Chapter 4
Digital Electronics

Digital electronics is the basic technology for today's computers and for most of today's communications equipment. Mastering digital electronics allows us to move some tasks an embedded system has to perform into dedicated digital hardware. Such hardware implementation usually exhibits much higher performance than a software implementation of the same task executing on a processor. The temporal behavior of bistable storage elements forces us to embrace a synchronous design discipline. Following this discipline produces circuits with predictable timing properties. Tasks demanding highly predictable timing, therefore, benefit most from being implemented directly in digital hardware.

4.1 Inputs

A digital circuit consists of components and of wires connecting the components' inputs and outputs. It drives wires carrying a logic One to a voltage close to the voltage of its positive supply rail and wires carrying a logic Zero to a voltage close to the voltage of its negative supply rail. Between the voltage band representing a logic One and the voltage band representing a logic Zero sits the forbidden zone. The larger the forbidden zone the more immune the circuit is to noise.

A logic input expects the connecting wire to either sit at a voltage level representing a logic One or at a voltage level representing a logic Zero. When being driven into the forbidden zone for a time longer than is required for switching between the logic states it penalizes such unruly behavior with elevated power consumption. In extreme cases oscillation and destruction are possible. The time it takes a wire to swing from logic Zero to logic One is the rise time t_r; the time it takes to swing from logic One to logic Zero is the fall time t_f. A logic input requires rise and fall times in the range of 30 ns. An analog signal, which may sit in the forbidden zone for an indefinite amount of time, is not suitable for driving a logic input.

A Schmitt trigger input can be driven with an analog signal. When the signal at the input transitions slowly from 0 V to the supply voltage the input initially interprets the signal as Zero. As soon as the signal reaches the threshold V_{T+} the input interprets the signal as One. When the signal ramps down from the supply voltage toward 0 V the input initially interprets the signal as One. As soon as the signal reaches the threshold V_{T-} the input changes its interpretation of the signal to Zero. The thresholds obey $V_{T+} > V_{T-}$, the Schmitt trigger has a hysteresis of $V_H = V_{T+} - V_{T-}$. A large hysteresis improves immunity to noise. A Schmitt trigger input circuit shows elevated power consumption while its input signal sits in the middle between the supply rails.

4.2 Outputs

A push-pull output stage establishes a low-impedance connection between the wire it connects to and the positive supply rail in order to drive the wire to a One. In order to drive the wire to a Zero it establishes a low-impedance connection between the wire and the negative supply rail. Connecting two push-pull outputs to the same wire is an error. The first output may try to drive the wire to a One while simultaneously the second tries to drive the same wire to a Zero. This results in a low-impedance connection between the supply rails and excessive power consumption.

A tri-state output buffer has an additional enable input besides its signal input. As long as the output is disabled the buffer effectively disconnects itself from the wire connected to it. When it is enabled it behaves like a push-pull output. As long as at most a single output is enabled several tri-state outputs can connect to a single wire. This allows building bidirectional buses. Typically a bus master, a microprocessor, for example, is responsible for providing the enable signals for the tri-state drivers in such a way that no two drivers are enabled simultaneously. In order to establish a valid logic level when the wire is not driven by any output a pull-up or pull-down device is added to the wire. A resistor between the wire and the positive rail acts as a pull-up, while a resistor between the wire and the negative rail acts as a pull-down. Pull-up and pull-down devices typically have resistances between $1\,k\Omega$ and $10\,k\Omega$. An alternative consuming less power is a bus hold device, also known as keeper latch, which pulls the wire with a high impedance to the logic level the wire was last driven to. Bus hold devices sometimes are built into logic inputs.

An open-drain output pulls the wire connected to it to Zero by establishing a low-impedance connection between the wire and the negative supply rail. It does not pull the wire to One; it disconnects itself from the wire instead. Therefore, a pull-up device has to provide the logic One. Several open-drain outputs may connect to the same wire. If at least one of them drives Zero onto the wire, overriding the pull-up, the wire will carry the Zero. Otherwise the pull-up will drive the wire to One.

4.2.1 The Nine Logic Values

Several outputs may drive a single wire. For representing the impedance of the different outputs we need more than two values for describing a wire's state. An idealized

Table 4.1 Function table of the wire operation

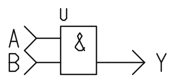
	U	X	0	1	Z	W	L	H	–
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
–	U	X	X	X	X	X	X	X	X

wire is a commutative and associative operation, Table 4.1, combining the logic values driven onto it. The idealization does not take into account the time for signal propagation.

When simulating a logic circuit the uninitialized value, `U`, serves as initial value for all wires in a circuit. Persisting uninitialized values in a simulation hint at errors in the simulated design. The Forcing Unknown, `X`, indicates error conditions such as bus conflicts. The Forcing Zero, `0`, represents a low-impedance connection to the negative supply rail, while the Forcing One, `1`, represents a low-impedance connection to the positive supply rail. A push-pull output imprints Forcing Zeros and Forcing Ones onto the wire it connects to. A disabled tri-state output drives the high-impedance value, `Z`, onto the wire it connects to. Termination structures, necessary when dealing with long wires, drive the Weak Unknown, `W`, onto the wires they connect to. A pull-down device drives a Weak Zero, `L` onto the wire it connects to, while a pull-up device drives a Weak One, `H`. The don't care value, `–`, is used in truth tables for indicating that the output value is irrelevant for a combination of input values or that in input is irrelevant for a certain combination of the other inputs. Don't Cares allow logic synthesis tools to generate simpler circuits. Inputs have very high impedance. Therefore, they do not distinguish between Forcing and Weak Zeros and between Forcing and Weak Ones.

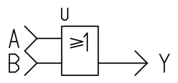
4.3 Combinatoric Logic

Combinatoric logic is built from logic gates and wires. A logic gate has a number of inputs and a single push-pull output. A circuit is combinatoric if we can arrange its diagram with the circuit's inputs to the left and the circuit's outputs to the right. Furthermore, the gate whose output drives a wire must appear left of any gate which connects to the wire with one of its input and left of any output connected to the wire. Therefore, combinatoric circuits have no loops. They are stateless, and realize logic formulas. For building these circuits we have and-gates, Fig. 4.1, or-gates, Fig. 4.2, and inverters, Fig. 4.3, at our disposal. In circuit diagrams the supply rails



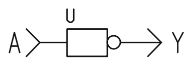
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 4.1 And-gate with two inputs and its truth table



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 4.2 Or-gate with two inputs and its truth table



A	Y
0	1
1	0

Fig. 4.3 Inverter and its truth table

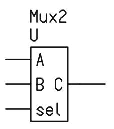
are shown only when necessary for understanding the circuit. Instead of drawing separate inverters we indicate inverted inputs and inverted outputs with small circles. When drawing a circuit we strive for presenting the circuit's function clearly. We delegate the minimization the number of gates used by the circuit to an electronic design automation (EDA) tool.

A multiplexer, Fig. 4.4, switches its output between two or more inputs under the control of select inputs.

A decoder, Fig. 4.5, decodes binary input. It has an output for any possible combination. The outputs of a decoder often drive inverted inputs. Therefore, they usually are inverted themselves.

A gate cannot react instantaneously to changes at its inputs. It changes its output to the new value within its propagation delay time t_{pd}. If several inputs change their value almost instantaneously, the output of the gate is Forcing Unknown until the time t_{pd} has elapsed.

The temporal behavior of logic circuits is described by timing diagrams. Such a diagram shows how the states of the circuit's wires evolve over time. Both a Weak and a Forcing One are represented by ⊞; a Weak and a Forcing Zero are represented by ⊞. A Weak and a Forcing Unknown shows as ■; a Don't Care shows as ⊞. High Impedance shows as ⊞ .



sel	C
0	A
1	B

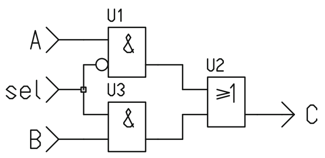
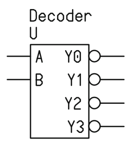


Fig. 4.4 Multiplexer with two inputs, its truth table and its circuit



A B	Y0	Y1	Y2	Y3
0 0	0	1	1	1
0 1	1	0	1	1
1 0	1	1	0	1
1 1	1	1	1	0

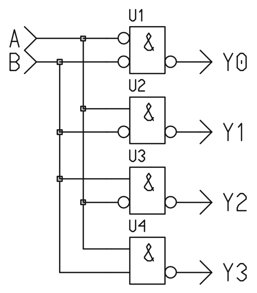
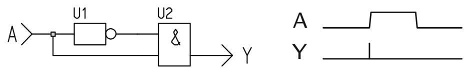


Fig. 4.5 Two to four decoder, its truth table and its circuit

Fig. 4.6 Circuit with a logic race. Timing diagram with a glitch on the *right*



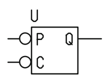
Consider the circuit in Fig. 4.6. Naively we may think that the output Y of the and-gate U2 is constant Zero. When the input A switches from Zero to One the two inputs of the and-gate may be both One momentarily because of the propagation delay of the inverter U1. Whether the output Y reflects this depends on the physical properties of the circuit. The output may become One momentarily, a so-called glitch; it may only make it to halfway between the supply rails before returning to Zero, a runt pulse; or it may not move much at all. A logic hazard occurs when two inputs to a circuit change their states (almost) simultaneously, or when two signals derived from the same inputs reconvene at the same gate.

4.4 Sequential Logic

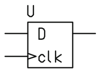
Sequential circuits have state. Flip-flops store this state. The simple flip-flop, Fig. 4.7, is the building block for static random access memories. A Zero at the clear input C clears the flip-flop unconditionally. A Zero at the preset input P sets the flip-flop provided the input C is One.

A rising-edge triggered flip-flop changes its state at points in time determined by the rising edges, the transitions from Zero to One, of a signal fed into the flip-flop's clk input. A falling-edge triggered flip-flop reacts to the One to Zero transitions at its clk input. A rising-edge triggered D flip-flop, Fig. 4.8, copies at the points in time defined by the rising edges at its clk input the values at these times at its D input to

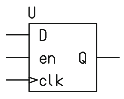
Fig. 4.7 Simple flip-flop and its truth table



C P	Q
0 –	0
1 0	1
1 1	Q



clk	D	Q
0	–	Q
1	–	Q
↑	D	D



en	clk	D	Q
0	–	–	Q
0	↑	–	Q
1	0	–	Q
1	1	–	Q
1	↑	D	D

Fig. 4.8 *Left side* D flip-flop and its truth table. *Right side* D flip-flop with clock enable and its truth table. The symbol ↑ indicates a rising edge

its Q output. The output Q will change within the flip-flop's propagation delay. The clock enable input en, when available, allows skipping clock edges.

4.4.1 Flip-Flops and Metastability

For ensuring correct operation a D flip-flop requires its D input to be stable before a rising edge on its clk input for its setup time t_{su}. Furthermore, it requires D to be still stable after the edge for its hold time t_{h}. The outputs of any gate or flip-flop react to changes at the inputs not earlier than specified by the minimum value for the propagation delay time t_{pd}. Within a logic family this time is longer than the hold time t_{h} of a flip-flop. The signal driving the clock enable input of a flip-flop must satisfy similar requirements. Violating the flip-flop's setup or hold time results in unpredictable behavior. The flip-flop's output may swing to a voltage in the middle of the forbidden zone, indicated with 〰 in timing diagrams, and stay at this voltage for an arbitrary time; the flip-flop has become metastable. Eventually, the output will fall back randomly either to Zero or to One, see Fig. 4.9. The times the flip-flop stays in the metastable state are distributed exponentially.

The signals driving the preset and clear inputs of a simple flip-flop must have a minimum pulse width. If this pulse width requirement is violated the flip-flop may become metastable. Metastability is not a deficiency of flip-flops; any bistable mechanism has a metastable state.

4.4.2 Synchronous Logic

Glitches and metastability seem to turn the design of digital logic into an endeavor with unpredictable outcome. Luckily we can identify a design discipline, the

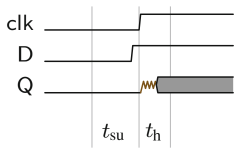


Fig. 4.9 Metastable behavior resulting from a setup time violation

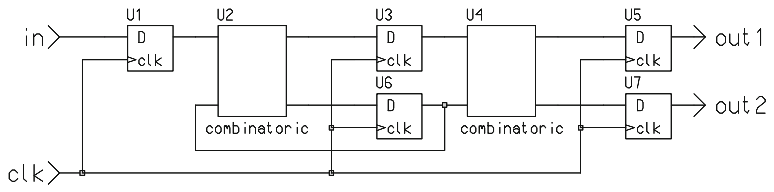


Fig. 4.10 Synchronous design style. All flip-flops receive their clocks from the same source. The inputs to combinatoric parts are provided by the outputs of flip-flops or by input signals synchronized to the global clock. The outputs of combinatoric blocks are caught by flip-flops

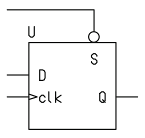
synchronous discipline, avoiding both problems. A synchronous circuit uses either rising-edge triggered or falling-edge triggered flip-flops only. For our discussion we choose the rising edge. The signals driving the clock inputs of the flip-flops, the clock signals, are periodic and exhibit constant frequency and phase relationships. Moreover, each input signal of the circuit is synchronous to one of the clock signals. More precisely, each edge of the input signal must fall into a short time interval after a rising edge of the associated clock signal. An input must be delayed for at least a flip-flop's hold time t_h. Outputs may still exhibit glitches. If a glitch-free output is required it must be the direct output of a flip-flop. The write-enable line of asynchronous memory for example must be glitch-free.

In the simplest case, Fig. 4.10, a single clock signal clk drives the clock inputs of all flip-flops. In timing diagrams arrows indicate the clock edges at which the flip-flops change state. Input signals, synchronous with the clock, or the outputs of flip-flops drive the inputs of combinatoric circuits. After each rising edge of the clock signal the outputs of the combinatoric parts may exhibit glitches and runt pulses. The flip-flops at the outputs of a combinatoric part hide this behavior from the rest of the system. The outputs must have settled down to the correct values at least a setup time before the next rising edge of the clock signal. The times required for the outputs of the combinatoric circuits to settle together with the setup time of the flip-flop determine the maximum clock frequency the circuit is able to operate at. Such a synchronous circuit is active right after each rising edge of the clock. For the rest of each clock cycle it sits idle.

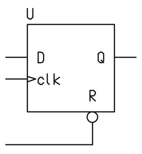
More elaborate synchronous systems use not only the single clock signal clk but also signals derived by clock dividers from clk for triggering flip-flops. The frequencies of the derived signals are integer fractions of the frequency of the clock signal.

4.4.3 State Machines

When we implement a state machine in synchronous logic we may assume that the inputs are synchronous with the common clock signal. First we represent each state with a separate flip-flop. One flip-flop—representing the state the machine is in—is set; all others are reset. This scheme is called one-hot encoding. For establishing the state machine's initial state at startup we use D flip-flops with set or reset input,



clk	D	S	Q
0	–	–	Q
1	–	–	Q
↑	–	0	1
↑	D	1	D



clk	D	R	Q
0	–	–	Q
1	–	–	Q
↑	–	0	0
↑	D	1	D

Fig. 4.11 *Left side* D flip-flop with synchronous set and its truth table. *Right side* D flip-flop with synchronous reset and its truth table. The symbol ↑ indicates a rising edge

Fig. 4.11. If the set input S of a settable D flip-flop is Zero during a rising edge on the flip-flop's clock input the flip-flop is set regardless of the D input. If the reset input R of a resettable D flip-flop is Zero during a rising edge the flip-flop is reset. We represent the initial state of the state machine with a settable flip-flop and all other states with resettable flip-flops. A common reset signal, synchronous to clk, drives the single set and all reset inputs. When the reset signal becomes One the state machine starts to work.

Next we realize the state transitions. All gates we use in the sequel shall have the required number of inputs. To each state S we associate an or-gate. We wire the gate's output to the D-input of the flip-flop representing S. We consider each transition T terminating in S. Let S' be the state in which T originates. An and-gate represents T; its output is wired to an input of the or-gate associated with S. The output of the flip-flop representing S' is wired to an input of the and-gate. The other inputs of the and-gate are wired to the inputs of the state machine, directly for the inputs that must be One for the transition to be taken, via an inverter for those that must be Zero.

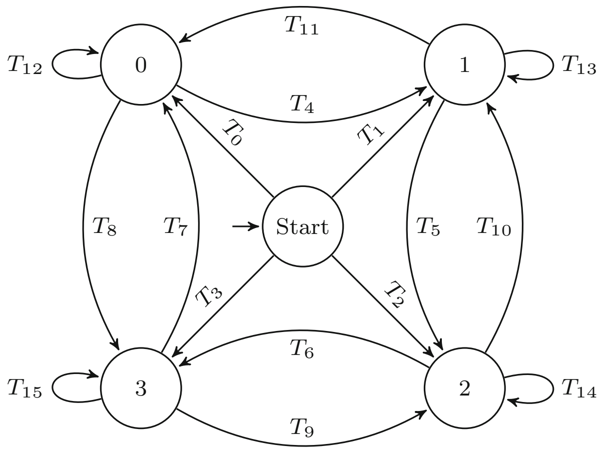
The circuit in Fig. 4.14 realizes the state machine in Fig. 4.12; Fig. 4.13 shows a timing diagram for this circuit. The flip-flops U0–U3 represent states 0–3; the flip-flop U4 represents the state Start. The gates U5 and U6 for example, realize the transition T_{12}, the gate U8 realizes T_{11}, U9 realizes T_7 and U10 realizes T_0.

Last we have to realize the output function. We consider each output O of the state machine. We wire the output of an or-gate to the output O. For every combination of a state S with the input values for which O is one we add an and-gate. We wire the output of the and-gate to an input of the or-gate associated with O. We wire the output of the flip-flop representing S to an input of the and-gate. The other inputs of the and-gate are wired to the inputs of the state machine, directly for the inputs that must be One for the output O to become One, via an inverter for the inputs that must be Zero for O to become One.

In Fig. 4.14 the gates U29–U33 realize the output function for up. For the output function of the cnt output we reuse these gates and add gates U34–U39.

4.4.4 Transparent Latch

Another storage element is the transparent latch, Fig. 4.15. While its latch enable input LE is One the latch copies the value of its D input to its output Q; the latch is transparent. The moment LE becomes Zero the latch freezes its output Q.



	Condition		Condition		Condition		Condition
T_0	(A, B) = (0, 0)	T_4	(A, B) = (1, 0)	T_8	(A, B) = (0, 1)	T_{12}	**default**
T_1	(A, B) = (1, 0)	T_5	(A, B) = (1, 1)	T_9	(A, B) = (1, 1)	T_{13}	**default**
T_2	(A, B) = (1, 1)	T_6	(A, B) = (0, 1)	T_{10}	(A, B) = (1, 0)	T_{14}	**default**
T_3	(A, B) = (0, 1)	T_7	(A, B) = (0, 0)	T_{11}	(A, B) = (0, 0)	T_{15}	**default**

State	A	B	cnt	up	State	A	B	cnt	up	State	A	B	cnt	up	State	A	B	cnt	up
0	0	0	0	0	1	0	0	1	0	2	0	0	0	0	3	0	0	1	1
0	1	0	1	1	1	1	0	0	0	2	1	0	1	0	3	1	0	0	0
0	1	1	0	0	1	1	1	1	1	2	1	1	0	0	3	1	1	1	0
0	0	1	1	0	1	0	1	0	0	2	0	1	1	1	3	0	1	0	0
Start	–	–	0	0															

Fig. 4.12 State transition diagram, transitions, and output function of a sample state machine. The state machine has two inputs A and B and two outputs cnt and up

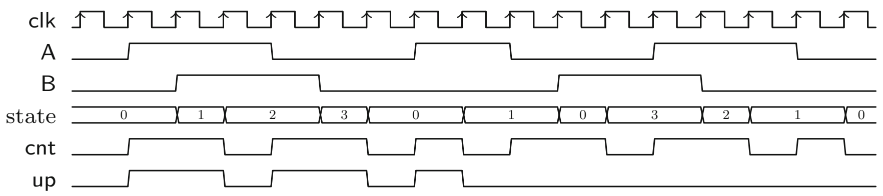


Fig. 4.13 Timing diagram of the circuit in Fig. 4.14 realizing the state machine in Fig. 4.12. The state is not visible outside the state machine

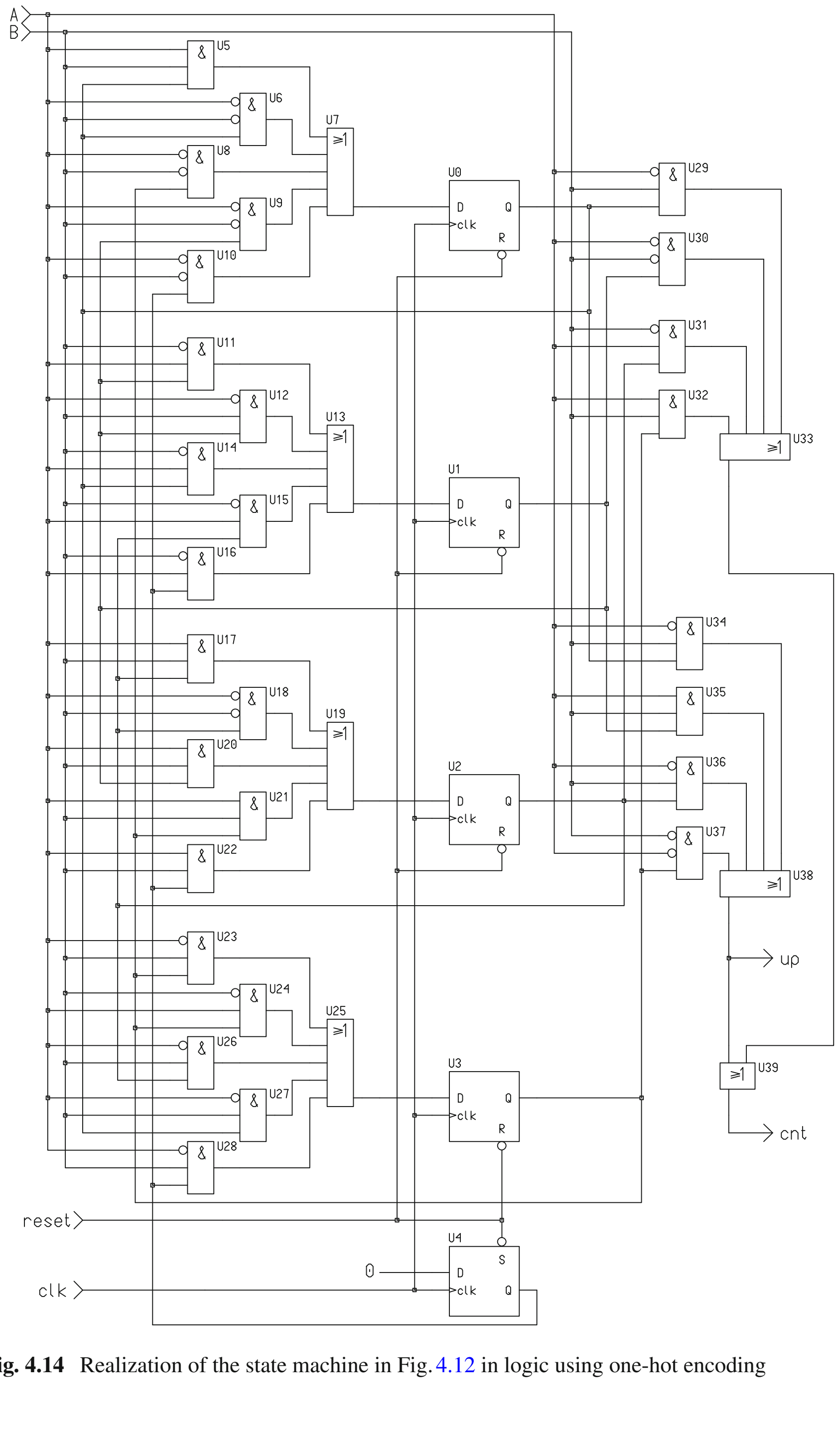
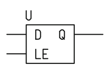


Fig. 4.14 Realization of the state machine in Fig. 4.12 in logic using one-hot encoding



LE	D	Q
0	–	Q
1	D	D

Fig. 4.15 A transparent latch and its truth table

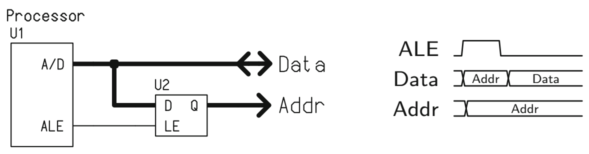


Fig. 4.16 Circuit and timing of a latch catching addresses off a multiplexed bus. Thick lines indicate parallel wires

Some processor chips use the same lines for transferring addresses and data. During a memory access the processor first sends the address and then switches the lines to sending or receiving data. It provides an address latch enable signal ALE to indicate when it outputs the address. A transparent latch catches the address and provides it to the memories for the rest of the access. Edge triggered flip-flops can be used, but they make the address available to the rest of the system only after the falling edge of ALE, while a transparent latch makes the address available shortly after the rising edge of ALE, Fig. 4.16.

In order to work perfectly a transparent latch requires its D input to be stable before a falling edge on its LE input for at least its setup time t_{su}. Furthermore, it requires its D input to be stable after a falling edge for at least its hold time t_{h}. Violating the setup time or the hold time of a transparent latch may cause metastability.

4.5 Clock Domains

The synchronous design discipline solves the problems associated with glitches and metastability. Several problems remain, though. Building a fully synchronous system forces us to reliably distribute a common clock signal to all parts of the system. Transmitting a high-frequency clock over long distances is technically challenging. Moreover, the clock source becomes a single point of failure. For these reasons practically all spatially dispersed systems have several clock sources. The physical world follows its own rhythms; signals originating there will be asynchronous with the clocks of any system processing these signals.

The systems we build consist of several clock domains. A clock domain has a single clock generator. All edges within a clock domain have fixed temporal relation to the domain's clock signal. A signal entering a clock domain must be synchronized

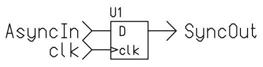


Fig. 4.17 A single stage synchronizer. The probability for it to become metastable is too high for this synchronizer to be useful

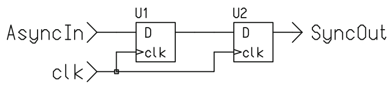


Fig. 4.18 A two stage synchronizer. In order to further reduce the probability of failure additional flip-flops can be added

to the domain's clock. The circuit in Fig. 4.17 will do that, but there is a hitch. At some point the signal to be synchronized, AsyncIn, will exhibit an edge violating the flip-flop's setup or hold time. The flip-flop may become metastable long enough to cause havoc in the downstream logic. The probability of malfunction is too high for this circuit to be useful.

In principal, we cannot build a perfect synchronizer. What we can do, however, is to reduce the probability of malfunction to very low levels by giving the synchronizing flip-flops time to drop out of the metastable state. The circuit in Fig. 4.18 does just that. Adding a third flip-flop will reduce the probability of failure further. Using metastability-hardened flip-flops helps too, but might come at the cost of increased power consumption.

4.6 Digital Building Blocks

A register consists of a number of flip-flops controlled by a common clock signal and, if available, by a common clock enable. Usually a register stores a word of data.

An accumulator is a register paired with an adder-subtractor circuit. In a single clock cycle an accumulator can initialize the register with the data at its inputs, clear the register, add or subtract the data at its inputs from the data stored in its register and store the result in its register.

Counters consist of flip-flops combined with combinatoric logic so that for each clock cycle the outputs take a step in the code the counter counts in. A simple binary counter counts up to a predetermined value, not necessarily a power of two, and wraps around to zero. An up-down counter takes a step up or a step down in each clock cycle depending on a control input.

A shift register converts a serial input into a parallel output or a parallel input into a serial output. It consists of a linear arrangement of flip-flops having a common clock. The output of one flip-flop is wired to the input of the next. In shift registers allowing parallel loading multiplexers are introduced between the flip-flops for switching between the load and the shift operation.

4.7 Realization of Logic Circuits

So-called random logic is composed of digital building blocks where each building block resides in a separate integrated circuit. A circuit board, designed for the circuit, realizes the connections between the building blocks. The building blocks usually are chosen from a single logic family. A logic family consists of integrated circuits having the same electrical specifications and compatible timing. In particular the minimum propagation delay times of the gates and the hold times of the flip-flops match. The individual integrated circuits implement a single building block, such as a counter, a decoder, or a shift register. Although the number of the building blocks available within a family has dwindled, we still can build small digital systems by hardwiring a handful of these chips together. Prototyping is rather painful, every bug forces us to rewire parts of the circuit.

Programmable logic chips allow us to configure the chips' functions freely using electronic design automation tools. A programmable logic chip consists of a number of identical cells and a configurable interconnection network. A cell consists of a configurable building block for combinatoric logic and a flip-flop. The actual structure of a programmable logic chip is of secondary importance to the designer. Design tools translate a circuit into the resources available on the chip automatically.

A complex logic device, CPLD for short, uses logic gates with configurable switches as building block for the combinatoric logic in each cell. Nonvolatile memory cells control the switches in the cells and in the interconnection network. Complex logic devices are available with up to 300 cells in packages with up to 200 connections usable for logic signals. They lend themselves well to the implementation of autonomous periphery in microprocessor-based systems. Most complex logic devices can be reconfigured. Fixing a bug requires us to erase the memory cells and reprogram them.

A field programmable gate array, FPGA for short, uses random access memory storing truth tables as building block for the combinatoric logic in each cell. Some of this memory can be combined for providing on-chip storage capabilities. The content of random access memory cells configures the interconnection network. An FPGA must be configured by initializing the memories before it can operate. Besides the cells and the interconnection network an FPGA may contain complex functions such as multipliers or even complete microprocessors. The largest FPGAs available contain more than two million cells in packages with up to 1,200 connections usable for logic. They lend themselves for the implementation of complete systems on a single chip. Changing the configuration entails rebooting the chip.

Custom logic chips offer the highest performance and flexibility. Developing such a chip, however, is costly. Fixing a bug usually requires a new production run.

Small designs can be accomplished by drawing circuit diagrams. Anything larger than a peripheral is better designed using a text-based hardware description language such as VHDL or Verilog (IEEE 2006, 2009). A synthesis tool translates a design into the primitives provided by the target technology. The result is a circuit diagram consisting of the target's primitives only. A so-called net list represents this circuit diagram. Another tool places the primitives on the chip and routes the connections. For finding bugs one simulates first at the design level without considering propagation delays, setup times, and hold times. After synthesis the design is simulated again, this time considering the propagation delays, setup times, and hold times of the primitives used. After placement and routing the design is simulated a third time also considering the delays introduced by the interconnections. Small designs can be completed using free tools provided by the silicon vendors. The design of a large digital system, however, is a complex project requiring an elaborate and costly toolchain.

4.8 Support Circuits

Clock generators, often integrated into logic chips, either use a quartz crystal or a ceramic resonator as timing element. Quartz crystals provide higher precision than ceramic resonators but are more costly and require more space. The maximum frequency of such a clock generator's output signal is limited to about 20 MHz. A frequency multiplier produces the high-frequency signal required by the logic circuit from the output of the clock generator.

A reset generator monitors the supply voltage. It keeps the system in the reset state until the supply voltage is within specifications. The voltage at which the reset generator releases the system should be halfway between the minimum voltage the system requires for operation and the minimum voltage the power supply delivers.

4.9 Analog Aspects of Fast Logic

When we realize a high-speed digital system on a circuit board, analog and high-frequency effects deserve our attention. Disregarding these effects will at best result in a system exhibiting excessive power consumption and excessive emission of radio-frequency interference. At worst the system will not work at all.

Logic inputs and logic outputs exhibit a parasitic capacitance to the negative supply rail of about 10 pF. When we design a bidirectional bus driven with tri-state outputs we have to ensure each bus line is at a valid voltage when no output drives the bus. When using a pull-up resistor we also have to obey the required rise time t_r of the inputs. The pull-up resistor has to provide enough current to load the paralleled parasitic capacitances from a logic Zero to a logic One within t_r. Let the supply voltage be 3.3 V, the minimum voltage for a logic One be 2.15 V, the rise time t_r be 33 ns, and the combined parasitic capacitance be 50 pF. Choosing a pull-up with a resistance of 560 Ω results in a voltage of about 2.3 V after the rise time. Whenever some output drives the bus line to a logic Zero a rather high current of about 6 mA flows through the pull-up. A bus hold device does not consume such a high current. Unlike a pull-up it does not impose a logic One onto the bus line. It merely keeps the bus line at the state to which a low-impedance output has pulled it previously. Holding the bus line requires enough current for the combined leakage currents of all inputs and outputs, about 50 μA in our example.

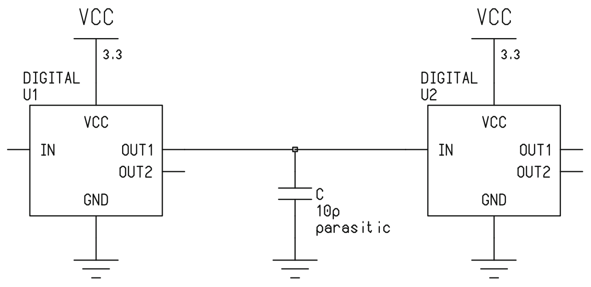


Fig. 4.19 A digital output driving an input via a wire with parasitic capacitance

The outputs of fast logic chips produce signals with rise and fall times of about 1 ns. These signals have an appreciable frequency content well above 3 GHz. A copper trace on a circuit board delays electrical signals by about 55 ps per centimeter of length. Therefore, a rising edge has a spatial length l of 18 cm. For a wire shorter than about $l/6$ we can assume that the wire distributes its state instantaneously to all inputs connected to it. Circuits smaller than $l/6$ are called lumped, larger ones are called distributed. In distributed circuits we have to take into account wave propagation effects on the wires, such as reflections, see, Johnson and Graham (1993).

A digital output has to charge or discharge the combined parasitic capacitance of the wire and the inputs it drives, Fig. 4.19. When the output is driven to One, the parasitic capacitance will store an energy of 50 pJ. When the output switches from One to Zero the parasitic capacitance has to be discharged by a current that flows into the output and further into ground. Thus, a 100 MHz square-wave signal transmitted across such a wire will make the output dissipate a power of 50 pJ 100 MHz = 5 mW. If the output produces rise and fall times of 1 ns, an average current of 33 mA will flow out of the output during a rising edge and into the output during a falling edge. The charge for these currents must be stored locally at the output. Digital chips have several connections for the positive and the negative supply rail. Each such pair of connections must be decoupled with a capacitor with excellent high-frequency properties. A capacitance of about 100 nF will do. These decoupling capacitors store the charge for the currents flowing during the fast edges. In order to close the electrical circuit the currents flowing from the output to the inputs over the wire have to flow back to the output in the ground connection. Given a choice the return current will take the path of the wire. Any detour from the ideal results in the emission of radio frequency interference. A ground plane, a solid layer of copper connected to the negative supply rail, creates ideal paths for all return currents. Together with a plane connected to the positive supply rail it forms a capacitor with excellent high-frequency properties.

In the circuit in Fig. 4.20 the digital component U1 drives the input of another digital component U2. The ground connection of U1 has an inductance of 20 nH which is typical for a trace with a length of about 3 cm on a circuit board combined with the inductance of the ground pin itself. When the output OUT2 of U1 swings from high to low within one nanosecond, an average current of 33 mA has to flow into OUT2 and out of the GND pin of U1 for the duration of the edge. For simplicity we

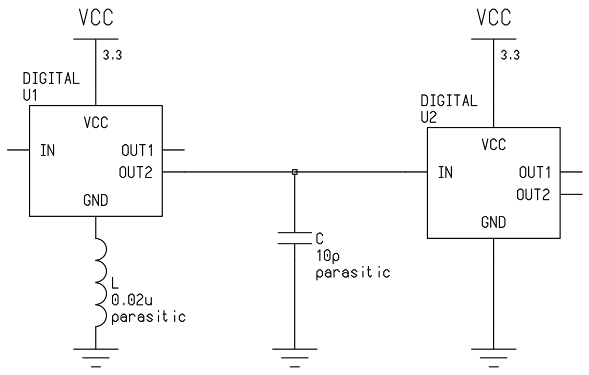


Fig. 4.20 A digital output driving an input via a wire with parasitic capacitance. The ground connection of the driving chip has parasitic inductance

assume that the current ramps up linearly to 66 mA during the first 500 ps and down linearly to 0 mA during the second 500 ps. The voltage at the GND pad of the silicon die inside of U1 will jump to $20\,\text{nH}\ 132\,\text{mA}\,\text{ns}^{-1} = 2.64\,\text{V}$ during the first half and to $-2.64\,\text{V}$ during the second, before returning to ground. Such an exertion, known as ground bounce, definitely interferes with the function of the circuit. The shortest possible connections of the GND pins to the ground plane minimizes the parasitic inductance and fixes ground bounce.

4.10 Bibliographic Notes

The construction of complex digital circuits is covered in great depth by many books on very large scale integrated circuit design. The book by Kaeslin is both comprehensive and readable (Kaeslin 2008). Before one embarks on the construction of a circuit board for high-speed logic the book by Johnson and Graham is required reading (Johnson and Graham 1993). Their second book provides even more in-depth information on distributing high-speed signals (Johnson and Graham 2003). The two most widely used hardware description languages are defined in IEEE (2009) and IEEE (2006). The discussion in 4.2.1 follows IEEE (1993). The book by Abelson and Sussman contains a beautiful program for simulating digital circuits written in Scheme (Abelson and Sussman 1996).

4.11 Exercises

Exercise 4.1 Design an edge detector. The edge detector has a clock input clk and a signal input A. The signal feeding A is synchronous to the clock. It has two outputs A↑ for indicating a rising edge at the beginning of the current clock cycle and A↓ for indicating a falling edge.

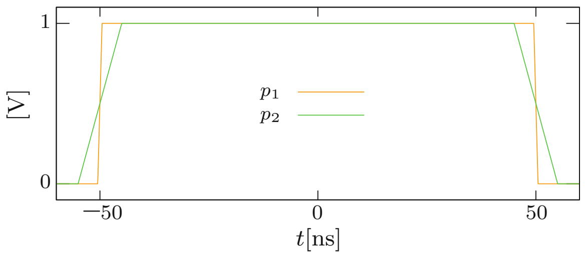


Fig. 4.21 The pulse p_1 has a rise and a fall time of 1 ns, the pulse p_2 of 10 ns

Exercise 4.2 Implement the state machine described in Fig. D.13.

Exercise 4.3 Compare the frequency content of the two pulses p_1 and p_2 in Fig. 4.21. The pulse p_1 has a rise and a fall time of 1 ns, while p_2 has a rise and a fall time of 10 ns.

4.12 Lab Exercises

Exercise 4.4 Implement the edge detector you designed in Exercise 4.1.

Exercise 4.5 Implement the circuit in Fig. 4.14.

Exercise 4.6 Implement the circuit you designed in Exercise 4.2.

References

Abelson H, Sussman GJ (1996) Structure and interpretation of computer programs, 2nd edn. MIT Press, Cambridge

IEEE (1993) IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std-logic1164). IEEE Std 1164–1993.doi:10.1109/IEEESTD.1993.115571

IEEE (2006) IEEE Standard for Verilog Hardware Description Language. IEEE Std 1364–2005 (Revision of IEEE Std 1364–2001).doi:10.1109/IEEESTD.2006.99495

IEEE (2009) IEEE Standard VHDL Language Reference Manual. IEEE Std 1076–2008 (Revision of IEEE Std 1076–2002).doi:10.1109/IEEESTD.2009.4772740

Johnson H, Graham M (1993) High speed digital design: a handbook of black magic. Prentice Hall Modern Semiconductor Design Series, Upper Saddle River

Johnson H, Graham M (2003) High-speed signal propagation: advanced black magic, 1st edn. Prentice Hall Press, Upper Saddle River

Kaeslin H (2008) Digital integrated circuit design: from VLSI architectures to CMOS fabrication, 1st edn. Cambridge University Press, New York