FF's are driven by some combination of FF outputs.

3. The synchronous counter requires more circuitry than does the asynchronous counter.

<u>Circuit Operation</u>

For this circuit to count properly, on a given NGT of the clock, only those FF's that are supposed to toggle on that NGT should have J=K=1 when that NGT occurs. The counting sequence shows that the A flip-flop must change states at each NGT. For this reason, it's J and K inputs are permanently HIGH so that it will toggle on each NGT of the clock input.

The basic principle for constructing a synchronous counter can therefore be stated as follows:

Each FF should have its J and K inputs connected so that they are HIGH only when the outputs of all lower-order FF's are in the HIGH state.

<u>Advantage of Synchronous Counters over Asynchronous</u>

In a parallel counter, all of the FF's will change states simultaneously; that is, they are all synchronized to the NGT's of the input clock pulses. Thus, unlike the asynchronous counters, the propagation delays of the FF's 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.

This total delay is the same no matter how many FF's are in the counter, and it will generally be much lower than with an asynchronous counter with the same number of FF's. 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.

Actual ICs

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

Display Counter States

Sometimes during normal operation, and very often during testing, it is necessary to have a visible display of how a counter is changing states in response to the input pulses.

Decade Counters/BCD Counters

The MOD-10 counter is also referred to as decade counter. In fact, a decade counter is any counter that has 10 distinct states, no matter what the sequence. A decade counter which counts in sequence from 0000(zero) through 1001(decimal 9), is also commonly called a BCD counter because it uses only the 10 BCD code groups 0000, 0001,…, 1000, and 1001. To reiterate, any MOD-10 counter is a decade counter, and any decade counter that counts bin binary from 0000 to 1001 is a BCD counter.

Decade counters, especially the BCD type, find widespread use in applications where pulses or events are to be counted and the results displayed on some type of decimal numerical readout. A decade counter is also often used for dividing a pulse frequency exactly by 10. The input pulses are applied to the parallel clock inputs, and the output pulses are taken from the output of flip-flop D, which has one-tenth the frequency of the input signal.

Synchronous Down and Up/Down Counters

When you use the output of lower-order FF's to control the toggling of each FF creates a synchronous up counter. A synchronous down computer is constructed in a similar manner except that we use the inverted FF outputs to control the higher order J, K inputs.

Presettable Counters

Many synchronous(parallel) counters that are available as IC's 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.

Synchronous Presetting

Many IC parallel counters use synchronous presetting whereby the counter is preset on the active transition of the same clock signal that is used for counting. The logic level on the parallel load control input determines if the counter is preset with the applied input data at the next active clock transition

Examples of IC counters that use synchronous presetting include the TTL7ALS160, 74ALS161, 74ALS162, AND 74ALS163 and their CMOS equivalents, 74HC160, 74HC161, 74HC162, and 74HC163.

IC Synchronous Counters

The 74ALS160-163/74HC160-163 Series

The 74ALS160 and 74ALS161 each has an asynchronous clear input. The clear input has priority over all other functions for this series of IC counters. Another priority function available in this series of IC counters is the parallel loading of data into the counters flip-flops. This series of IC counter chips has one or more output pin, RCO. The function of this active-HGH output is to detect (decode) the last or terminal state of the counter. The terminal state for a decade counter is 1001(9), while the terminal state for a MOD-16 counter is 111(15). ENT, the primary enable input, also controls the operation of RCO. ENT must be HIGH for the counter to indicate with the RCO output that it has reached its terminal state.

The 74ALS190-191/74HC190-191 Series

The only difference between the two part numbers is the counters modulus. The 74ALS190 is a MOD-10 counter and the 7ALS191 is a MOD-16 binary counter. Both chips are up/down counters and have an asynchronous, active-LOW load input.

Multistage Arrangement

Many standard IC counters have been designed to make it easy to connect multiple chips together to create circuits with a higher counting range. All of the counter chips presented in this section can be simply connected in a multistage or cascading arrangement.

Decoding a Counter

Digital counters are often used in applications where the count represented by the states of the FF's 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 FF to a small indicator LED. IN THIS WAY THE STATES OF THE FF's 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. For instance, suppose that this method is used for a BCD counter and the states of the LEDs are off-on-on-off, respectively. This would represent 0110, which we would mentally decode as decimal 6. Other combinations of LED states would represent the other possible counts.

The indicator LED method becomes inconvenient as the size (number of bits) of the counter increases because it is much harder to decode the displayed 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.

An even 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. For example, a certain system operation might have to be initiated when a counter reaches the 101100 state. A logic circuit can be used to decode for or detect when this particular count is present and then initiate the operation. Many operations may have to be controlled in this manner in a digital system. Clearly, human intervention in this process would be undesirable except in extremely slow systems.

Active-High Decoding

A MOD-X counter has X different states; each state is a particular pattern of 0s and 1s stored in the stored in the counter FF's. 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 produce either a HIGH or a LOW level when the detection occurs. An active HIGH decoder produces HIGH outputs to indicate detection.

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

BCD decoders provide 10 outputs corresponding to the decimal digits 0 through 9 and represented by the states of the counter FFs. These 10 outputs can be used to control 10 individual indicator LEDs for a visual display. More often, instead of using 10 separate LED's, a single display device is used to display the decimal numbers 0 through 9. One class of decimal displays contains seven small segments made of material (usually LEDs or liquid-crystal displays) 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 decimal 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. The process of designing completely synchronous counters will be investigated in the next section. First, let's see how to analyze a counter design of this type predicting the FF control inputs for each state of the counter. A PRESENT state/NEXT state is a very useful toll in this process. The first step is to wire the logic expression for each FF control input. Next 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 FF and the resulting NEXT state for the counter after clocking. Repeat the analysis process until the entire count sequence is determined. A self-correcting counter is one in which normally unused states will all somehow return to the normal sequence, the counter is said to be not self-correcting. The gating resources for most PLDs actually consists 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 FF.

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 using the clearing or loading methods we demonstrated for the 74ALS160-163 and 74ALS190-191 series of ICs. There are situations, however, where a custom counter is required that follows a sequence that is not a regular binary count pattern, for example, 000, 010, 101, 001, 110, 000,…   The process of designing a synchronous counter thus becomes one of designing the logic circuits that decode the various states of the counter to supply the proper logic levels to each J and K input at the correct time. The inputs to these decoder circuits will come from the outputs of one or more of the FF's.

Stepper Motor Control

A stepper motor is a motor that rotates in steps, typically 15 degrees per step, rather than in a continuous motion. Magnetic coils or windings within the motor must be energized and deenergized in a specific sequence in order to produce this stepping action. Digital signals are normally used to control the current in each of the motors coils. Stepper motors are used extensively in situations where precise position control is needed, such as in positioning of read/write heads on magnetic disks, in controlling print heads in printers, and in robots.

Synchronous Counter Design with D FF

Historically, J-K flip-flops have been used to implement counters because the logic circuits needed for the J and K inputs are usually simpler than the logic circuits needed to control an equivalent synchronous counter using D flip-flops. When designing counters that will be implemented in PLD's, where abundant gates are generally available, it makes sense to use D flip-flops instead of J-Ks.

Designing counter circuits using D flip-flops is even easier than using J-K flip-flops.

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. The various classifications are listed below:

1. Parallel in/parallel out (PIPO)
2. Serial in/serial out (SISO)
3. Parallel in/serial out (PISO)
4. Serial in/parallel out (SIPO)

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.

Parallel IN/Parallel Out- The 74ALS174/74HC174

A group of flip-flops that can store multiple bits simultaneously and in which all bits of the stored binary value are directly available is referred to as a parallel in/parallel out register.

The 74ALS174 is normally used for synchronous parallel data transfer whereby the logic levels present at the D inputs are transferred to the corresponding Q outputs when a PGT occurs at the clock CP. This IC, however, can be wired for serial data transfer.

Serial In/Serial Out- The 74ALS166/74HC166

A serial in/serial out shift 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. The 74HC166 (and also the 74ALS166) can be used as a serial in/serial out register. The logic diagram and schematic symbol for the 74HC166 is an eight-bit shift register of which only FF OH is accessible. The serial data is input on SER and will be stored in FF QA. The serial output isobtained at the other end of the shift register. The synchronous serial shifting and parallel loading functions can be inhibited (disabled) by applying a HIGH to the CLK INH control input. The register also has an active-LOW, asynchronous clear input (CLR).

A shift register is often used as a way to delay a digital signal by an integral number of clock cycles. The digital signal is applied to the shift register's serial input and is shifted through the shift register by successive clock pulses until it reaches the end of the shift register, where it appears as the output signal. This method for delaying the effect of a digital signal is common in the digital communications field. For instance, the digital signal might be the digitized version of an audio signal that is to be delayed before it is transmitted.

Parallel In/Serial Out- The 74ALS165/74HC165

This IC is an eight-bit parallel in/serial out register. CP is the clock input used for the shifting operation. The clock inhibit input, CP INH is used to inhibit the effect of the CP input. The shift/load input, SH/LD, controls which operation is taking place- shifting or parallel loading.

The serial shifting function will always be synchronous, since the clock is required to ensure that the input data moves only one bit at a time with each appropriate clocking edge.

Serial In/Parallel Out- The 74ALS164/74HC164

The logic diagram for the 74ALS164/74HC164 is an eight-bit serial in/parallel out 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 Q.

The shift operation occurs on the PGTs of the clock input CP. The MR input provides asynchronous resetting of all FFs on a low level.

The following is a list of some other register IC's that are variations on those already presented:

74194/ALS194/HC194: This is a four-bit bidirectional universal shift register IC that can perform shift-left, shift-right, parallel in, parallel out operations. These operations are selected by a two-bit mode select code applied as inputs to the device.

74373/ALS373/HC373/HCT373: This is an eight-bit (octal) parallel in/parallel out register containing eight D latches with tristate outputs. A tristate output is a special type of logic circuit output that allows device outputs to be tied together safely.

74374/ALS374/HC374/HCT374: This is an eight-bit (octal) parallel in/parallel out register containing eight edge- triggered D flip-flops with tristate outputs.

The IC registers that have been represented here are representative of the various types that are commercially available. Although there are many variations on these basic registers, most of them should now be relatively easy to understand from the manufacturer's data sheets.

Shift Register Counters

Shift register counters use feedback, which means that the output of the last FF in the register is connected back to the first FF 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. This arrangement is shown using D-type FFs (J-K flip-flops can also be used). The FFs are connected so that information shifts from left to right and back around. In most cases, only a single 1 is in the register, and it is made to circulate around the register as long as clock pulses are applied. For this reason, it is called a ring counter.

The waveforms, sequence table, and state diagram show the various states of the FFs as pulses are applied, assuming a starting state of Q3 = 1 and Q2 = Q1 =Q0. After the first pulse, the 1 has shifted from Q3 to Q2 so that the counter is in the 0100 state. The second pulse produces the 0100 state, and the third pulse produces the 0001 state. On the fourth clock pulse, the 1 from Q0 is transferred to Q3, resulting in the 1000 state, which is, of course, the initial state. Subsequent pulses cause the sequence to repeat.

The counter functions as MOD-4 counter because it has four distinct states before the sequence repeats. Although this circuit does not progress through the normal binary counting sequence, it is still a counter because each count corresponds to a unique set of FF states. Each FF output waveform has a frequency equal to one-fourth of the clock frequency because this is a MOD-4 ring counter. Ring counters can be constructed for any desired MOD number.