Guidelines for Synchronous Digital Circuit Design

Some of the rules given in this chapter are automatically obeyed by modern synthesizers, others have to be obeyed by the designer. Many ideas of this chapter were taken from [5.3].

1 Synchronous Circuits

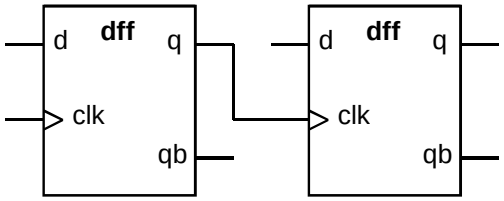
Experience has shown that the safest method for design and test of digital circuits and systems is synchronous circuit design.

A design is synchronous, if

- all data storage elements are clocked and in normal operation change state only in response to the active edge of the clock signal,
- the same active edge of the clock signal is applied at precisely the same time point at every clocked cell in the design.

Not recommended: Non-synchronous changes of FlipFlops states:

Figure 1:
FlipFlop driving clock input of an other FlipFlop



Do not gate clocks !

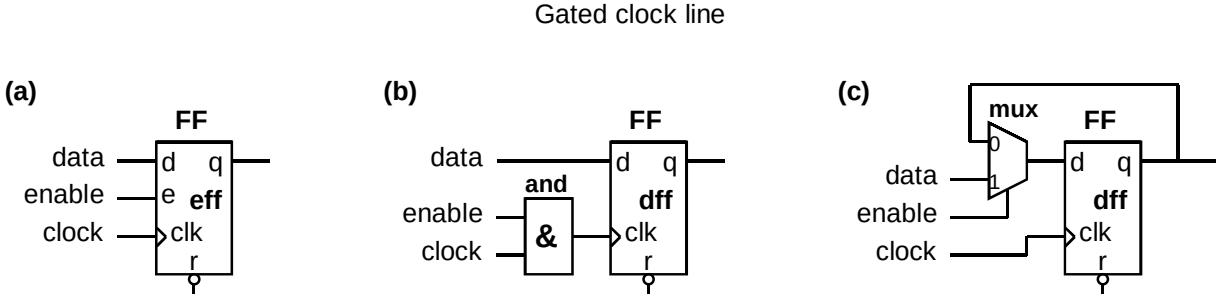


Figure 2: (a) Enabled D-Flipflop, realized: **(b)** bad: gating the clock, **(c)** good.

Figure 3:
Double edged clocking is hardly testable. Double the clock frequency if necessary.

Exception: RAM may require double edged clocking.

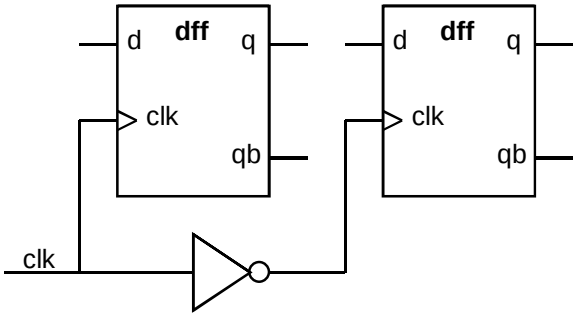
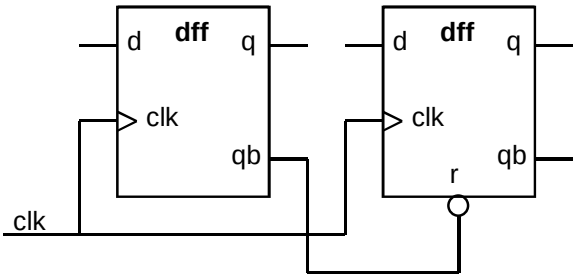


Figure 4:
FlipFlop driving an asynchronous, local reset of an other FlipFlop.



Exercise:

a) What is the definition of a synchronously clocked circuit?
b) Which asynchronous change of data storage elements do we accept in a synchronously clocked design?
c) Is it good design practice to gate clocks?

2 Clock Buffering

The demand of synchronous clocking requires an appropriate clock. Two problems have to be avoided: clock skew and slow clock edge.

Figure 5:
Clock skew: The active clock edge of the clock signal is not applied at the same time point.

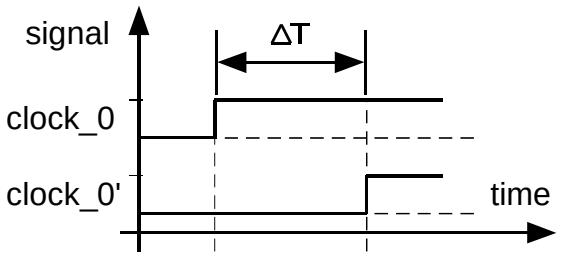
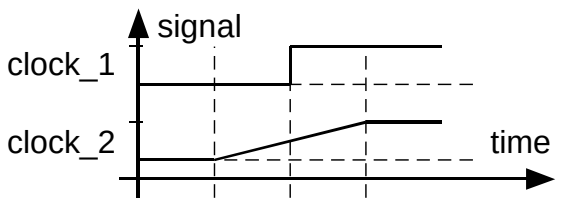


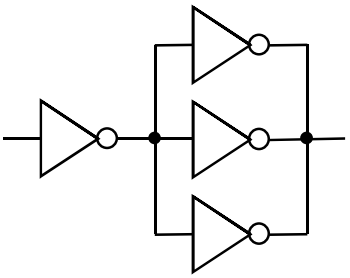
Figure 6:
Slow clock edge: Signal `clock_2` has a slow edge compared to signal `clock_1`.

- Avoid excessive fan-out!



Non-recommended Clock Buffering

Figure 7:
Parallel buffers for geometric clock buffering are not recommended.



Do not use parallel buffers to increase buffer strength as shown in Fig. 7. Due to parameter scattering the buffers will not change state at the same time point and dissipate energy while their connected outputs drive different states values.

Clock skew originates from unbalanced clock tree depth and unbalanced loading of clock buffers as shown in Fig. 8(a). Slow clock edges are caused by excessive fan-out, as illustrated in Fig. 8(b).

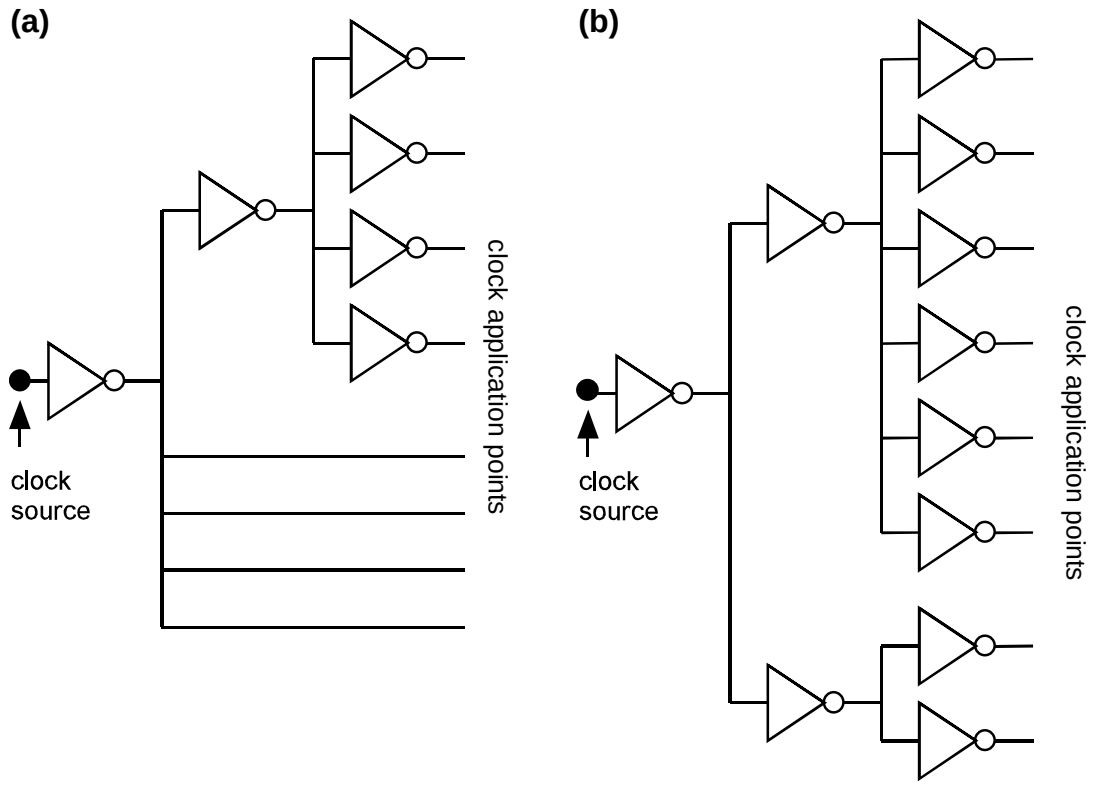
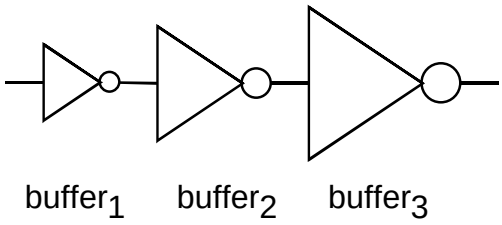


Figure 8: (a) Unequal depth of clock buffering, **(b)** Unbalanced fan-out on clock buffers

Recommended Circuits

To buffer a week signal we take a number of serial buffers with increasing strength, i.e. increasingly wide driver MOSFETs. This is called geometric buffering. It can be shown mathematically, that geometric clock buffering is optimal with respect to speed when the buffer size increases by a factor e from $buffer_n$ to $buffer_{n+1}$, where e is the Euler number $e=2.718...$. This maximum is not very sensitive to the load. Therefore, increasing the buffer strength by a factor 2, 3 or 4 is good, too. Increasing the buffer strength with larger factors increases the buffer's capacitive loads given by the wide gates of the following buffers. Increasing the buffer strength with low factors makes the buffers fast due to small capacitive loads but requires more serial buffer stages.

Figure 9:
Geometric clock buffering is optimal when the buffer strength increases with every buffer by a factor $e=2.7$.



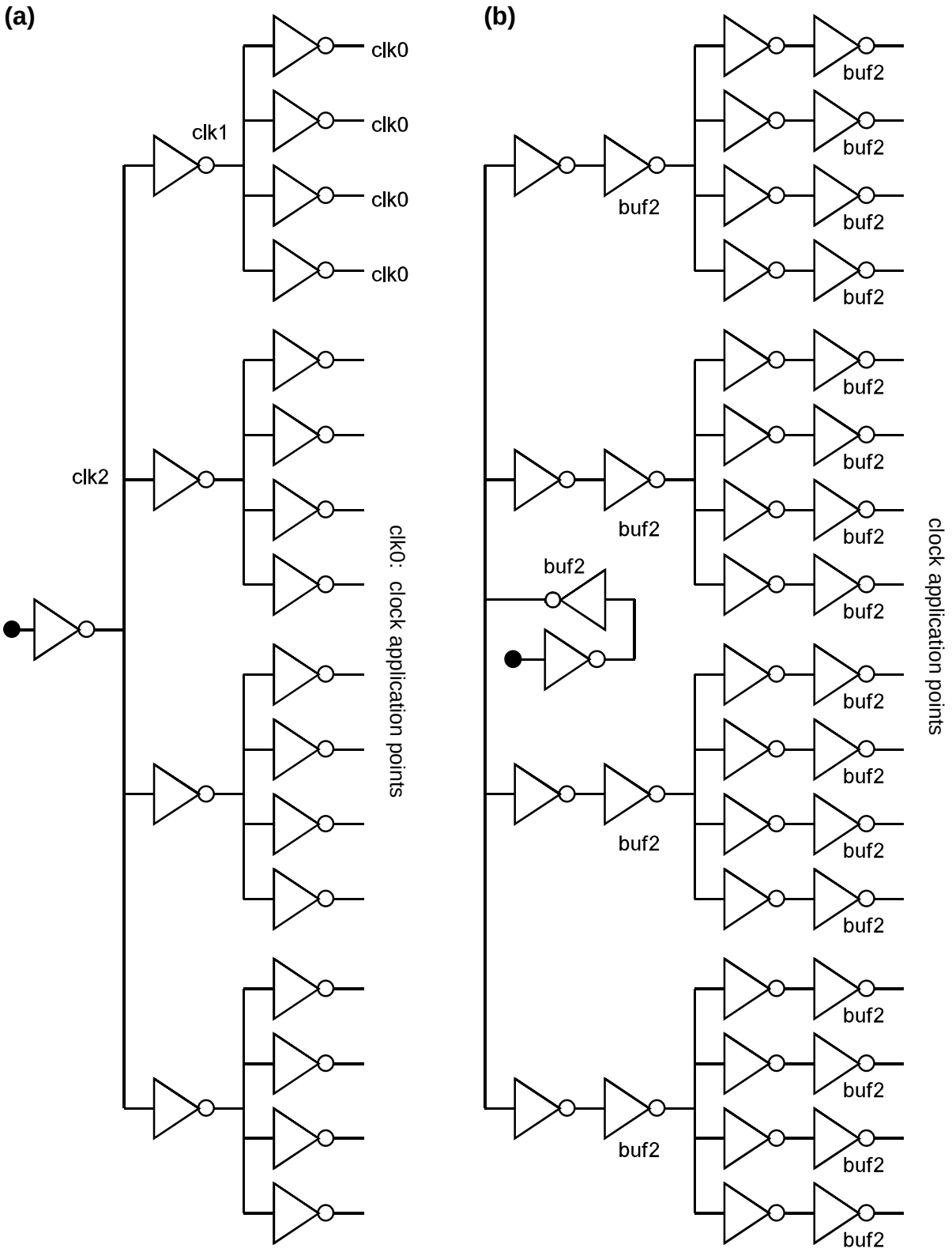


Figure 10: (a) Balanced clock tree buffering, **(b)** Combined geometric tree buffering.

Fig. 10(a) shows balanced clock tree buffering, which is recommended. An indexed clock level naming scheme is proposed to avoid confusion. If a stronger buffer type `buf2` is available, combined geometric tree buffering according to Fig. 10(b) is good, too.

Exercise:

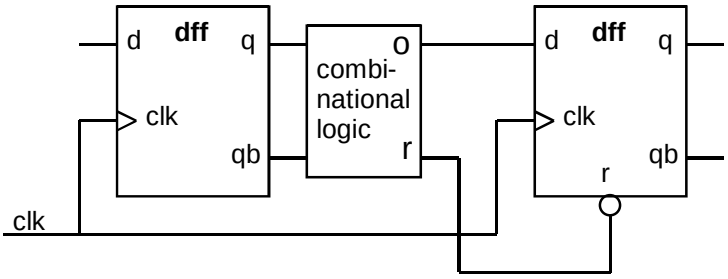
a) What is clock skew?
b) What is a slow clock edge?
c) What is geometric buffering?
d) What is tree buffering?
e) What is **balanced** tree buffering?

3 Resets

The circuit must be brought in to a known state, both within test and normal operation, within a stated and agreed number of clock cycles.

Not Recommended: Local Asynchronous Reset

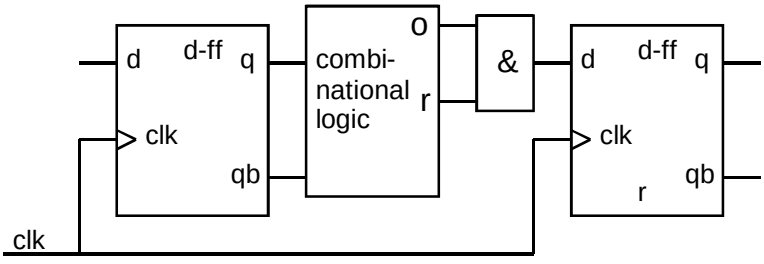
Figure 11:
Not recommended: Local asynchronous reset.



Recommended: Local Synchronous Reset

The (active low) reset signal (r) is gated with the d-input of the second FlipFlop, making it synchronous.

Figure 12:
Recommended: Local synchronous reset.



Recommended: Global Asynchronous Reset

A single external reset signal (rext) is connected to all FlipFlops. The buffering, which may be required, is not shown.

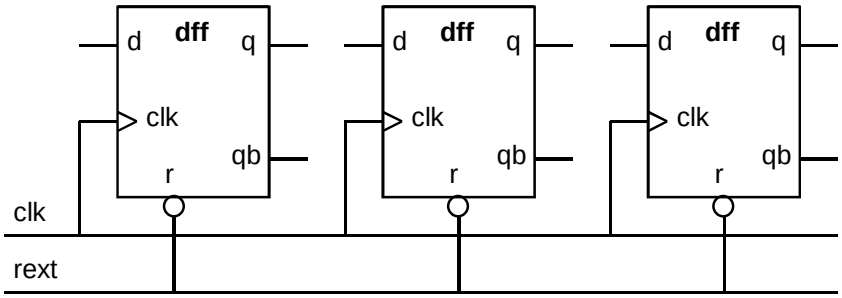


Figure 13: Recommended: Global asynchronous reset.

4 Shift Registers

Shift registers are particularly intolerant of clock skew.

Not Recommended: Shift registers with clock buffer chains

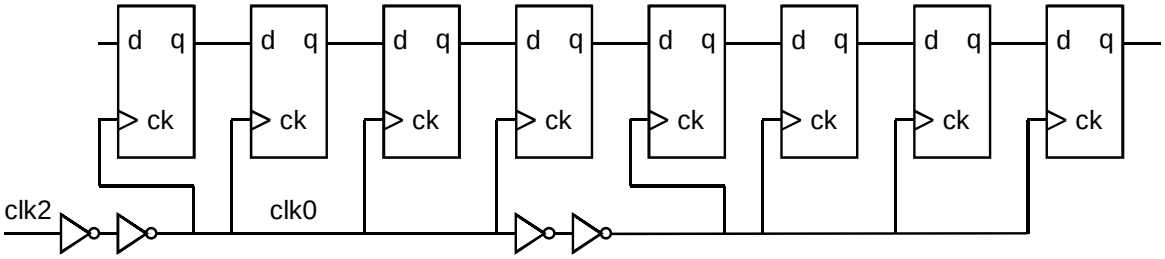


Figure 14: Shift registers with forward chain of clock buffers

Shift register with reverse chain of clock buffers as shown in Fig. 15 move the problem to the input of the first FlipFlop in the chain. This circuit works well if the first FlipFlop captures asynchronous data.

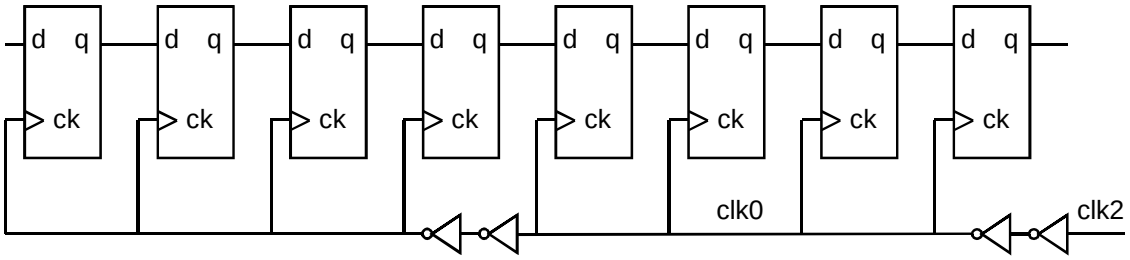


Figure 15: Shift registers with reverse chain of clock buffers

Recommended: Shift register with balanced clock tree buffering

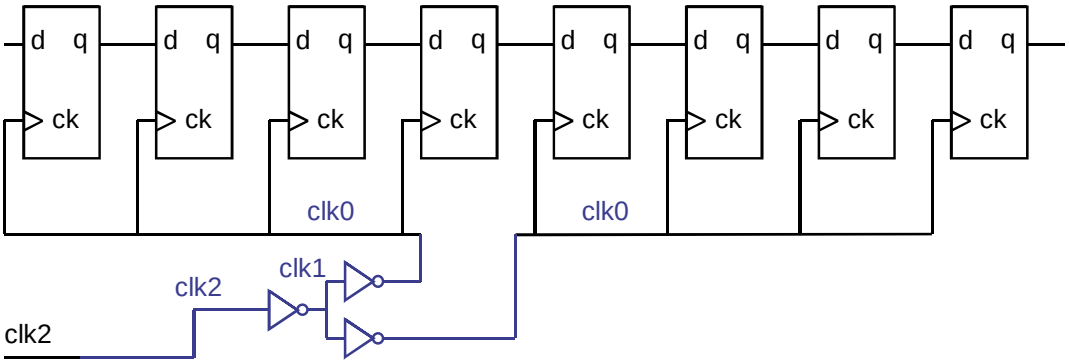


Figure 15: Shift registers with balanced clock tree buffering.

Exercise:

a) Why are shift registers particularly intolerant of clock skew?
b) Is a reverse chain of clock buffers a solution within a synchronous design?
c) A reverse chain of clock buffers is acceptable if ...?

4.1.1 Asynchronous Signal Capturing

When a FlipFlop captures an asynchronous event, there is a probability of metastability, i.e. the output floats around ½(V_{DD}+V_{SS}). The probability of metastability decreases exponentially with time. In other words: it increases exponentially with speed. The effective clock period can be increased by using several FlipFlops in series.

- For large designs, inter-block communication is similar to external asynchronous interfacing.
- Use Schmitt trigger inputs in noisy environments.

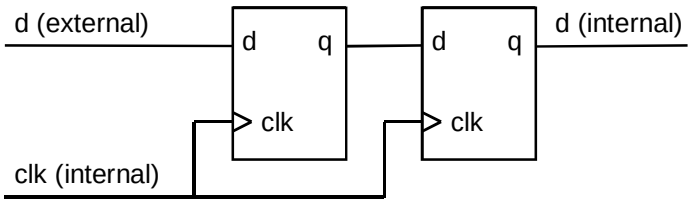
Non-Recommended Circuits

Not recommended is any circuit using a complicated feedback loop to capture an asynchronous input. The functions of such circuits is obscure, and they run the risk of creating more problems than they solve. They are also very sensitive to noise

Recommended Circuits

Two or more D-FlipFlop in series as shown in Fig. 16 or the specific asynchronous handshake circuit shown in Fig. 17.

Figure 16:
Two or more FlipFlops in series are appropriate for asynchronous signal capturing. More FlipFlops offer more safety and cause more delay.



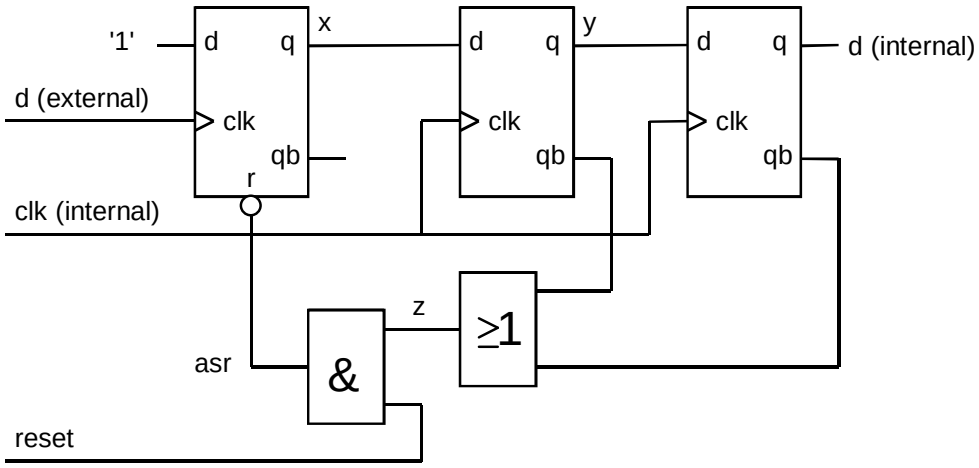


Figure 17: Asynchronous handshake circuit

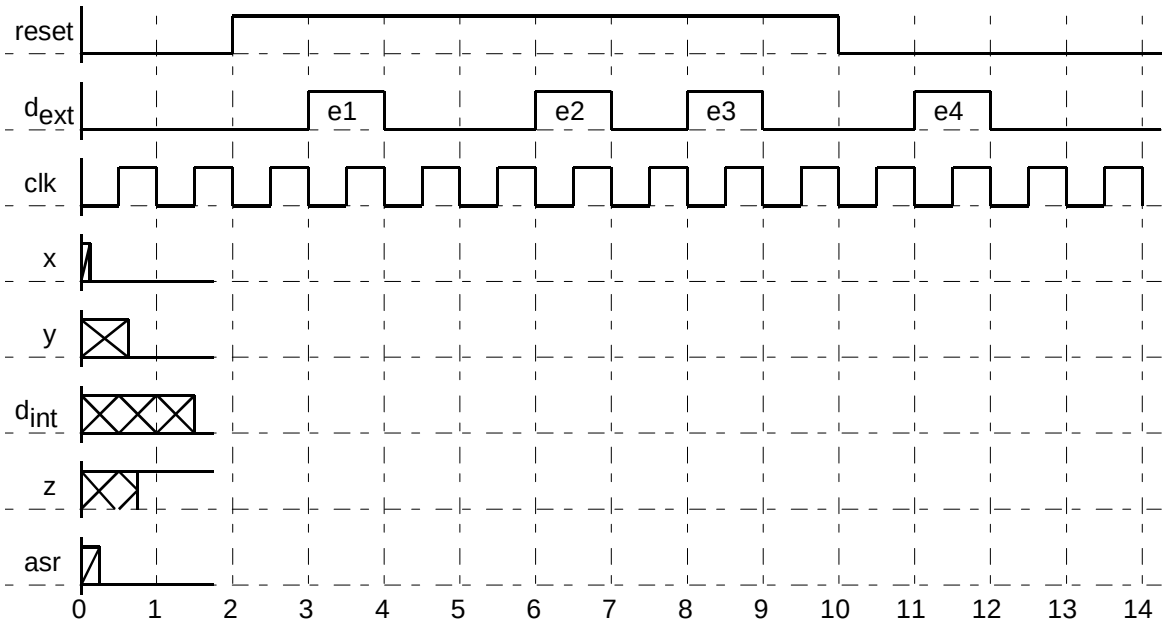


Figure 18: Signal versus time diagram of asynchronous handshake circuit.

Exercise:

a) What is the most simple and even good method to capture asynchronous events?
b) What is metastability?
c) Can metastability propagate through several FlipFlops?
d) Does the probability of metastability increase with increasing clock speed?
e) Complete the signal versus time diagram of Fig. 18 to illustrate the operation of the circuit shown in Fig. 17.

5 Delay Lines and Monostables

Delay lines are used sometimes to build multivibrators, monostables or to avoid races. However, the practice of delay-line dependent circuits is not recommended, as the actual timing of the delay line is difficult to predict, and is highly sensitive to temperature and process spread.

Non-recommended Circuits

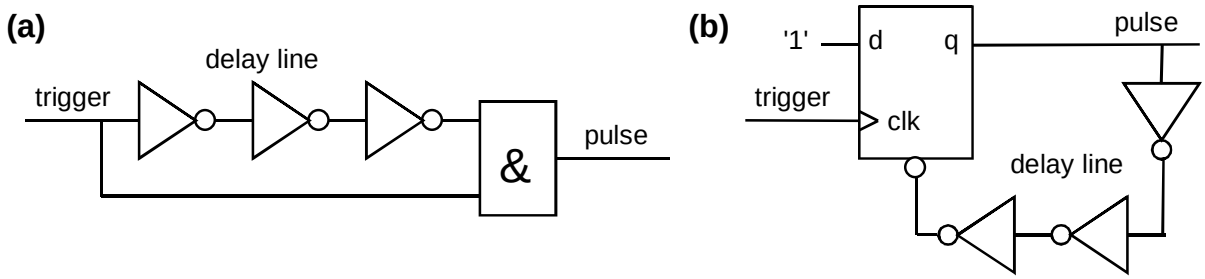
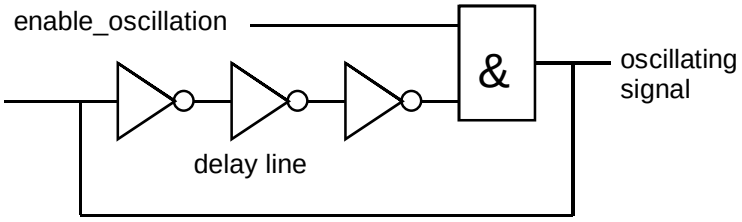


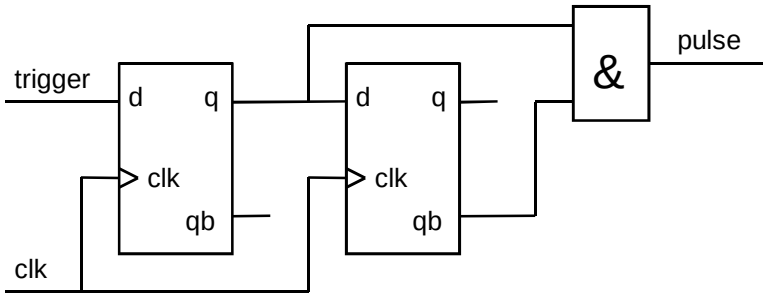
Figure 19: Pulse generators (monostables) using delay lines.

Figure 20: Multivibrator using delay line.



Recommended Circuit

Figure 21: Synchronous pulse generator.



Exercise:

a) Is the delay of delay lines a reliable quantity?

b) What is the minimum length of a synchronous pulse?

6 Bistable Elements

Data storage elements should not be created by cross-coupling NAND or NOR gates to form bistable elements. There are a number of problems associated with bistable elements of this nature, including asynchronous operation, unknown output states for certain input combinations, sensitivity to input spikes and the lack of timing constraint checking in simulation.

Non-recommended Circuits

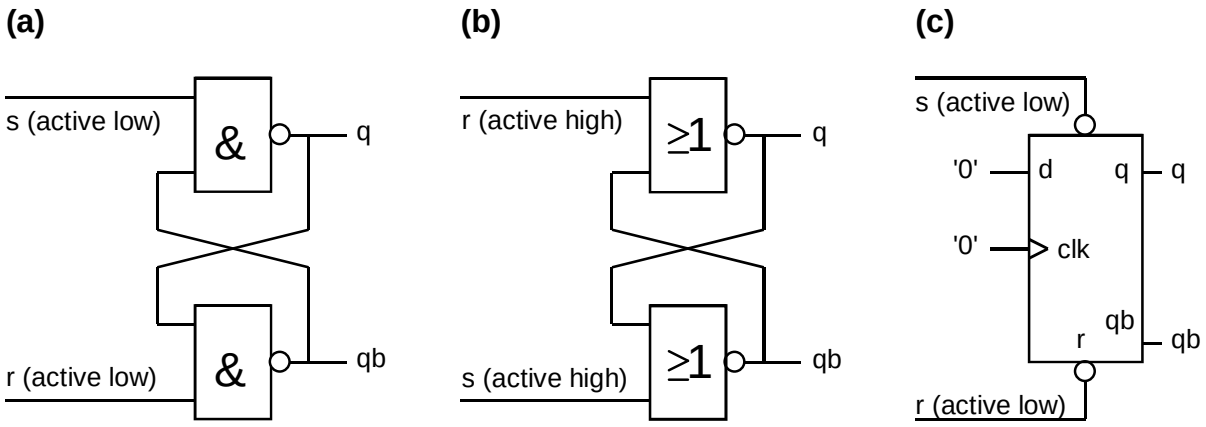


Figure 22: Non-recommended data storage elements: **(a)** Cross-coupled NAND gates, **(b)** cross-coupled NOR gates, **(c)** asynchronous RS FlipFlop.

Exercise:

a) What is a key problem of cross-coupled `NAND` or `NOR` gates as bistable elements?

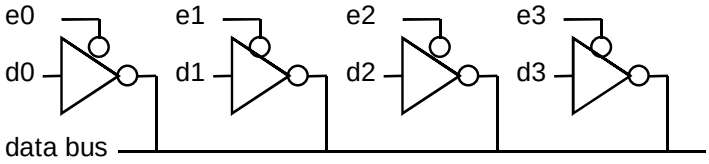
b) Should we rely on races, i.e. that one result is evaluated a little bit faster than an other?

7 Internal TriStates

Internal tristates for data bus access within a circuit must be used with care. Potential problems (particularly at initialization time) are an undriven bus and conflicting bus drivers. An undriven bus floats to an intermediate state, causing high static currents in subsequent gates.

Non-recommended Circuits

Figure 24:
Locally controlled tristate enables.



Recommended Circuits

- Either use a multiplexer instead of tristate buffers.
- Or use a central control decoder to control the enable entries of the tristate drivers and use additional securities like an hzpull FlipFlop and a tristate driver which is activated when all other drivers are high impedant.

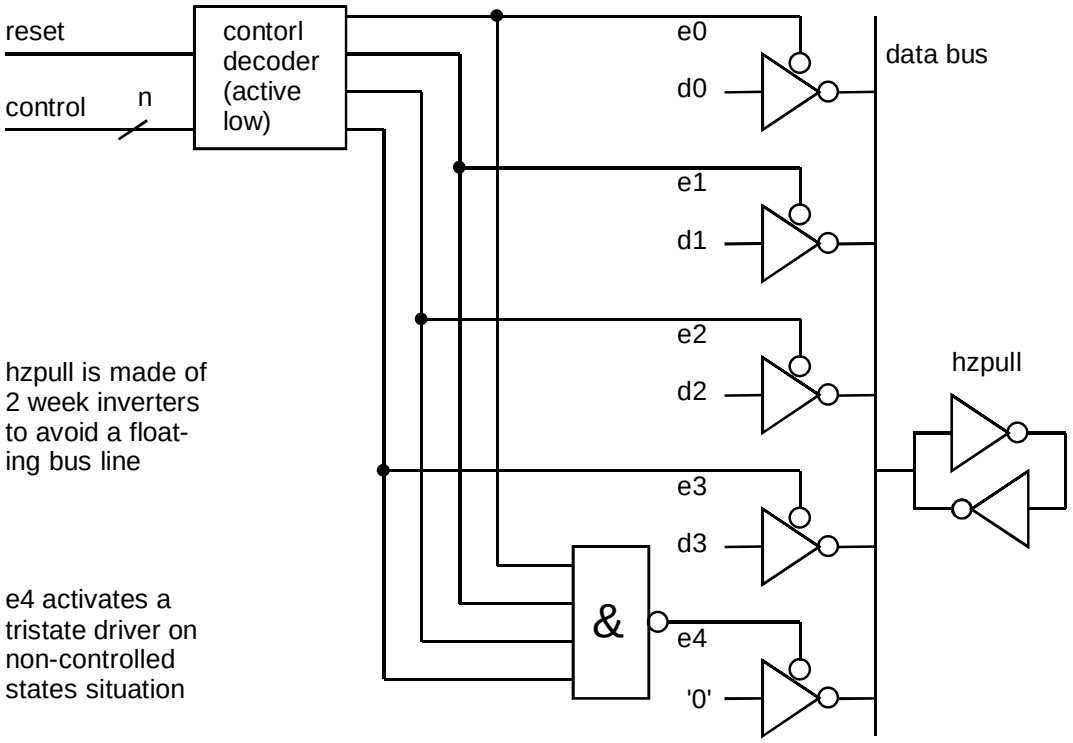


Figure 25: Recommend data bus control with central control decoder and double security: (1) An hzpull FlipFlop made of two minimum inverters prevents the bus from floating, (2) an additional tristate driver drives the bus to a defined state when all other tristates are high impedant.

Exercise:

a) What is the sense of a hzpull FlipFlop attached to a bus line?
b) Why is an undriven bus dangerous?
c) The functionality of an internal bus driven by tristate drivers can be obtained also by ... ?

8 Hardware Design Aspects

8.1 KISS:

Keep It Save and Simple!

- Build small modules with exactly defined and tested functionality like counter, adder, multiplier. (no “spaghetti code”!)
- Avoid mutual dependencies if possible, e.g. several linked state-machines that all fails when one fails.

8.2 Save connectivity rather than logic.

The typical limitation order is: pins, wiring, logic:

- Often, you would have more logic available but have no more pins on your device (e.g. ASIC or FPGA) to connect it.
- Building huge amounts of logic is easier than connecting it. Often it is advantageous from realization to build e.g. 2 or 3 identical counters counting the same values rather than one counter connected through the entire design.

8.3 Design for Reusability

- Well defined, tested and documented design blocks. Avoid jack-of-all-trades designs (German: eierlegende Wollmilchsau).
- 4-eyes principle: any code must be red and checked by a second person, and must be documented for that (particularly using comment).

8.4 Design for Testability

Design for testability is a wide field which is beyond the scope of this script. The reader is referred to the literature.

Consider some basic keywords concerning this topic:

- Avoid inaccessible internal logic.
- Use built-in self test (BIST).
- Use test vectors obtaining 100% fault coverage for single "stuck-at faults".
- Divide long counter chains into smaller counters.
- Consider scan path testing and JTAG interface.

Exercise:

a) What does "design for testability" mean?

b) Is it worth to respect "design for testability" aspects?

9 Summary: Rules of Synchronous Digital Circuit Design

9.1 Digital Hardware Design

1. Synchronous clocking! (do not gate clocks, no clock skew, no slow clock edge)
2. Supply global asynchronous reset!
3. No local asynchronous reset (-> local synchronous reset)
4. Use given clock signals (DE2-Board: CLOCK_50 and CLOCK_27)
5. Frequency control: use enable flip-flops
6. pulses: synchronous, width >= 1 clock period
7. Edge detection: generate legal pulses
8. Avoid Races
9. No loops in combinational logic (use given memory devices, no self made oscillators!)
10. Any wire / bus line driven exactly 1 x !
11. Build small modules with exactly defined functionality.
12. Save connectivity rather than logic.

9.2 Matlab Modeling

Coding a process synthesizing to memory:
(a) state = NextState;

Coding a process that does not synthesize memory (NextState-, Output-Logic)
(b) assign all output signals actively in any clock cycle

9.3 VHDL Modeling

Coding a process synthesizing to memory:
(a) signals reset and clock in sensitivity list only.

Coding a process that does not synthesize memory (NextState-, Output-Logic)
(b) All output signals actively driven under any condition
(c) All input signals in sensitivity list

10 References

[ES21994] ES2 ASIC Design Guidelines, European Silicon Structures, Munich.

[Wak1994] Wakerly, J. F.; Digital Design Principles and Practices, Prentice Hall, 1994, ISBN 0-13-059973-5

[Ska1997] Skahill, Kevin; "VHDL for Programmable Logic", Cypress Semiconductor, Addison Wesley 1997, ISBN 0-201-895586-2

[Kea1999] Keating, Michael; Bricaud, Pierre, "Reuse methodology manual for System-on-a-Chip Designs", Kluwer Academic Publishers, 1999, ISBN 0-7923-8175-0.

[Rus1995] Rushton, Andrew, "VHDL for Logic Synthesis, An Introduction Guide for Archieving Design Requirements", McGraw-Hill, 1995, ISBN 0-07-709902-6.

[Cha1999] Chang, Kou-Chuan, "Digital Systems Design with VHDL and Sysnthesis", IEEE Computer Society, 1999, ISBN 0-7695-0023-4, IEEE Comp. Soc. Order Number BP00023.

[Hei2000] Heinkel, U.; Padeffke, M.; Haas, W.; Buerner, T.; Braisz, H.; Gentner, T.; et al.., "The VHDL Reference", Wiley & Sons, England, 2000, ISBN 0-471-89972-0, Internet: www.vhdl-online.de.

[Arm2001] Armstron, James R.; Gray, F. Gail, "VHDL Design, Representation and Synthesis", Prentice Hall, 2001, ISBN 0-13-021670-4; + 1CD im Buch.

[Rei2002] Reichard, Jürgen; Schwarz, Bernd, "VHDL-Synthese", Oldenbourg Verlag, 2002, ISBN 3-486-25809-5.