Answers to Exercises

1. What happens if an infinite loop includes a watchdog timer reset? Name one thing that can be done to guard against this.

Ans.

If possible reset the timer only from the mainline of the program (as we show in the code fragment). Long-running functions may need to be broken into smaller modules.

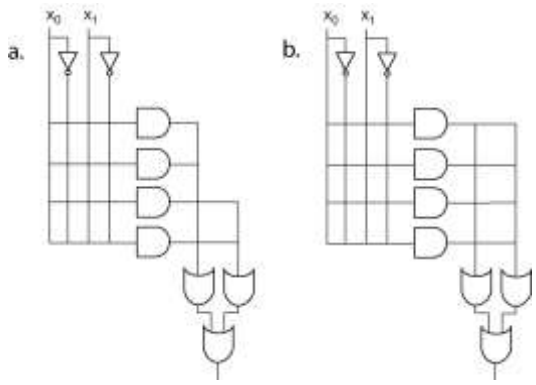
♦2. In the sidebar concerning watchdog timer engineering decisions, we stated that rebooting an embedded system typically takes less time than rebooting a personal computer. Why do you think that this is so?

Ans.

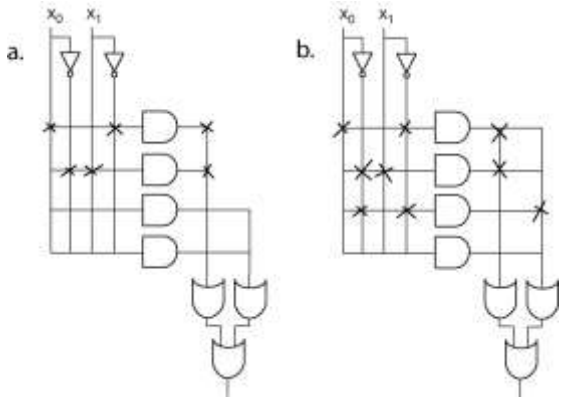
Small embedded systems have no need to perform the complicated power-on self-test (POST) sequence that is carried out by personal systems. For one thing, embedded systems memories are much smaller, thus take less time for power-on checks. Secondly, peripheral devices on most embedded systems-- if they exist at all-- are fewer and simpler than those connected to most personal computers. This means less hardware to check, and fewer drivers to load.

---

3. a) Show how a two-input XOR gate can be implemented in the PAL shown below.
   b) Show how a two-input NAND gate can be implemented in the PLA shown below.

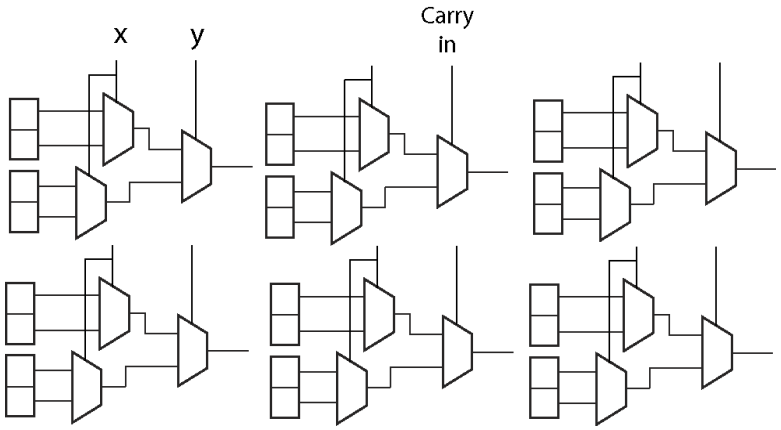


Ans.

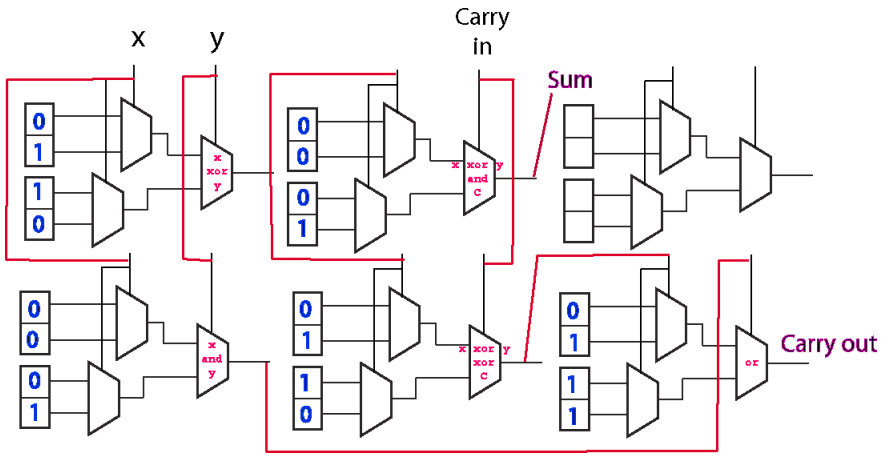


---

4. Use the FPGA illustrated below to implement a full adder. Label the outputs clearly. Draw lines between the cells to indicate a connection between the logic functions.



Ans.



---

5. Provide a detailed logic diagram for a multiplexer that could be used in an FPGA.

Ans.

Refer to the multiplexer circuit given in Chapter 3.

---

6. Present arguments for and against the use of dynamic memory in embedded systems. Should it be banned for use under all circumstances? Why or why not?

Ans.

The main risk with dynamic memory in embedded systems is that the heap can overflow. This is particularly risky for systems that must run for months or years without being restarted. Even the smallest memory leak will add up over prolonged periods. Heap overflow can be disastrous. The aim of this question, however, is to get the student thinking about the advantages of dynamic memory and finds some applications where its benefits outweigh its risks. For example, a computer game could use the memory in creative ways without putting anyone's life at risk.

---

7. We say that in embedded operating systems, if the highest-priority user thread is executing when a high-priority interrupt occurs, most operating systems will continue to process the user thread and keep the interrupt in queue until processing is completed. Under what circumstances would this be and would this not be a problem? Give an example of each.

Ans.

If the application is highly time-critical, such as an automobile airbag inflation control, latency of a few microseconds can be catastrophic. If the application is not so time-critical, such as in a cell phone or game machine, a delay of a few milliseconds usually will not be noticed.

♦8. Explain interrupt latency. How is it related to context switch time?

Ans.

Interrupt latency is the elapsed (wall clock) time between the occurrence of an interrupt and the execution of the first instruction of the interrupt service routine (ISR). In order to execute the first instruction of the ISR, the thread currently executing in the CPU must be suspended: A context switch takes place. The interrupt latency must therefore be greater than the context switch time.

---

9. In an ideal embedded operating system, would all non-kernel threads always execute at lower priority than interrupts?

Ans.

The ideal approach would be to be able to assign interrupt and thread priorities so that a low-priority interrupt would not preempt a thread spawned by a higher-priority interrupt. However, if a low priority thread is holding a resource that a higher priority thread needs, the lower priority thread must be aborted or allowed to finish ahead of the higher priority thread.

---

10. Explain the challenges of embedded software development. How do designers answer these challenges?

Ans.

Challenges include: resource constraints, the need to deal with signal timing, often the lack of tool support and debuggers. Development methods should be more rigorous than in general systems because embedded systems are less tolerant of bugs.
