ASIC Design Guidelines

Introduction

The Atmel ASIC Design Guidelines constitute a general set of recommendations intended for use by designers when preparing circuits for fabrication by Atmel. The guidelines are independent of any particular CAD tool or silicon process. They are applicable to Gate Arrays, Cell-Based ASICs (CBICs) and full-custom designs. Although they do not give specific coding recommendations, they apply equally to designs captured in Verilog or VHDL as to designs captured as schematics.

These guidelines do not cover general principles of ASIC design; rather they highlight specific design practices which are regarded as unsafe, and which can lead to devices which are difficult to test, and whose correct operation cannot be guaranteed under all circumstances. For each unsafe, and therefore non-recommended design practice, an alternative safe, and therefore recommended practice is proposed.

The current paradigm shift towards system level integration (SLI), incorporating multiple complex functional blocks and a variety of memories on a single circuit, gives rise to a new set of design requirements at integration level. These design guidelines do not fully address these issues yet. The recommendations are principally aimed at the design of the blocks and memory interfaces which are to be integrated into the system-on-chip. However, the guidelines given here are fully consistent with the requirements of system level integration. Respect for these guidelines will significantly ease the integration effort, and ensure that the individual blocks are easily reusable in other systems.

These design guidelines have been drawn up in the light of experience with large numbers of ASIC designs over more than a decade.

The Atmel ASIC Design Guidelines have a particular significance during the signoff of each design prior to submission for fabrication:

*Atmel customers must sign off a design to confirm that it complies with **all** the recommendations in the Atmel ASIC Design Guidelines. For each case of non-compliance, the case must be discussed with the ASIC Support Center, and if necessary a formal Authorization must be obtained.*

Application Specific IC (ASIC)

Application Note

Rev. 1205A–12/99

Synchronous Circuits

Experience has shown that the safest methodology for time-domain control of an ASIC is **synchronous design**.

A synchronous circuit is one in which:

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

Examples of circuit elements which contradict these principles are given below, and methods of achieving synchronous design are given in the four sections which follow.

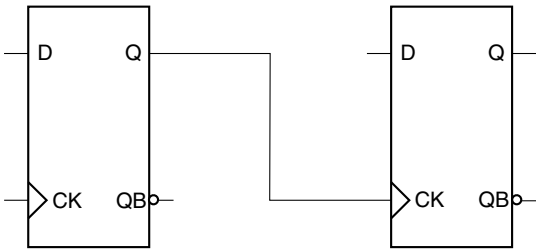
Non-recommended Circuits

Circuits which violate the principles of synchronous design include the following elements:

Flip-flop driving clock input of another flip-flop

The clock input of the second flip-flop is skewed by the clock-to-q delay of the first flip-flop, and is not activated on every clock edge. See Figure 1.

Figure 1. Flip-flop driving clock input of another flip-flop

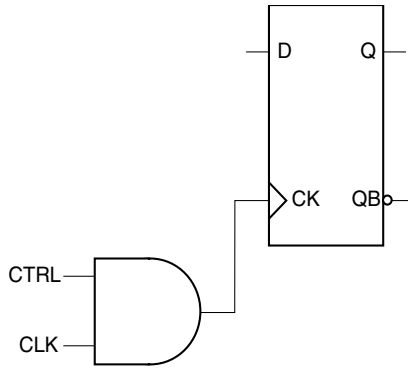


An example of a circuit containing this element is a ripple counter.

Gated clock line

Gating in a clock line (Figure 2) causes clock skew and can introduce spikes which trigger the flip-flop. This is particularly the case when there is a multiplexer in the clock line.

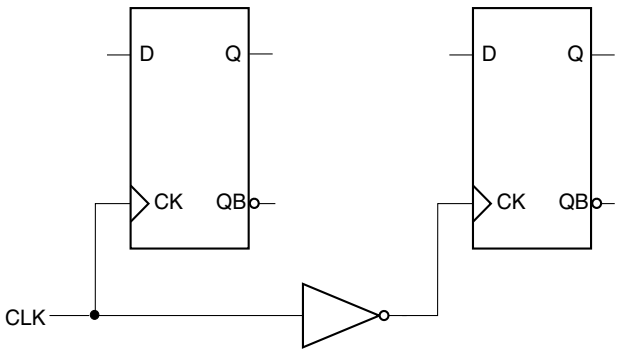
Figure 2. Gated clock line



Double-edged clocking

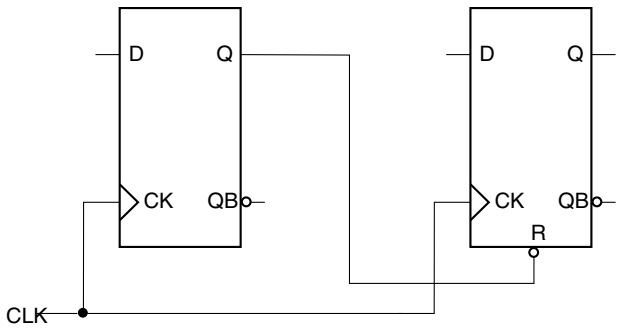
The two flip-flops are clocked on opposite edges of the clock signal (Figure 3). This makes synchronous resetting and test methodologies such as scan-path insertion difficult, and causes difficulties in determining critical signal paths.

Figure 3. Double-edged clocking



Flip-flop driving asynchronous reset of another flip-flop.
In Figure 4, the second flip-flop can change state at a time other than the active clock edge, violating the principle of synchronous design. In addition, this circuit contains a potential race condition between the clock and reset of the second flip-flop.

Figure 4. Flip-flop driving asynchronous reset of another flip-flop



An example of a circuit containing this element is an asynchronously reset counter.

Recommended Circuits

Methods of achieving the requirements of synchronous design, and avoiding the non-recommended situations described above are dealt with in subsequent sections, as follows:

- Synchronous clocking by means of clock buffering: See “Clock Buffering” on page 4.
- Flip-flop driving clock signal of another flip-flop: See “Gated Clocks” on page 10.
- Gated clocks: See “Gated Clocks” on page 10.
- Double-edged clocking: See “Double-edged Clocking” on page 11.
- System clock generation: See “Clock Generation and Overall Circuit Control” on page 12.
- Asynchronous resets: See “Asynchronous Resets” on page 13.

Clock Buffering

To achieve the requirement of a simultaneous application of a single clock signal at all storage elements in a design, and avoid problems due to fanout, a clock buffering scheme needs to be implemented consistently throughout a circuit. This is often done automatically as part of placement and routing; if not, the principles described in this section should be followed.

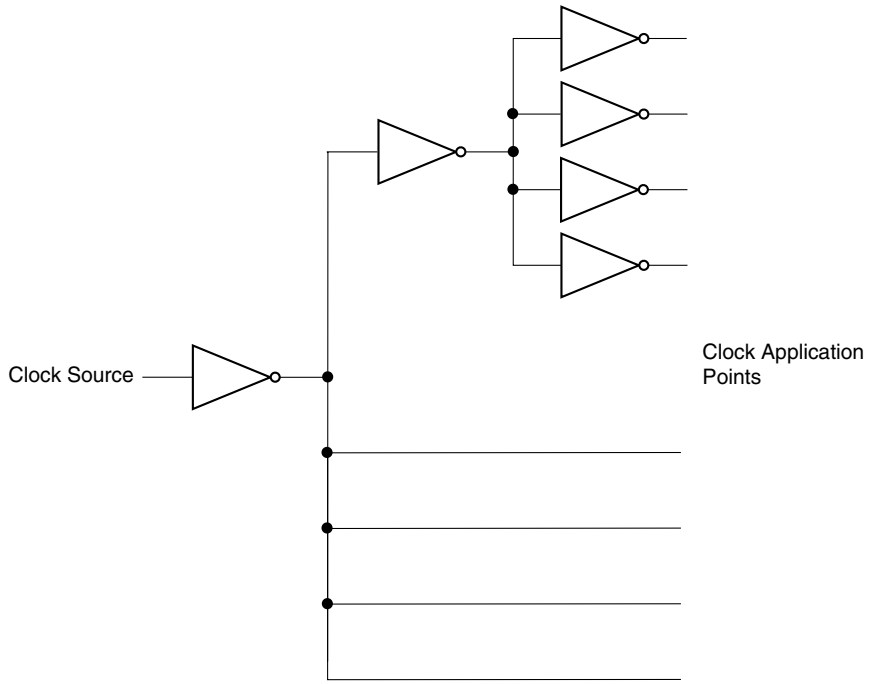
Non-recommended Circuits

Circuits which violate the principles of consistent clock buffering include the following elements:

Unequal depth of clock buffering

The depth of clock buffering differs between different clock application points, causing clock skew. See Figure 5.

Figure 5. Unequal depth of clock buffering



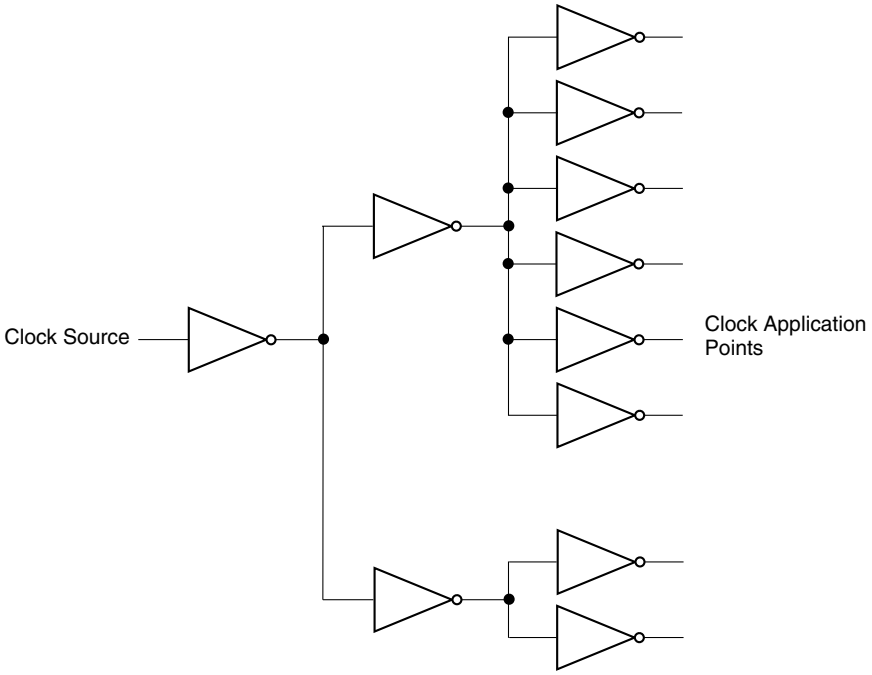
Unbalanced fanout on clock buffers

As shown in Figure 6, the difference between the fanouts at the two intermediate buffers gives rise to different load-dependent delays, causing clock skew.

Excessive clock fanout

Excessive clock fanout leads to slow clock edges, which can cause a number of problems, including an increased risk of metastability in flip-flops which capture external asynchronous signals.

Figure 6. Unbalanced fanout on clock buffers



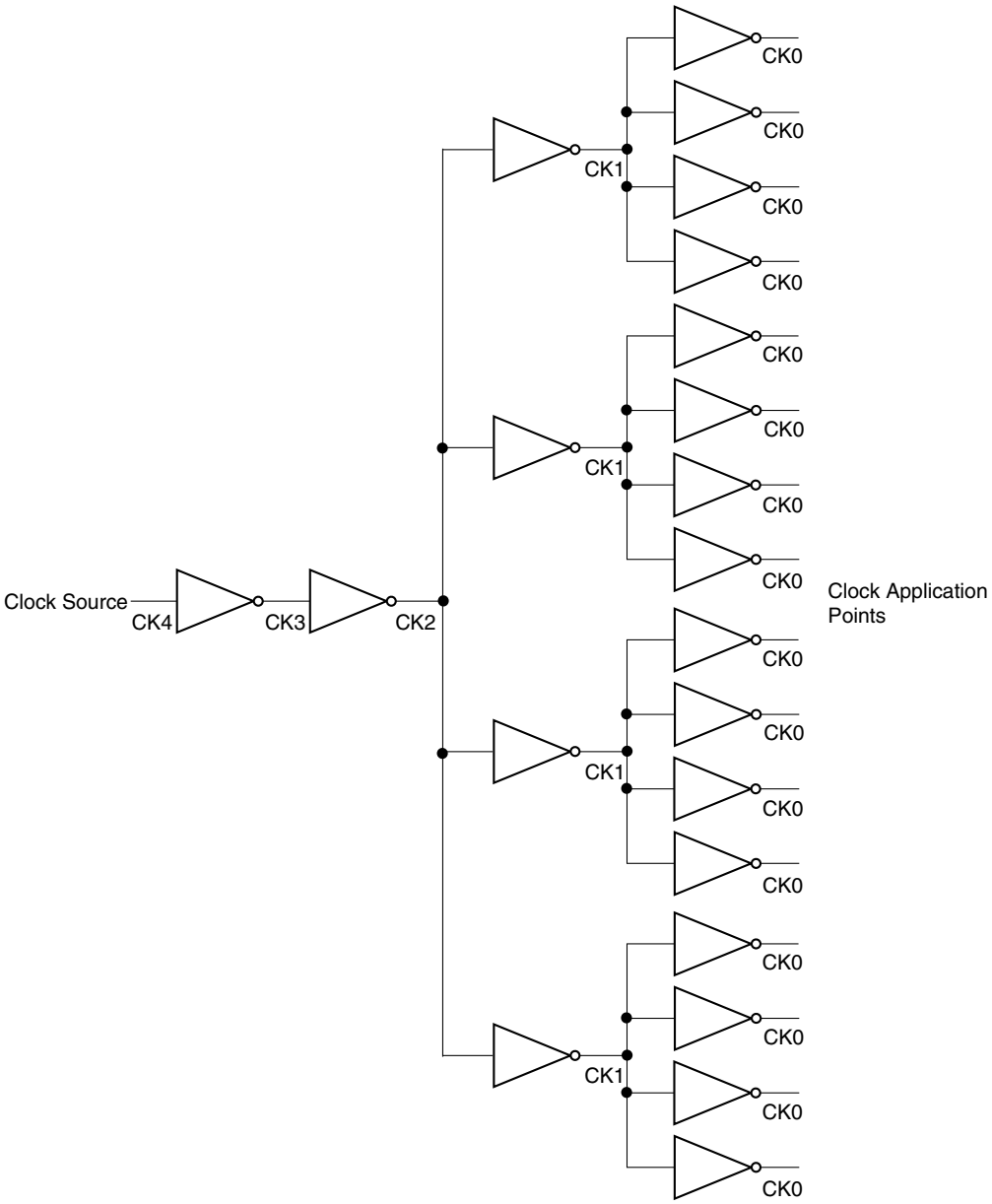
Recommended Circuits

The recommended clock buffering scheme is balanced tree buffering, which must satisfy the following conditions:

1. The same depth of buffering to all clocked cells. (A suggestion is to use the naming convention: ck0 at application point, then ck1, ck2, ... ckn, and join equivalent levels up the circuit hierarchy. Note that n must be even to retain clock polarity.) See Figure 7.
2. The same fanout on all buffers. This must be checked after placement and routing, to ensure that tracking capacitances do not unbalance the fanout.
3. Lightly loaded buffers to keep clock edges sharp (max 50% of max relative fanout). An alternative is to use a combination of geometric and tree buffering, as illustrated in Figure 8.

Balanced clock tree buffering

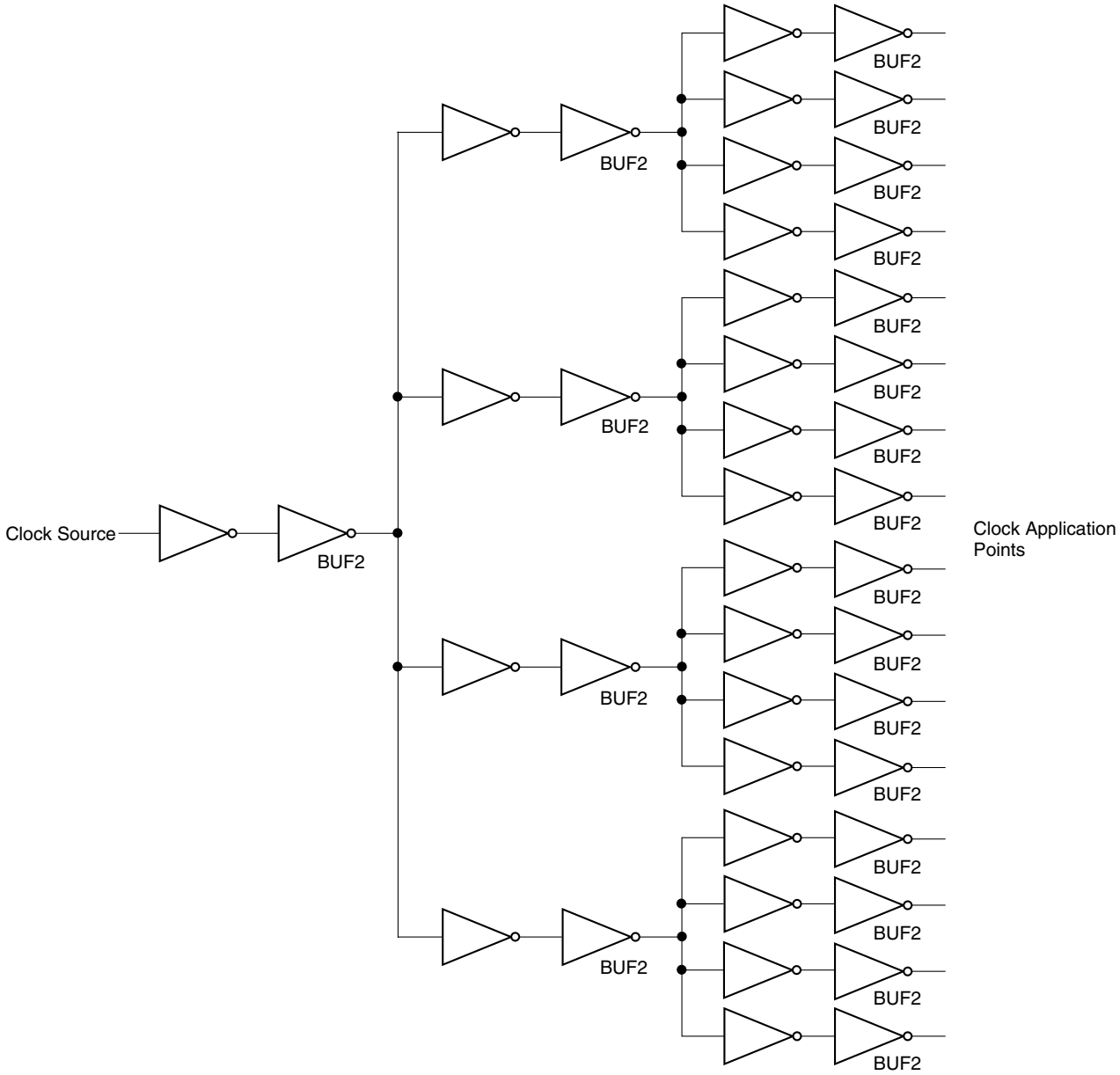
Figure 7. Balanced clock tree buffering



Combined geometric/tree buffering

By using an intermediate buffer of a suitable drive strength at each clock fanout point, the relative fanout at each buffer is reduced, and clock edges remain sharp.

Figure 8. Combined geometric/tree buffering



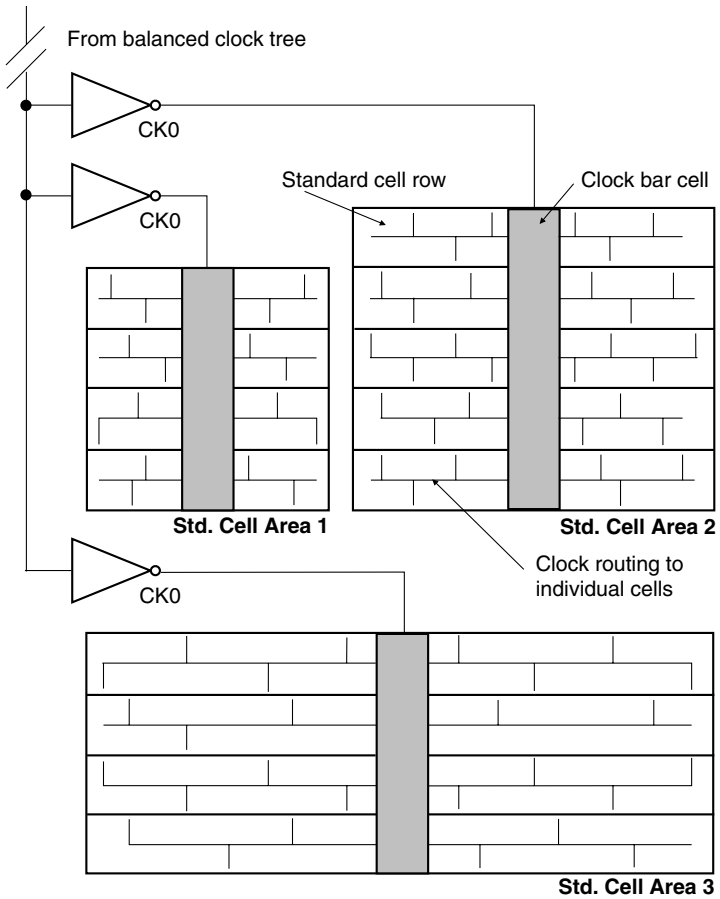
Clock Bar Cells

The use of clock bar cells for clock distribution from within a standard cell area is recommended (during placement and routing, if they are available), as shown in Figure 9. A single Clock bar cell, positioned correctly in the centre of the standard cell area, can provide a balanced clock net distribution. This runs a vertical clock trunk through the middle of the cell area, allowing clock net branches to feed cells on either side of the trunk. This method reduces the risk of clock skew by halving the effective clock path length along a row of cells, compared with a clock supplied from one end of the cell row. It also guides the router to prevent a long clock path being threaded through the standard cells, and prevents clock net looping.

It is recommended to use only one clock bar cell per standard cell area (otherwise clock looping may occur). By using clock bar cells, there will be a balanced clock net distribution within each standard cell area.

Balanced clock routing using clock bar cells

Figure 9. Balanced clock routing using clock bar cells

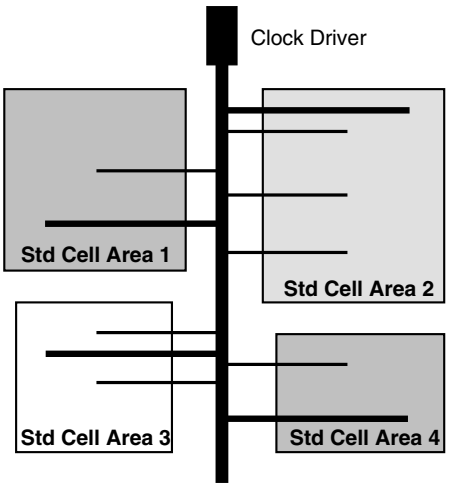


Clock Guidance

The use of clock guidance is recommended if available, before starting place and route. A central clock trunk should be run between the standard cell areas with branches feeding off either side into the standard cell areas themselves, as shown in Figure 10. A bad example of Clock guidance is given in Figure 11, highlighting the risk of clock skew.

Good example of clock guidance

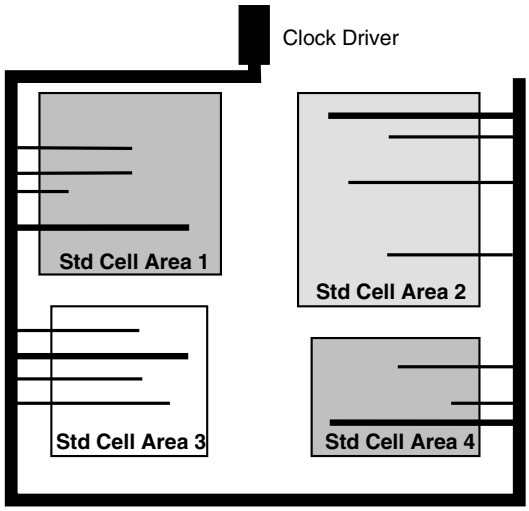
Figure 10. Good clock guidance for routing



It is important to have an even number of rows (in the standard cell areas), because an odd number of rows can force the place and route software to create loops on the clock net.

Bad example of clock guidance

Figure 11. Bad clock guidance for routing



Clock Compilers

If Clock compilers are available, they help to maintain a a balanced clock network, but should be used with care. The clock compiler automatically adjusts the clock buffering to make the equivalent delays for each cell area the same as the longest delay. This means additional buffer cells may be added both outside and inside the standard cell areas, and the cell areas themselves may be split.

Gated Clocks

A seemingly obvious way of controlling the operation of a flip-flop is to gate the clock signal with a control signal, or to multiplex two alternative clocks into its clock input. *This practice is dangerous on two counts:*

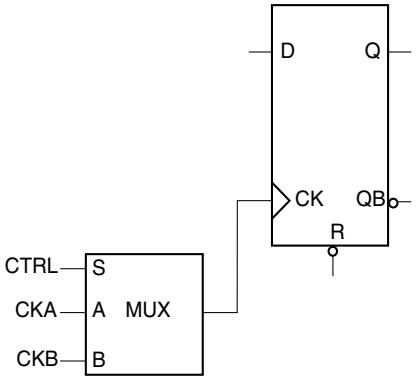
- A glitch on the gate output can cause a clock edge.
- Gating in the clock line introduces clock skew.

Non-recommended Circuits

A **particularly unsafe** circuit element is shown in Figure 12.

Multiplexer on clock line

Figure 12. Multiplexer on clock line



Toggling the multiplexer control signal inevitably causes a glitch on the ck input to the flip-flop, which may cause it to capture invalid data.

Recommended Circuits

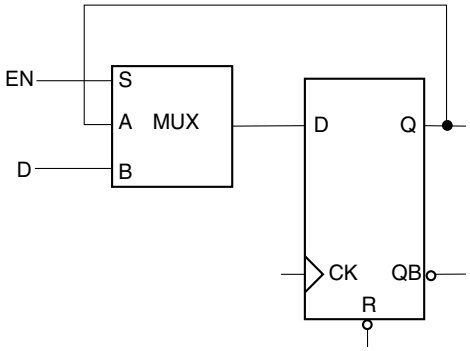
Two circuit elements which are recommended for use in synchronous designs are illustrated here. They are the enabled (E-type) flip-flop and the toggle (T-type) flip-flop. They remove the need for gated clocks, or for using the output from one flip-flop as the input to another.

Enabled (E-type) flip-flop

The enable signal (the multiplexer select line) controls the input of data to the flip-flop. If enable is low, the existing value of q is re-input at the next clock cycle. If enable is high, a new data value is clocked in. See Figure 13.

Note: A version of the E-type flip-flop can be constructed with a synchronous reset. A recommended way of constructing an E-type flip-flop is using AOI logic. See “Design for Speed” on page 31.

Figure 13. E-type flip-flop

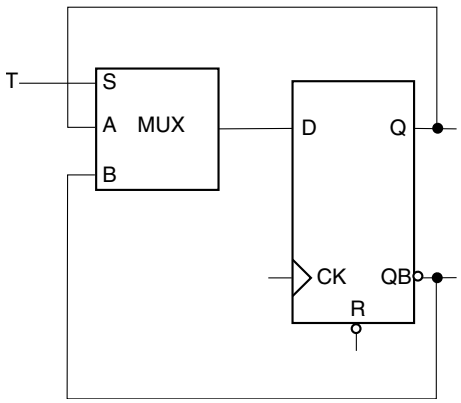


Toggle (T-type) flip-flop

The toggle flip-flop is the basic element in synchronous counters. The toggle signal (the multiplexer select line) controls state of the flip-flop. If toggle is low, the flip-flop retains its existing value at the next clock edge; if toggle is high, it takes the opposite value. See Figure 14.

Note: A version of the T-type flip-flop can be constructed with a synchronous reset. A recommended way of constructing a T-type flip-flop is using AOI logic. See “Design for Speed” on page 31.

Figure 14. T-type flip-flop



Double-edged Clocking

In an attempt to increase data throughput rates, use is sometimes made of both the rising and the falling clock edge for clocked elements. This practice, however, violates the principles of synchronous design given in “Synchronous Circuits” on page 2, and causes a number of problems, in particular:

- An asymmetrical clock duty cycle can cause setup and hold violations.
- It is difficult to determine critical signal paths.
- Test methodologies such as scan-path insertion are difficult, as they rely on all flip-flops being activated on the same clock edge. If scan insertion is required in a circuit with double-edged clocking, multiplexers must be inserted in the clock lines to change to single-edged clocking in test mode. *See, however, the warning in “Multiplexer on clock line” on page 10.*

The recommended alternative is to use a single-edged clocking scheme with a higher clock frequency.

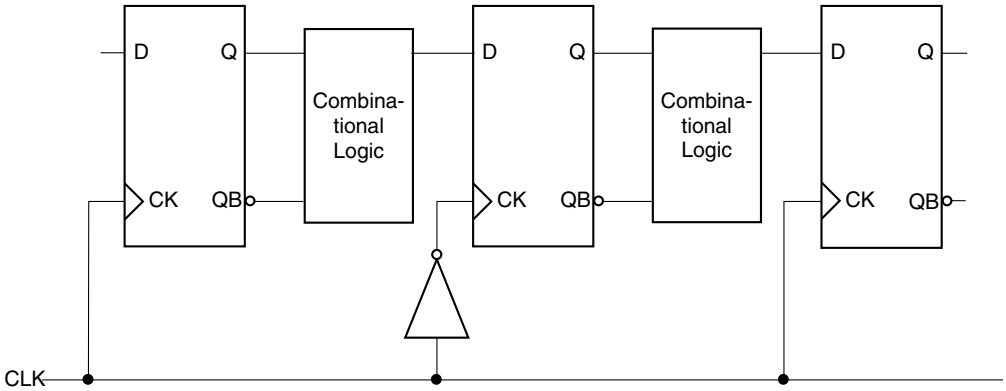
A general principle of synchronous circuit design is that the minimum time resolution available within the circuit is the duration of one complete clock cycle.

Non-recommended Circuit

Pipelined logic with double-edged clocking

In a circuit as shown in Figure 15, an asymmetrical clock duty cycle could cause setup and hold time violations, and a scan-path cannot easily be threaded through the flip-flops.

Figure 15. Pipelined logic with double-edged clocking



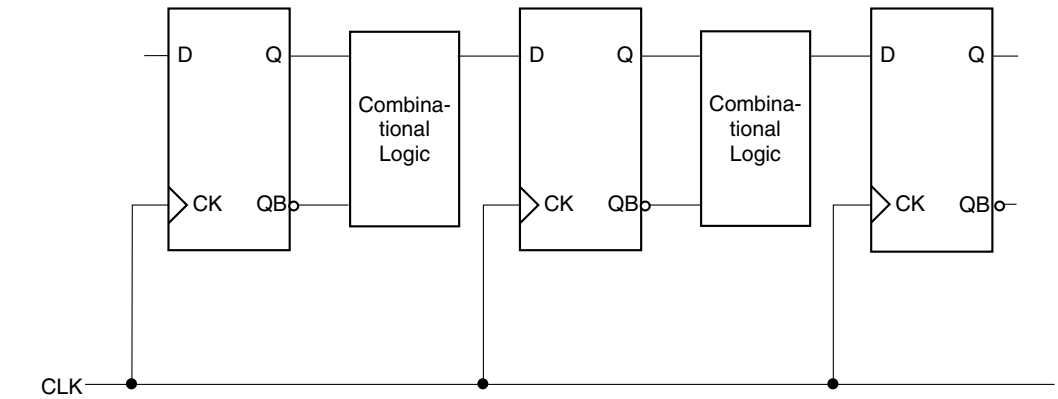
Recommended Circuit

Pipelined logic with single-edged clocking

The equivalent synchronous circuit (Figure 16) requires a clock frequency of double the previous version.

It is also recommended that enabled logic is used where required. See “Gated Clocks” on page 10.

Figure 16. Pipelined logic with single-edged clocking



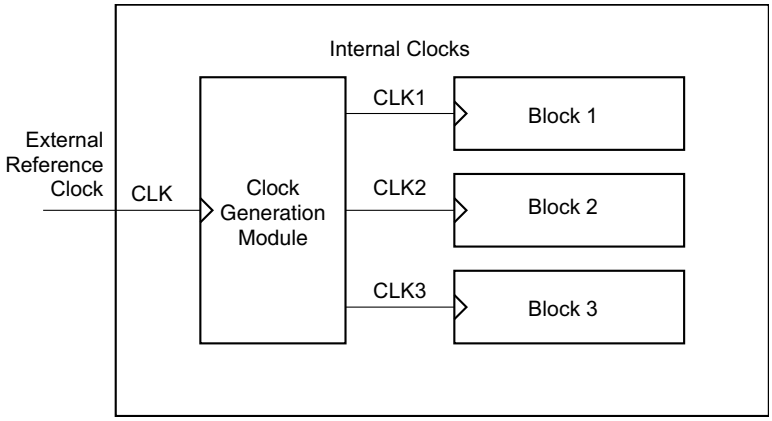
Clock Generation and Overall Circuit Control

If clocks of different speeds are required by different blocks, or the internal clock is required at a speed faster or slower than the externally available clock, it is recommended that a single clock generation block is constructed at the top level of a circuit. This produces the internal clocks required by all the functional blocks in the circuit. See Figure 17.

Communication between the internal blocks is achieved by the same principles as for asynchronous external inputs. See “Asynchronous Inputs” on page 17.

Recommended Circuit

Figure 17. Clock generation module at circuit top level



Generating higher- or lower-speed internal clocks

If the externally available clock signal is of a higher frequency than that required for an internal clock, a synchronous binary counter (made from T-type flip-flops) is recommended to perform the required clock division. Latching of data conditionally, or at a lower frequency than this internal clock is achieved by the use of individual E-type flip-flops for data storage.

Alternatively, a PLL can be used to produce a higher-speed internal clock than the external reference clock.

Asynchronous Resets

The general recommendations for dealing with resets within an ASIC are as follows:

1. The circuit must be brought to a known state, both within test and in operation, within a stated and agreed number of clock cycles. The known state is generally achieved by means of a reset mechanism.
2. If an asynchronous reset is required, use a *single global asynchronous reset* driven by an external input. A tree buffering scheme similar to that for clock distribution may be required to ensure a sharp edge on the reset signal. The benefit of a reset of this nature is that it places the entire circuit in a known state in response to a change on a single input signal, with no clock cycles required for the known state to propagate.
3. If a power-on reset (POR) pad is used, the circuit must contain another global reset for test purposes.
4. If a local reset is required, use a synchronous reset.

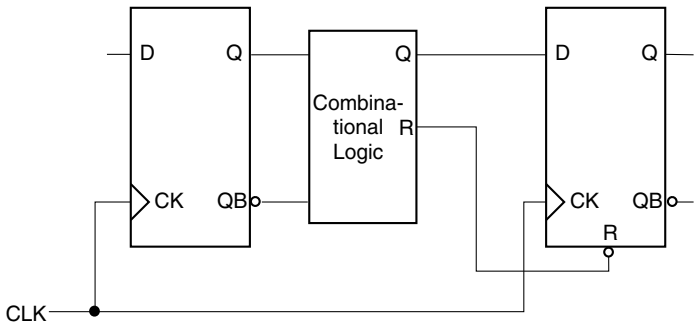
Non-recommended Circuit

A local asynchronous reset such as on a counter causes a change of state in a storage element which is not triggered by the active clock edge, and therefore violates the principles of synchronous design given in “Synchronous Circuits” on page 2.

Local asynchronous reset of a flip-flop

In Figure 18, the local asynchronous reset causes a change of state on the second flip-flop which is not synchronized with the active clock edge.

Figure 18. Flip-flop driving asynchronous reset of another flip-flop



Recommended Circuits

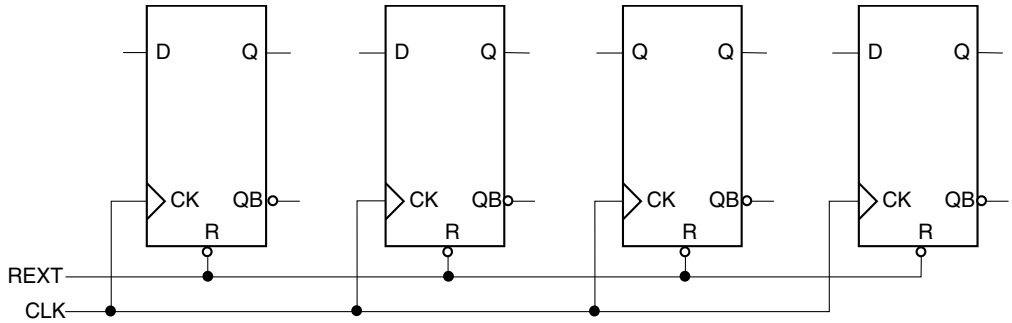
The circuits given below overcome the problems discussed in the previous section.

A general recommendation is, if necessary, to organize resets into a hierarchy, from global (which may be asynchronous) to local (which must be synchronous).

Global asynchronous reset of all flip-flops

In Figure 19, a single external reset signal (rext) is connected to all flip-flops. The buffering which may be required is not shown.

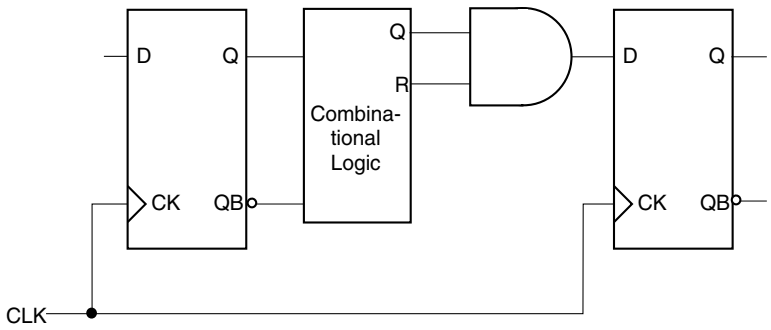
Figure 19. Global asynchronous reset of all flip-flops



Local synchronous reset of a flip-flop

In Figure 20, the (active low) reset signal (r) is gated with the d-input of the second flip-flop, making it synchronous. The second flip-flop changes state only on an active clock edge.

Figure 20. Flip-flop driving a synchronous reset of another flip-flop



Shift Registers

Shift registers are particularly intolerant of clock skew. A problem which occurs in their design is that long shift registers may require internal clock buffering. If not properly designed, this buffering can cause clock skew within the shift register, and interfacing problems between the shift register and the rest of the circuit.

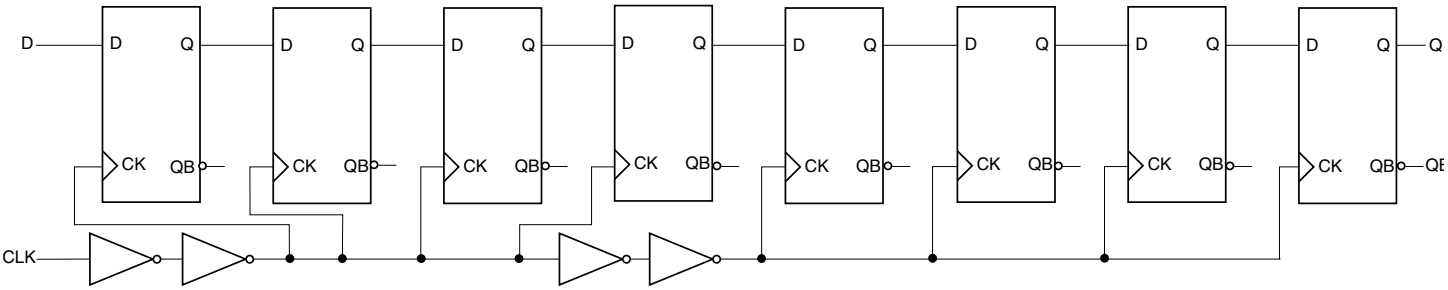
Non-recommended Circuits

Not recommended is a chain of clock buffers within shift register, in either the forward or the reverse direction. These cases are illustrated below.

Shift register with forward chain of clock buffers

The problem with a forward chain of clock buffers (Figure 21) is that internal clock skew can cause data fallthrough (where one stage of the shift register is skipped).

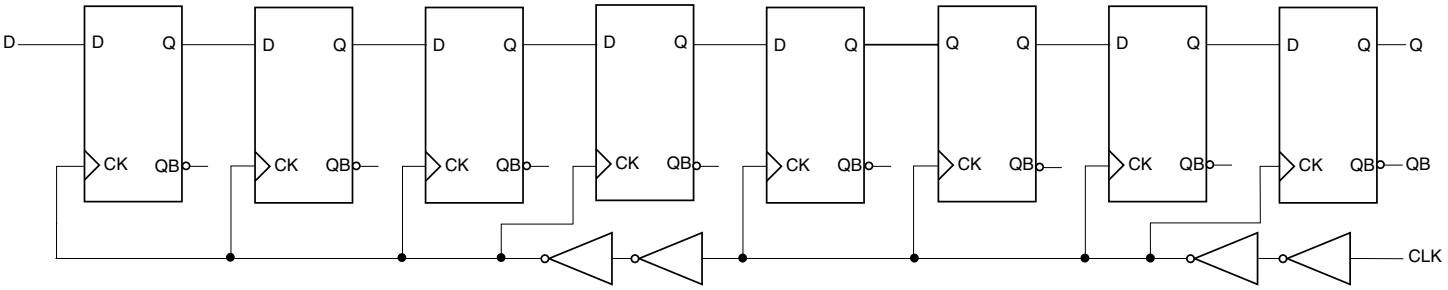
Figure 21. Shift register with forward chain of clock buffers.



Shift register with reverse chain of clock buffers

As shown in Figure 22 below, the problem with a reverse chain of clock buffers is the timing interface between the first D-type and the input data received from the rest of the circuit.

Figure 22. Shift register with reverse chain of clock buffers.

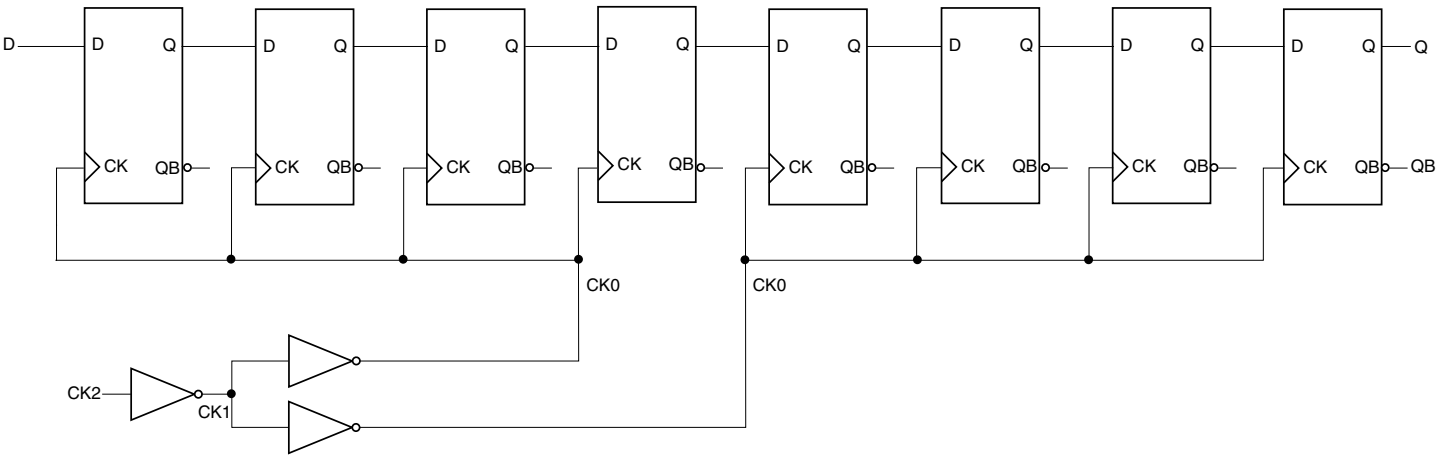


Recommended Circuits

There are two recommended ways of constructing the clock buffering scheme within a shift register:

1. Use balanced clock tree buffering as in the rest of the circuit. See “Clock Buffering” on page 4 and Figure 23 below. As an additional safety feature, buffering can be introduced in the data lines between each flip-flop.
2. Use a FIFO.

Figure 23. Shift register with balanced tree of clock buffers.



Shift register with balanced clock tree buffering

As shown in Figure 23, the clock tree within the shift register must be balanced (in terms of relative fanout) with the same levels of clock tree in other parts of the circuit. Note the naming convention for clock signals which facilitates this.

Asynchronous Inputs

A problem arises at the interface between a synchronous circuit and an external asynchronous input. At the flip-flop which captures the asynchronous input, there is a probability of **metastability** occurring. This section suggests some circuits which capture an external asynchronous input with a minimal risk of metastability.

Note: For large designs, inter-block communication is similar to external asynchronous interfacing.

Non-recommended Circuits

Not recommended is any circuit using a complicated feedback loop to capture an asynchronous input. The function of such circuits is obscure, and they run the risk of creating more problems than they solve. They are also very sensitive to noise, and their function can be altered by placement and routing delays.

Recommended Circuits

There are two recommended approaches to the problem of capturing an asynchronous input signal:

1. Two (or more) D-type registers in series to reduce the probability of metastability (Figure 24).
2. Use an asynchronous handshake circuit (Figure 25).

In all cases, the asynchronous event is a rising edge on the d (external) input to the first flip-flop. The pulse width of this signal is indeterminate, but is at least one clock cycle. The asynchronous event may occur simultaneously with a rising clock edge.

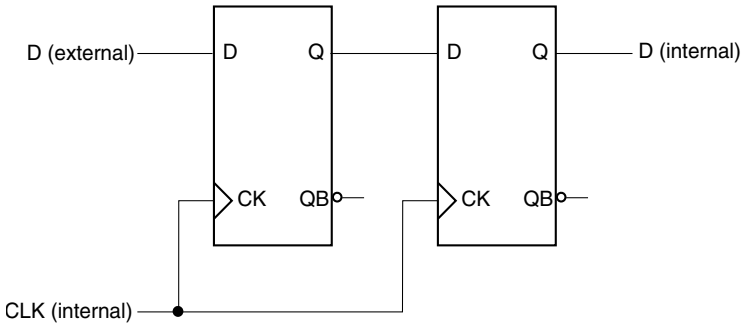
A general point which applies to all situations where metastability is possible is as follows:

- The rise and fall times of both the clock and data signals are significant: *fast edges reduce the probability of metastability.*

Two D-type flip-flops in series to capture an asynchronous input

If the first flip-flop goes into a metastable state, the probability that it will still be in that state at the next rising clock edge is low. Should this, however, occur, the metastable state is propagated to the d (internal) output and into the rest of the circuit. The probability of this situation is reduced by additional flip-flops in series.

Figure 24. Two D-type flip-flops in series to capture an asynchronous input



The common characteristics of circuits of this nature are as follows:

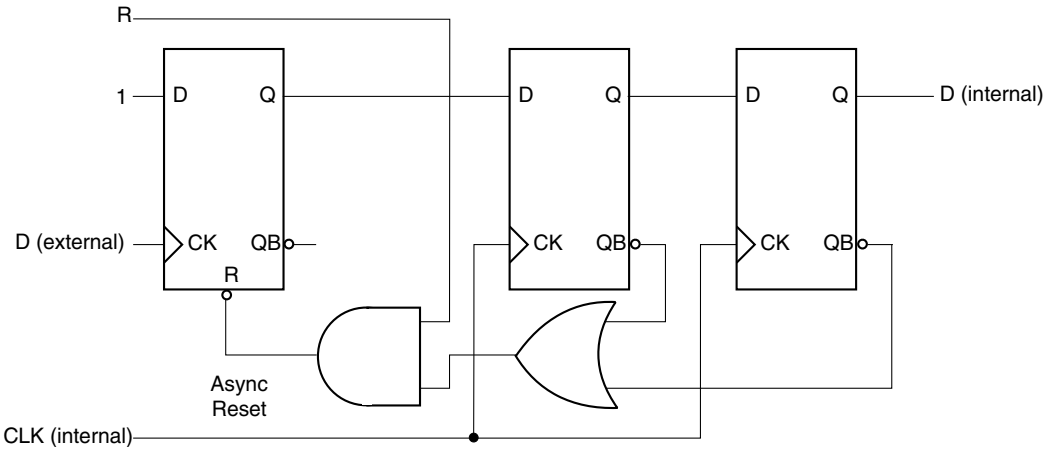
- In order for the d (external) rising edge to cause a rising edge on the d (internal) output, there must be at least one clock cycle between asynchronous inputs during which d (external) is low. This reduces the maximum frequency for the recognition of external events to half that of the internal clock frequency.
- If the flip-flop which receives the asynchronous d (external) rising edge settles (after a period of metastability) into the state with q = 0, the external input is lost unless it persists beyond the next rising clock edge.
- Metastability can be caused by a rising or a falling edge on the d (external) input.

Asynchronous handshake circuit

A circuit of the type shown in Figure 25 can be used to detect an asynchronous event: a rising edge on d (external). *These events must occur at longer time intervals than two clock cycles.*

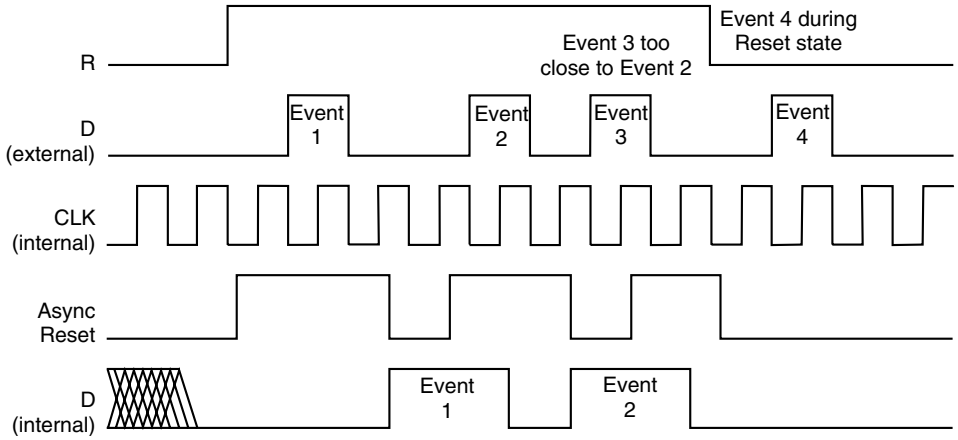
The external event (d) drives the clock input of the first flip-flop. This is the **only** flip-flop in the circuit which has a clock input not driven by the system clock (clk). The d-input to this flip-flop is tied to logic 1. It has an asynchronous input driven from the system reset (r) and from the qb outputs of the second and third flip-flops.

Figure 25. Asynchronous handshake circuit



In reset mode (r = 0), the first flip-flop is reset asynchronously. This state takes two clock cycles to propagate to the d (internal) signal. In active mode (r = 1), a rising edge on d (external) immediately drives the q-output from the first flip-flop high. After one rising clock edge, this propagates to the q-output from the second flip-flop, and after a second clock edge, to the d (internal) output from the third flip-flop. At this time, the qb outputs from the second and third flip-flops are both low. This logic level propagates through the OR and the AND gates in the feedback loop, forcing a reset on the first flip-flop, which is now ready to receive another rising edge on the d (external) input. The circuit function is illustrated in Figure 26.

Figure 26. Operation of asynchronous handshake circuit



The d (internal) signal can be used as an acknowledge signal to the external system which is supplying the d (external) inputs.

The risk of metastability is at the second flip-flop: caused by simultaneous rising edges on the (asynchronous) q-output from the first flip-flop and the system clock. If this occurs, there are three possibilities:

- The second flip-flop settles into a q = 1 state before the next rising clock edge. This is then clocked by the third flip-flop, and the circuit functions normally.

- The second flip-flop settles into a q = 0 state before the next rising clock edge. This causes no change to the third flip-flop, and the feedback loop to the first flip-flop is unaffected. Therefore the first flip-flop retains its q = 1 value to be clocked by the second flip-flop on the next rising clock edge. The effect of this is to delay the recognition of the asynchronous event by one clock cycle.
- The metastable state persists until the next rising clock edge. In this case there is a possibility of the third flip-flop entering a metastable state as well. However, the probability of a metastable state persisting for an entire clock cycle, and forcing the third flip-flop into a similar state, is extremely low. This risk can be further reduced by inserting additional flip-flops, at the expense of an additional clock cycle as the minimum delay between recognized inputs.

Note: Metastability can only be caused by a rising edge of the d (external) input, whereas in the previous two circuits it can be caused by either edge. The only restriction on pulse width for the asynchronous handshake circuit is the minimum pulse width of the first flip-flop.

This circuit will enter an unknown state if it receives simultaneous rising edges on the d (external) and reset (r) signals.

Delay Lines and Monostables

There is often an apparent requirement to create a short pulse within a circuit, of duration less than a clock cycle. This generally requires the use of a **delay line** within a monostable element, as shown in Figure 29 below. A multivibrator circuit (Figure 31) is based on a similar principle. More generally, asynchronous circuits often rely on delay lines for their correct operation, for example in an attempt to overcome race conditions.

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.

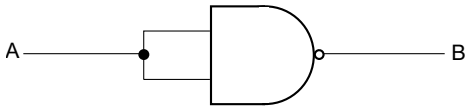
In particular, due to simulation model constraints it is not permitted to short two inputs of a logic gate to the same source signal (Figure 27). The problem is that the gate delays are characterized with one signal changing. For a NAND3 driven to a one (Figure 28), if two signals change simultaneously there are two transistors pulling the output high, instead of one. This will reduce the delay time by about 50% compared to the simulation model.

Non-recommended Circuits

In general, any circuit which relies on delays for its operation is not recommended. All gates in series which are not used for buffering must be considered as delay lines. Five specific examples are given below:

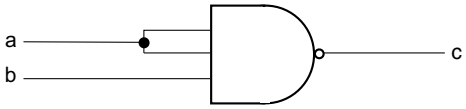
NAND2 gate used as delay element

Figure 27. NAND2 gate used as a delay element



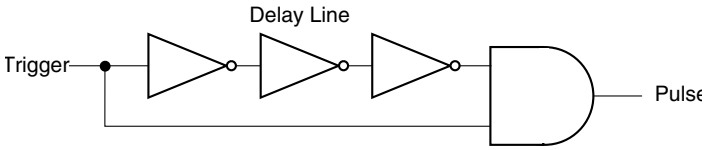
NAND3 gate with two inputs connected together

Figure 28. NAND3 gate with two inputs connected together



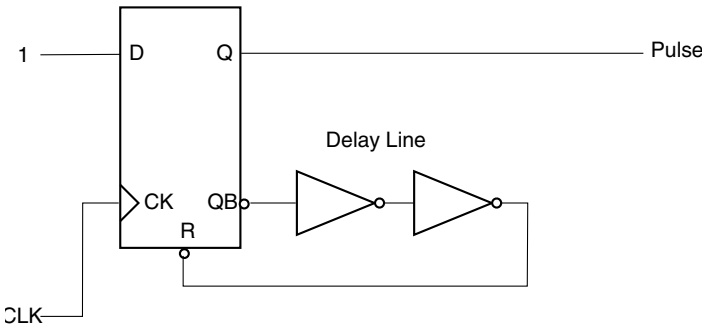
Monostable pulse generator

Figure 29. Monostable pulse generator



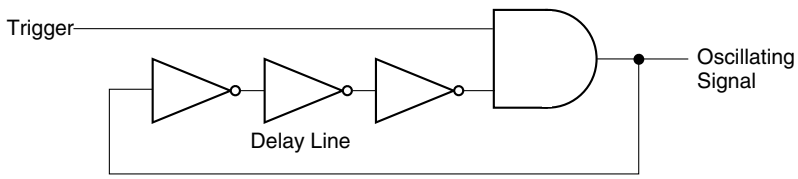
Pulse generator using a flip-flop

Figure 30. Pulse generator using a flip-flop



Multivibrator

Figure 31. Multivibrator



Care must be taken not to create inadvertently an equivalent circuit to this one, for example, in the (synchronous) reset loop of a counter.

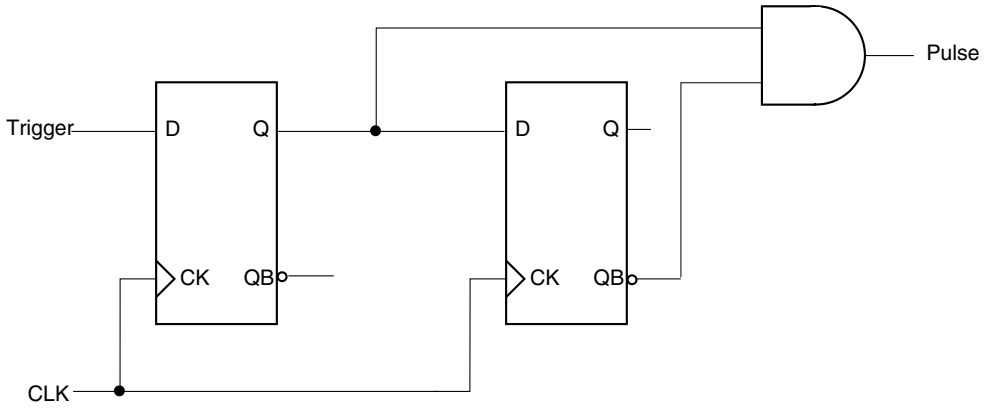
Recommended Circuit

If at all possible, delay-line dependent circuits should be avoided completely. The safe solution to the problem is as follows:

1. Use a higher clock speed. *The best time resolution available in a circuit is the width of one clock cycle.*
2. Use a synchronous pulse generator, as illustrated in Figure 32 below.

Synchronous pulse generator

Figure 32. Synchronous pulse generator



Authorization

Delay-dependent circuitry is only accepted by Atmel when it is accompanied by post-layout (H)Spice simulation results of the relevant circuit elements.

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.

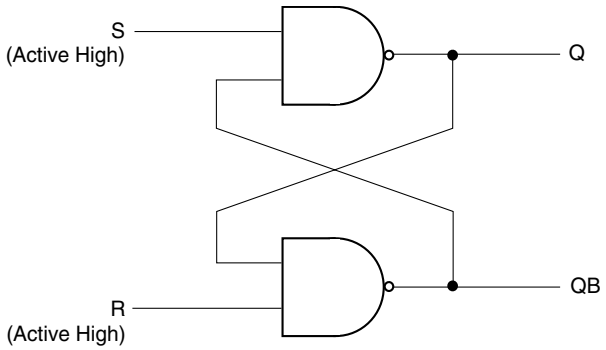
Non-recommended Circuits

Non-recommended circuits include cross-coupled NAND or NOR gates and RS flip-flops. These are illustrated in Figure 33, Figure 34 and Figure 35 below.

It is important to avoid the inadvertent creation of cross-coupled NAND/NOR gates by means of feedback loops within combination logic.

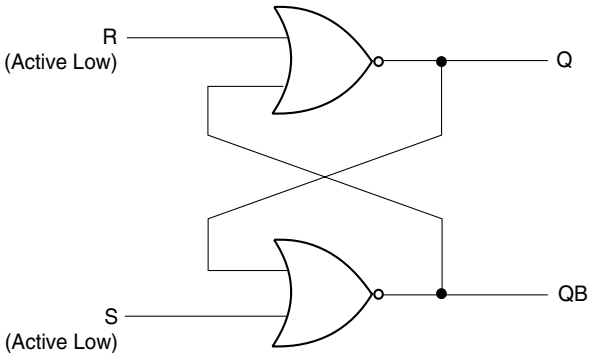
Cross-coupled NAND gates

Figure 33. Cross-coupled NAND gates forming bistable storage element



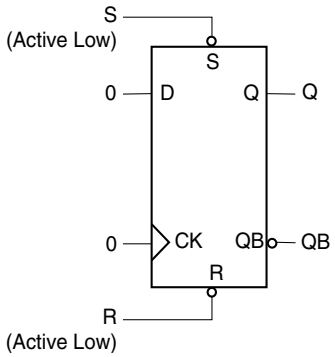
Cross-coupled NOR gates

Figure 34. Cross-coupled NOR gates forming bistable storage element



RS flip-flop

Figure 35. Asynchronous RS flip-flop



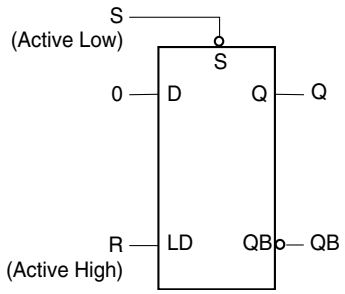
Recommended Circuits

The recommended methods of overcoming the problems listed in the previous section are as follows:

1. Use D-types with gated set/reset as required.
2. Use a latch configured as RS flip-flop. See the example circuit in Figure 36 below.
3. Avoid R-S races in the control of RS flip-flops.

Latch configured as RS flip-flop

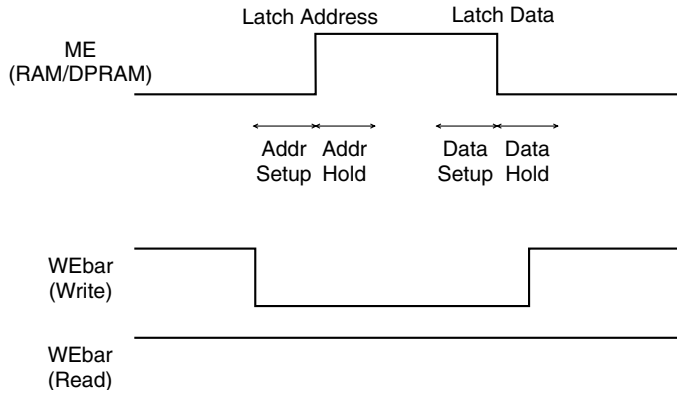
Figure 36. Latch configured as RS flip-flop



RAMs/ROMs in Synchronous Circuits

The problem of interfacing RAMs and dual-port RAMs into synchronous circuits is that they are double-edge triggered: the address is latched on the opposite clock edge to the data. This scheme is shown in relation to the ME and WEbar signals used by RAM and dual-port RAM in Figure 37 below. The ROM ME signal also latches the address on the rising edge.

Figure 37. ME and WEbar (RAM/DPRAM) timing scheme



Recommended Circuits

ME and WEBar Generation

To achieve synchronicity with the rest of the circuit, connect the RAM or dual-port RAM ME signal to an inverted system clock. One method of generating the WEbar signal is to use a D-type flip flop, with the inverted ME signal driving the clock, and an active-high external write request (wext) driving the d-input. The Webar signal is taken from the qb output. This produces the required delay of WEbar with respect to ME. This configuration is shown in Figure 38, and the resulting waveforms for a write cycle in Figure 39.

Figure 38. Interfacing RAM/DPRAM into a synchronous circuit

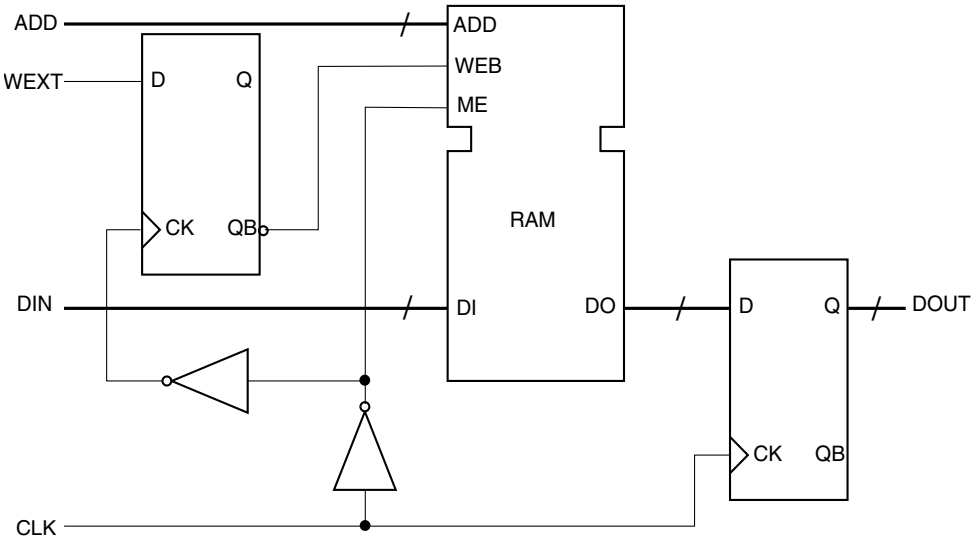
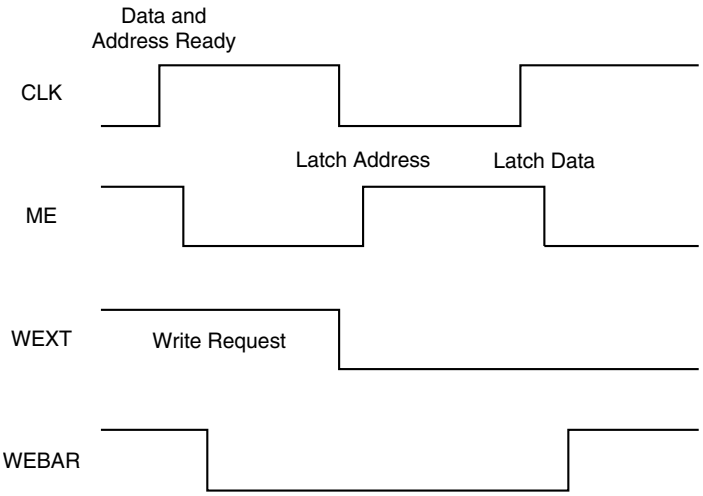


Figure 39. ME and WEbar timing scheme using flip-flop for WEbar generation

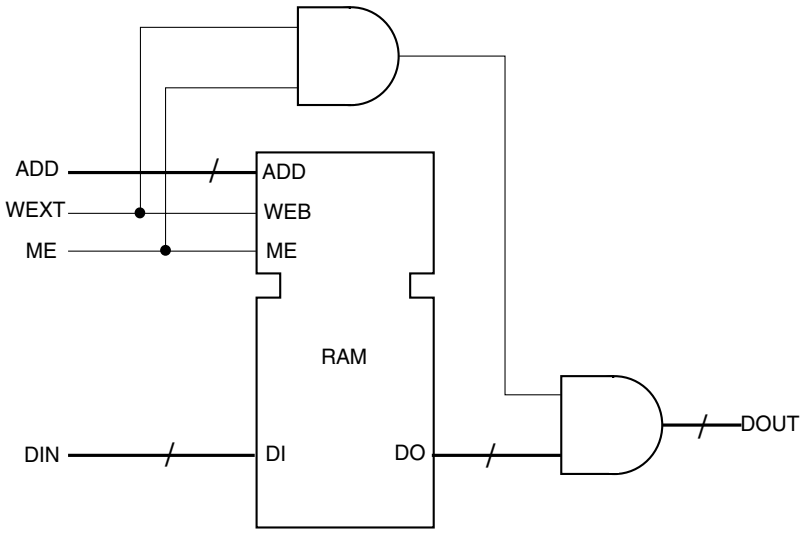


A consequence is that the clock duty cycle needs to be checked: the shorter phase needs to be longer than the setup and hold times and maximum propagation delay in the RAM, ROM, dual-port RAM and interfacing circuitry.

Avoiding Floating Outputs during Write Phase

During a write cycle, the output of a RAM/DPRAM (with tristate outputs) is floating. The propagation of this state can be avoided by means of the circuitry shown in Figure 40.

Figure 40. Avoiding floating RAM/DPRAM output propagation



Internal Tristates

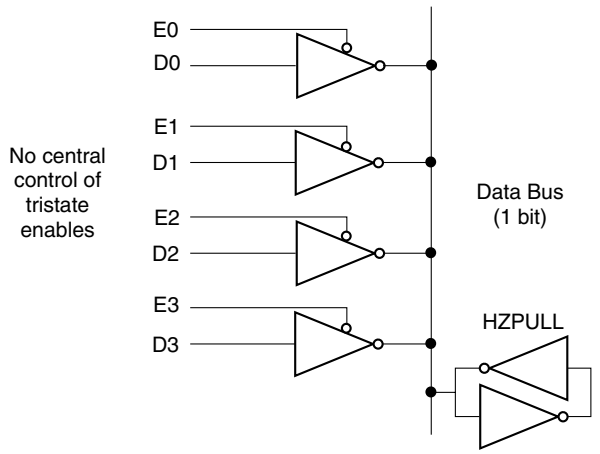
Internal tristates for data bus access within a circuit must be used with care, and should be avoided if possible. Potential problems are an undriven bus (particularly at initialization time) and conflicting bus drivers. An undriven bus floats to an intermediate state, causing high static currents.

Non-recommended Circuit

The general configuration of a circuit which is susceptible to problems of tristate control is shown in Figure 41 below.

Local control of tristate enables

Figure 41. Tristate bus with no central control of tristate enables. Do **not** use the Hzpull cell as a memory device.



The tristate enables are controlled locally, with no means of ensuring that there is no conflict (two driving simultaneously) or no undriven state, with no driver switched on. The Hzpull part retains the existing state of the bus, but it cannot initialize a tristated bus and creates asynchronous storage.

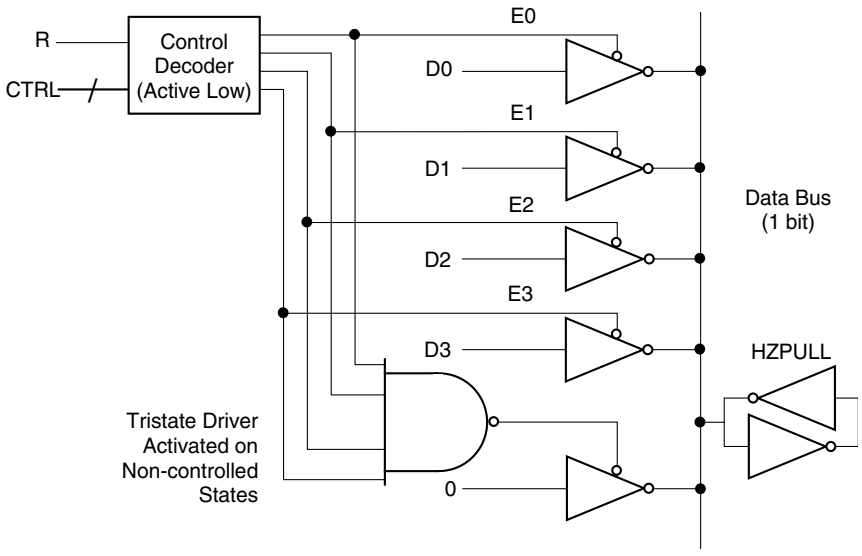
Recommended Circuits

1. Decode tristate control through a central control decoder. *It is recommended that the operation of this decoder is documented by means of a truth table or Karnaugh map.*
2. Provide one driver which is activated on non-controlled states. In particular, ensure that this driver is active during the reset state of the circuit.
3. Do **not** rely on Hzpull as a memory device. Its function is to prevent static dissipation, and it has a poor timing check.
4. Eliminate the tristates altogether by using multiplexed data bus lines. See “Multiplexers vs tristates” on page 26.

These three points are illustrated in Figure 42 below.

Central control of tristate enables

Figure 42. Tristate bus with central control of tristate enables and additional driver activated on non-controlled states



Note: The Hzpull part is not strictly necessary in the above schematic. It is included for additional security during control transitions.

Multiplexers vs tristates

5. Preferably, **multiplex** data lines instead of using tristate-driven buses. The factors to be taken into account are as follows:

Tristates (disadvantages):

- large area
- limited buffering
- large routing load, consequently slow

Multiplexers (advantages):

- small area
- efficient routing

Note: The control decoding is the same for a tristate-driven bus as for a multiplexed set of data lines.

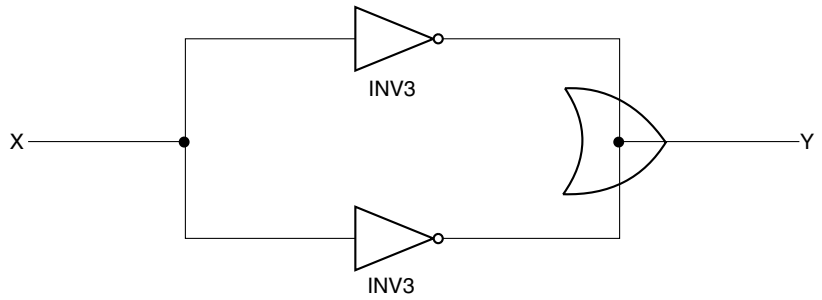
Paralleling Signals

For various reasons it sometimes appears necessary to include a wired OR or equivalent construction in a circuit, in order to provide parallel data signals. This practice is **not** recommended. *The use of wired OR parts should be avoided wherever possible.*

Non-recommended Circuit

Any circuit element which makes implicit or explicit use of the wired OR part is not recommended. An example is shown in Figure 43 below.

Figure 43. Wired OR part used to create higher fanout

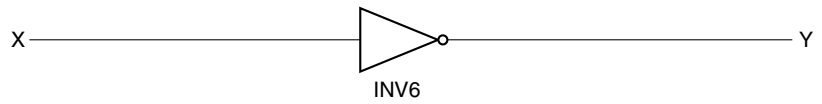


The function of this circuit may not be modeled properly, and there are placement and routing hazards.

Recommended Circuit

Use buffers of the appropriate strength and logic combinations which avoid the use of wired OR gates. The previous circuit can be replaced by the following equivalent:

Figure 44. Higher-fanout buffer replacing wired OR part



Fanout

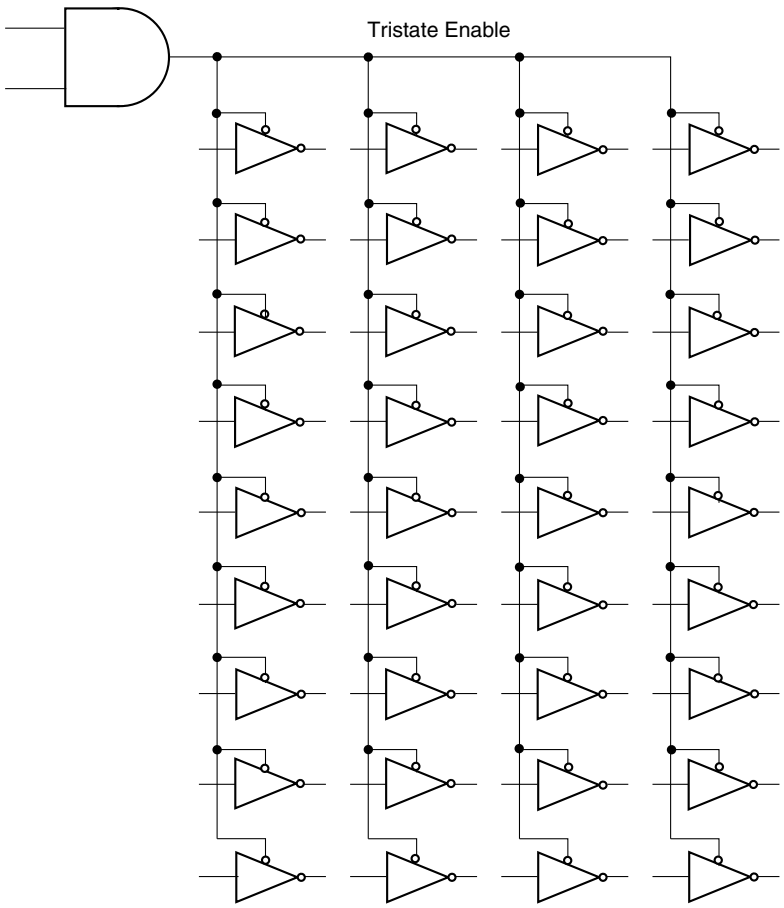
The relative fanout on any net in a circuit is the ratio of the total load (due to driven inputs and tracking capacitance) to the drive strength of the output driving the net. *In general the relative fanout should not exceed 12* (a process-independent figure derived from Atmel cell characterization data), otherwise the signals on the net are unacceptably delayed, and edges are unacceptably slow.

The special case of fanout in clock signals is dealt with in “Clock Buffering” on page 4.

Non-recommended Circuits

Any circuit which has excessive fanout on a data or control signal is not recommended. An example is shown in Figure 45.

Figure 45. Excessive fanout on control signal



Recommended Circuits

Use geometric or tree buffering in order to reduce fanout. Examples of each type are shown in Figure 46 and Figure 47.

Figure 46. Geometric buffering on control signal

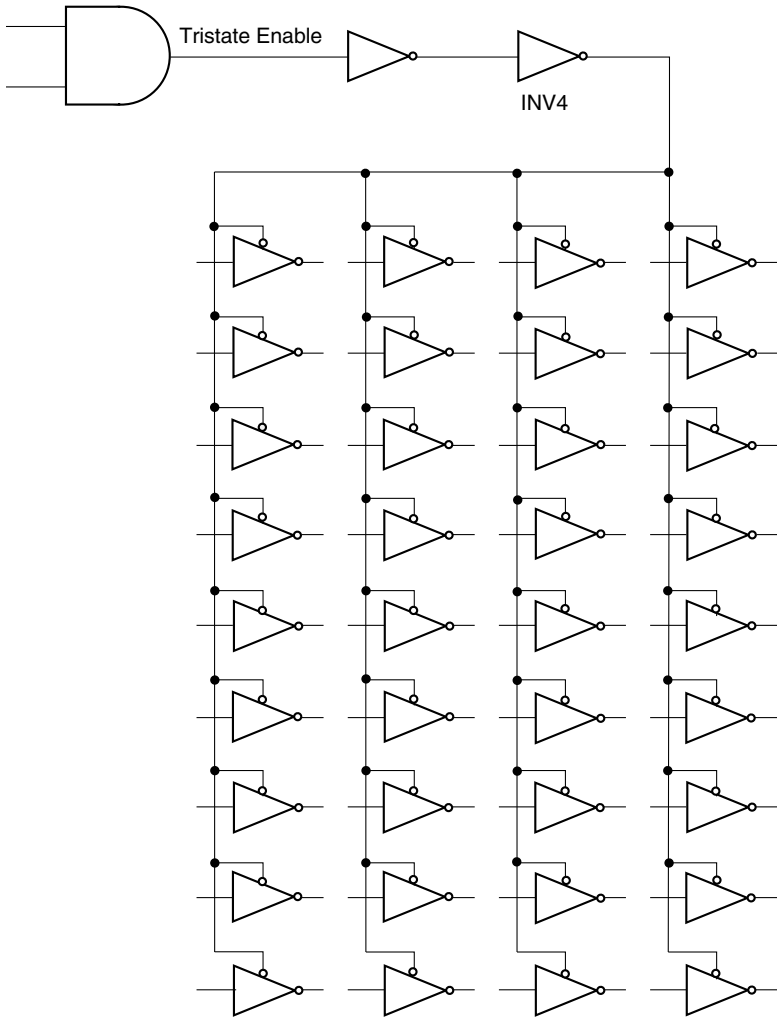
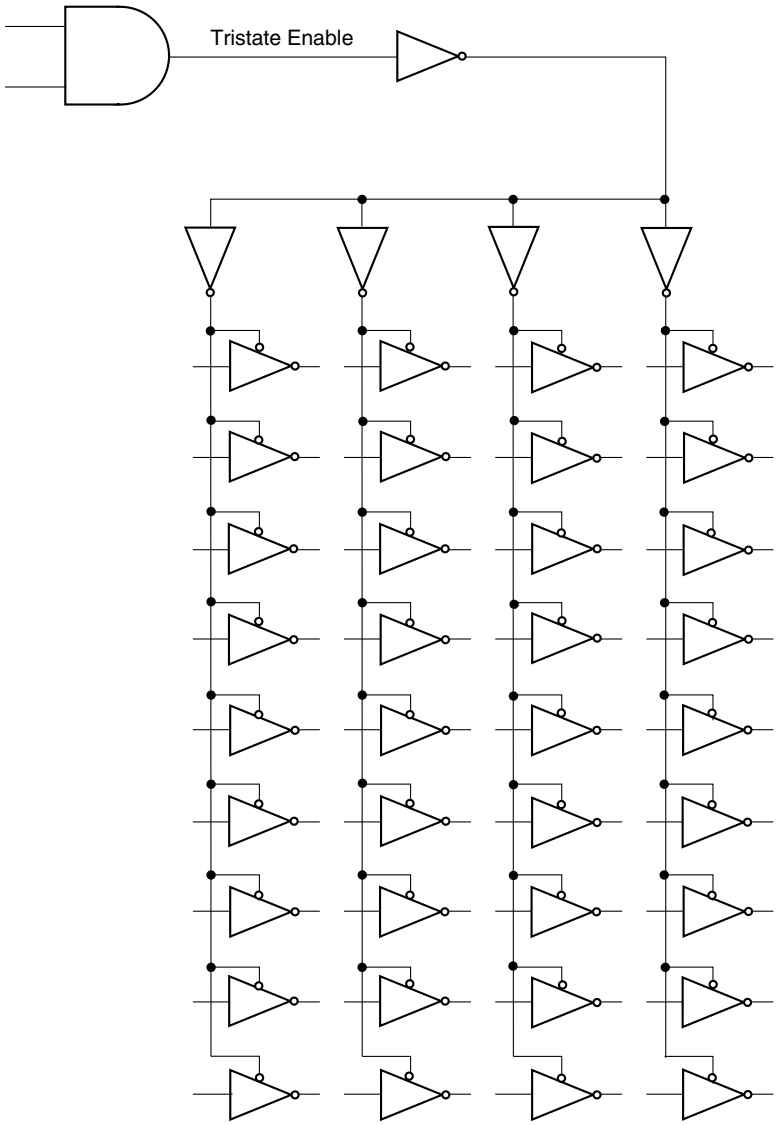


Figure 47. Tree buffering on control signal



Authorization

Relative fanout affects the speed of operation of a circuit. Given sufficient time, highly loaded nets will eventually settle to their correct logical value.

Accordingly, maximum relative fanout may be exceeded if no clock signals are involved, and data signals have sufficient time margin on input to clocked elements.

Design for Speed

A number of techniques can be used to increase the operational speed of a circuit. To an increasing extent, these are implemented automatically during design synthesis. If this is not available, some of the most popular methodologies are discussed in this section. *These generally involve a tradeoff between speed and silicon area.* These techniques are in addition to the fanout reduction methods described in “Fanout” on page 28.

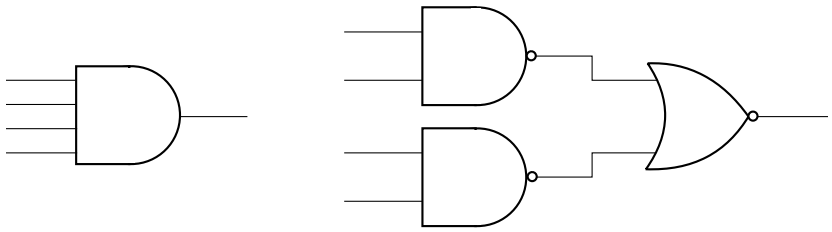
Recommended Circuits

Recommended techniques for increasing circuit speed, all of which involve safe design practices, are given below.

1. Use a maximum of 2 inputs on all combinational logic gates.

For example, an AND4 gate may be replaced by NAND/NOR logic as shown in Figure 48.

Figure 48. 4-input AND gate and equivalent NAND/NOR logic



AND4 gate

Equivalent faster NAND/NOR logic using 2-input gates

2. 2 Use AOI or OAI logic where possible.

AND/OR/Inverter (AOI) and OR/AND/Inverter (OAI) gates are particularly economical for both speed and area. *Their use is recommended wherever possible.*

Figure 49, Figure 50 and Figure 51 show three common examples of the use of AOI logic: a multiplexer, an enabled (E-type) flip-flop and a toggle (T-type) flip-flop.

Figure 49. Multiplexer using AOI logic

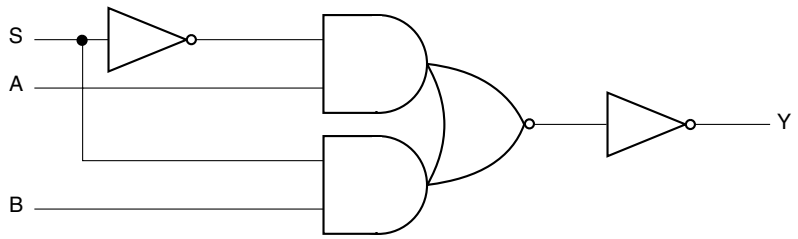


Figure 50. E-type flip-flop with reset constructed from AOI logic

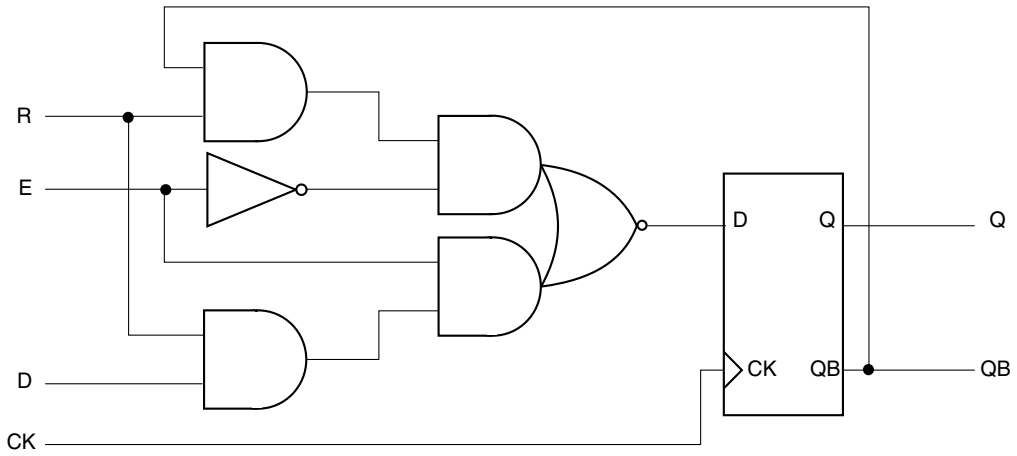
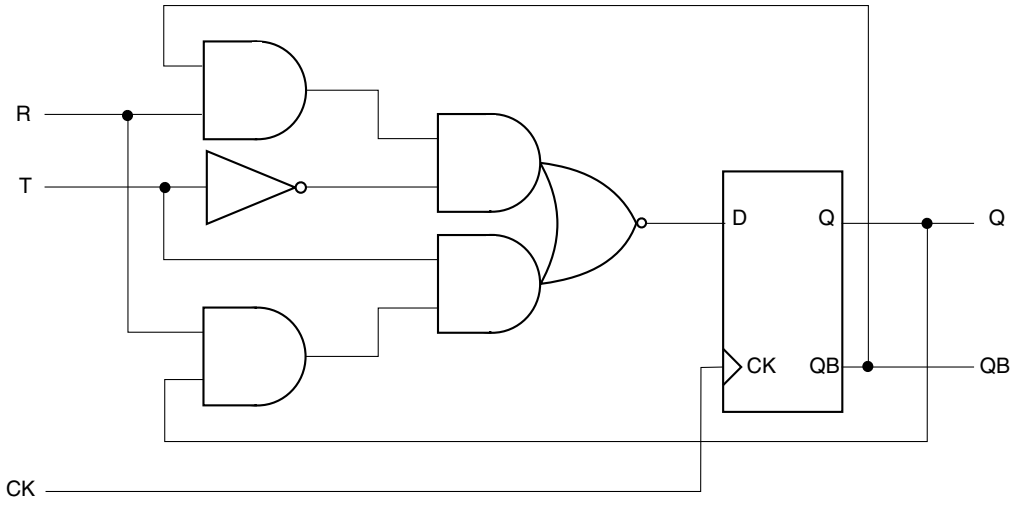


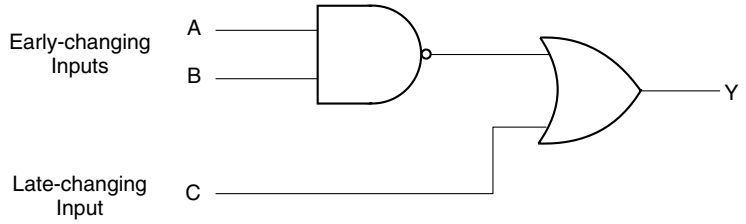
Figure 51. T-type flip-flop with reset constructed from AOI logic



3. Feed late changing inputs late into combinational logic.

An example of this technique is shown in Figure 52. *The aim is, as far as possible, to balance the total gate delay along each path of a combinational circuit.*

Figure 52. Late-changing input fed late into combinational logic



4. Use shift (Johnson) counters instead of binary counters.

A Johnson counter (Figure 53) is a shift register with the inverted output fed back into the primary input. An n-stage Johnson counter produces a set of distinct outputs of length 2n (see the truth table below), which can be decoded to give a count sequence. Its advantage is that, having no combinational logic between flip-flops, it can be run at the maximum speed permitted by setup and hold time constraints. The disadvantage of a Johnson counter is that, for a required count of m, it requires m/2 flip-flops, rather than log2(m) as required by a synchronous binary counter.

A Johnson counter can be provided with a synchronous reset, at the expense of an AND gate feeding into each flip-flop.

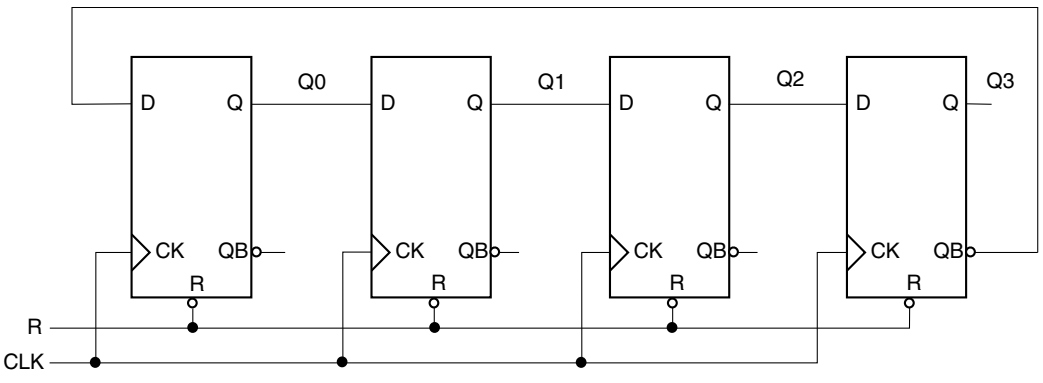
Note that the same buffering considerations apply to the clocking of long Johnson (and other) counters as they do to shift registers. See “Shift Registers” on page 15.

The truth table for a Johnson Counter is shown in Table 1 below.

Table 1. 4-stage Johnson Counter Truth Table

q0	q1	q2	q3
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
0	0	0	0

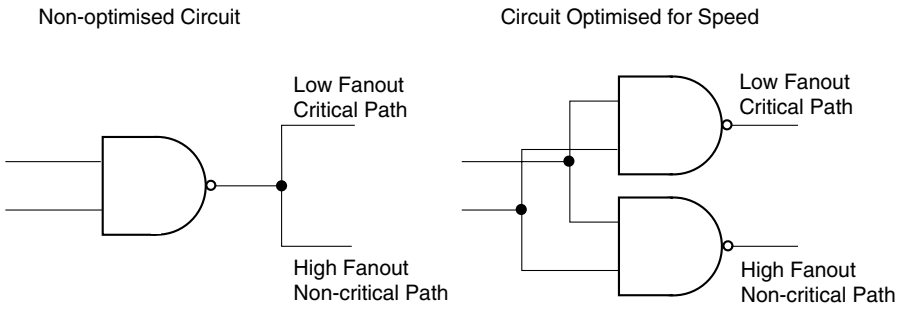
Figure 53. 4-stage Johnson counter



5 Use duplicate logic to reduce fanout.

This technique is analogous to the use of tree buffering to reduce fanout on clock signals. An example is shown in Figure 54.

Figure 54. Using duplicate logic to reduce fanout



6 Use fast library cells where available.

Consult the timing figures in the Atmel Databook for the relevant process, in order to identify the fastest available library cell for the required function.

7 Reduce the length of critical signal paths.

If a net priority scheme is available for automatic routing, raise the priority of nets in critical signal paths in order to give them preference in automatic routing.

In addition, if required, use the manual routing facilities of the design tool in order to identify the physical positioning of critical signal paths, and to re-route these manually in order to obtain the shortest path length.

8 Use Schmitt trigger inputs in noisy environments.

The use of Schmitt trigger inputs in noisy environments is strongly recommended.

Design for Testability

Testability within ASICs is based on the single ‘stuck-at’ fault model: *a faulty device is represented as having a single internal net held permanently at logic 0 or logic 1, regardless of how the net is driven*. Although this model is a simplification of the defects which can occur in an ASIC, experience has shown that it is adequate in practice for most purposes, and can form the basis of successful design-for-testability methodologies.

In terms of the single ‘stuck-at’ model, testability is based on two factors:

- **Controllability**: the ability to drive (from primary inputs) every internal net to both logic 0 and logic 1. In particular, the circuit must be reset to a known state within a specified number of clock cycles after initialization.
- **Observability**: the ability to detect (at primary outputs) that a single internal net is stuck at a state different from its driven state.

Controllability and observability are achieved by a combination of circuit design techniques and the selection of appropriate test vectors.

The fault coverage of a particular set of test vectors is:

$$\frac{\text{number of stuck-at faults identified by the vectors}}{2 \times (\text{number of nets in the circuit})}$$

Note that this formula is for net (output) `stuck-at' faults as defined in the first paragraph above, and not for input `stuck-at' faults.

Although the theoretical goal of 100% fault coverage is seldom achievable in practice, this section gives some techniques which enable an acceptably close figure to be obtained.

Non-recommended circuits

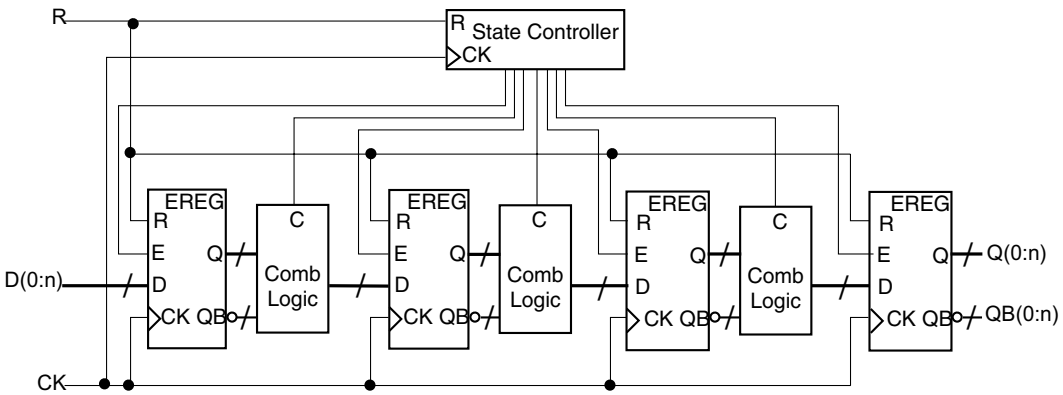
The following are examples of circuits with low observability/controllability. For each type of circuit, a recommendation is given later in this section for improving its testability.

Circuit with inaccessible internal logic

Figure 55 shows a typical circuit with a flow of bus-wide data through a sequence of enabled (E-type) registers linked by combinational logic. Control is by a central state controller, which has connections to the register enable lines and a control input to each combinational block.

From a testability point of view, the problem with this circuit is that only the first combinational block is directly controllable from external inputs, and only the last combination block is directly observable at external outputs. The central combinational block is neither directly controllable nor directly observable.

Figure 55. Circuit with inaccessible internal logic



Badly-designed state machines

For a circuit controlled by a state machine (such as that in Figure 55), problems can occur if the following two conditions are not met:

- all states must be decoded
- there must be no trap or lock states.

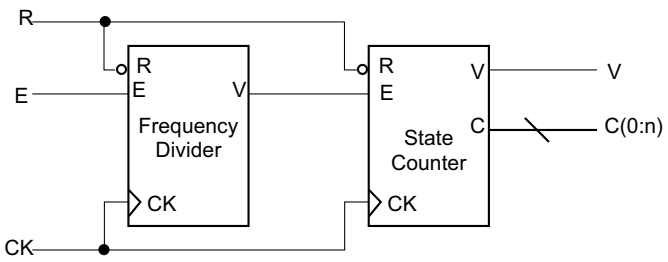
Chain of counters

Figure 56 shows the output of one counter feeding the enable of another. This configuration requires a large number of clock cycles to take the second counter through its entire sequence. The output from the first counter is not directly observable.

This design element violates both testability requirements: the first counter is not directly observable, and the second is not directly controllable.

A similar problem occurs with large single-element counters: they require a large number of test vectors (2^n for an n-bit counter) in order to take them through their entire count cycle.

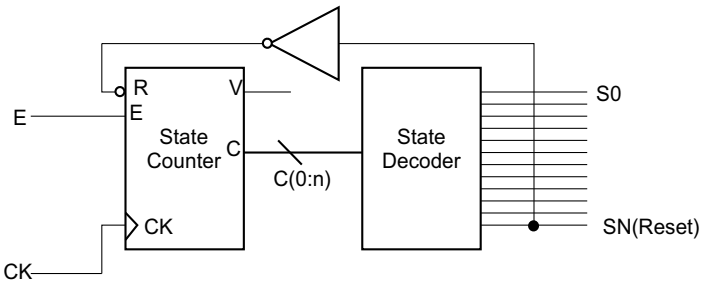
Figure 56. Chain of counters



Counter with feedback loop

The state counter in Figure 57 has its reset activated by a closed feedback loop triggered when it reaches an arbitrary internal state. This makes it impossible to reset to a known state at the start of a simulation.

Figure 57. Counter with closed feedback loop

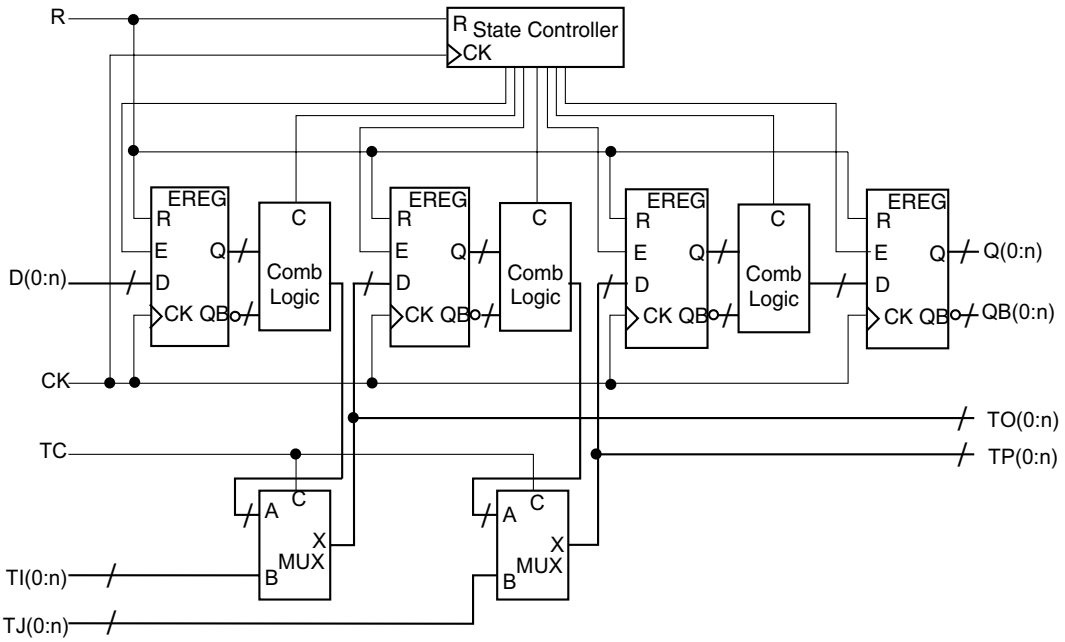


Recommended circuits

The following techniques are recommended for improving the testability of circuits which include elements of the types given in the previous sections. In any particular case, one or more techniques may be applied to a circuit, depending on its particular configuration. The preferred technique for complex, system-on-chip designs is scan path testing.

The techniques are described below in brief outline only. They do not form an exhaustive list, but are the methods found to be most successful in practice for synchronous designs based on a single clock signal. For more details, consult a standard reference on testability techniques.

Figure 58. Circuit with test inputs and outputs



At the level of primary input/outputs, multiplexers may be inserted in order to combine test and operational data signals, thus reducing the pin overhead of test circuitry. Another possibility is to replace uni-directional input/output cells by bi-directional cells, using the other direction for test input/output. However, clock and reset signals must not be treated in this way. This level of multiplexing is not shown in Figure 58.

The minimum requirement is for a separate test control signal, unless an otherwise unused combination of control inputs is used for test mode. If test control is achieved by an otherwise unused combination of control inputs, care must be taken to ensure that under no circumstances can the circuit be inadvertently placed in test mode.

1. Insert test inputs and outputs.

Additional inputs and outputs are inserted in order to make the internal logic of a circuit directly controllable and/or observable. Test inputs (ti(0:n) and tj(0:n)) are connected into the circuit via multiplexers. There is at least one test control signal (tc) in order to control the test multiplexers. Test out outputs to(0:n) and tp(0:n) are taken as outputs of the circuit element. The general concept is illustrated in Figure 58.

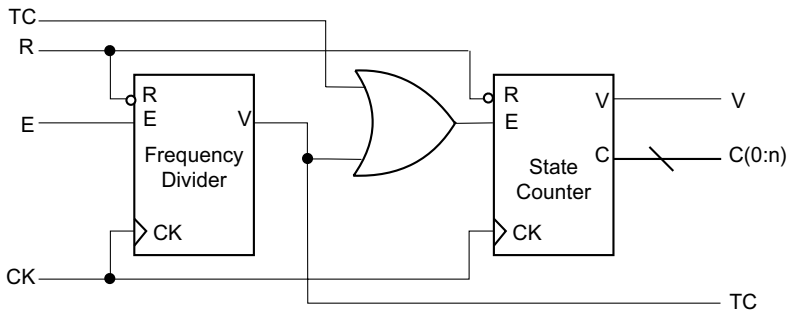
In test mode (tc high), test input data is fed directly into the internal registers, and test output is observed directly from the combinational logic blocks.

2. Break long counter/shift register chains.

Figure 59 shows how a chain of counters can be broken by a test control signal (tc) brought in using an OR gate. When the test control signal is high, the second counter increments at every clock cycle. The output from the first counter is taken to a primary output as the test output (to) signal.

A similar technique can be used to break up long single-element counters and shift registers.

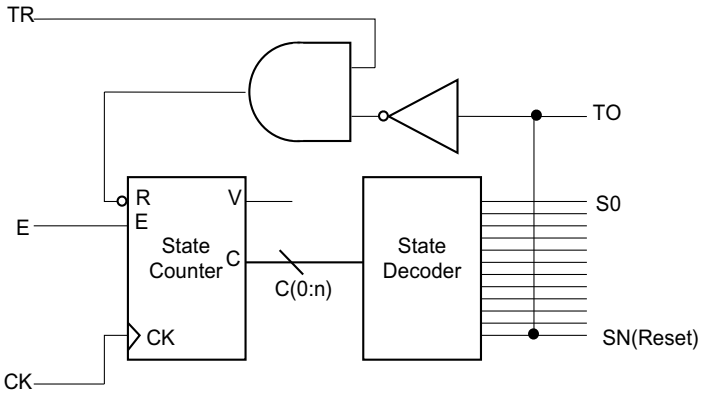
Figure 59. Chain of counters broken by test input and output signals



3. Open feedback loops.

Figure 60 shows how the feedback loop between a state decoder and a state counter is broken by the insertion of a test reset (tr) signal, and the connection of the reset state to a test output (to) signal. When tr is high, the circuit functions normally; the state counter is reset by forcing tr low. The value of the internal reset is monitored by the signal to.

Figure 60. Counter with feedback loop opened by test control and output signals



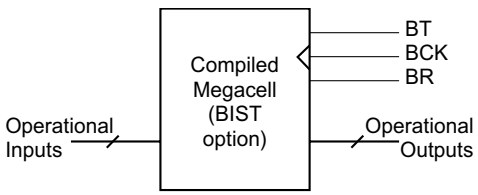
4. Use BIST with compiled megacells.

Compiled megacells (RAM, ROM, PLA, Multiplier, Dual-port RAM and FIFO) are inherently difficult to test, owing to their internal complexity, combinational depth and the small number of input/output signals relative to the number of internal cells.

To provide a solution to this problem giving 100% fault coverage for single 'stuck-at' faults, Atmel has implemented a *built-in self-test (BIST) option* for each compiled megacell. Selecting the BIST option produces a compiled megacell with customized test circuitry and three additional pins: BIST select (bt), BIST clock (bck) and BIST result (br). In most cases the BIST clock signal can be connected directly to the main system clock. See Figure 61.

In operational mode, the BIST select signal is low, and the normal working of the compiled megacell is unaffected. In BIST mode, with BIST select high, a sequence of clock edges on the BIST clock signal takes the megacell through a fixed test sequence which completely exercises all internal cells, and results in a unique signature on the BIST result signal after a specific number of clock cycles. On a fabricated device, any fabrication error within the megacell is revealed as a difference between the signature obtained and that resulting from a fault-free simulation of the device.

Figure 61. Compiled megacell with BIST input/outputs



5. Scan path testing.

An established technique for providing a high level of fault coverage in a circuit which has inaccessible internal logic is scan path testing. This requires the insertion of multiplexers in front of all storage elements in the circuit (such as the E-type flip-flop in Figure 62), and linking the additional inputs to form a single shift register which threads the entire device (Figure 63). This forms the scan path, from scan input si to scan output so. Note that buffering may be necessary between the q and so outputs of a scan flip-flop if they are both connected to external part-level or device-level outputs. Scan insertion can be performed automatically by sysntesis tools.

If the circuit contains gated clocks. dual-edged clocks or asynchronous control signals, multiplexers must be inserted where appropriate so that, in test mode, all clocked elements are activated on the same clock edge, and all control signals are synchronous. *Extreme care must be taken in inserting these multiplexers, in order not to introduce spikes or skew on clock or control lines.* See “Gated Clocks” on page 10.

Figure 62. E-type scan path flip-flop

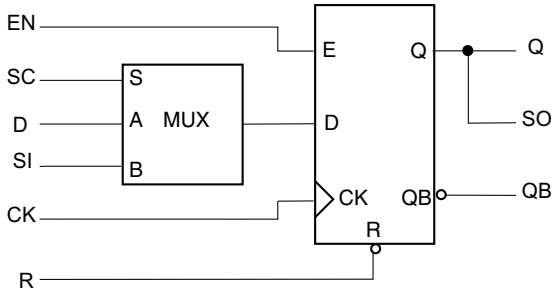
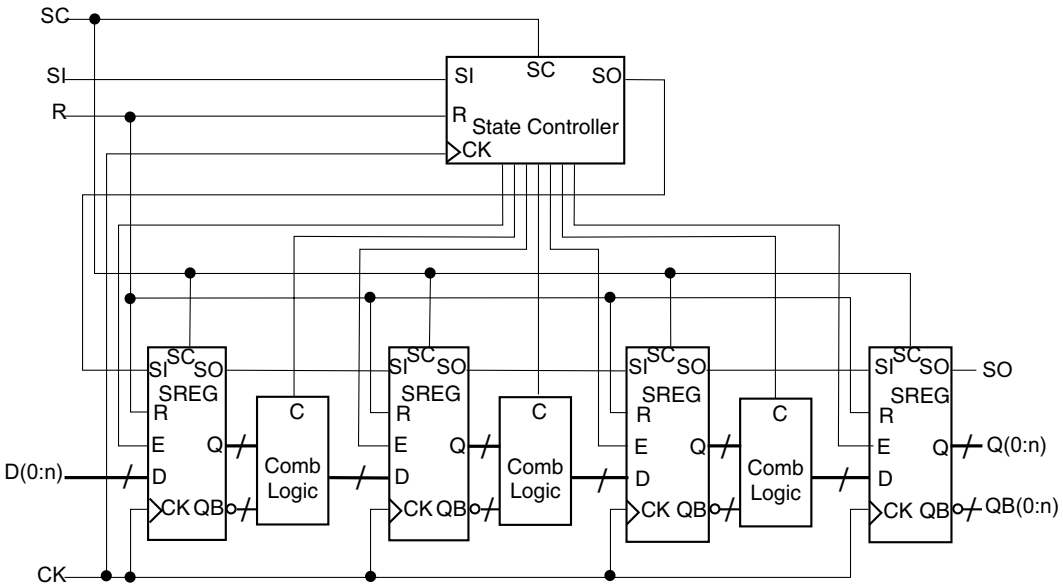


Figure 63. Circuit with scan path



In scan test mode, with the scan control signal sc high, a test pattern is shifted in serially through the scan path. The circuit is then put into operational mode for a single clock cycle, which propagates the test data through the combinational logic and back into the registers in the scan path. Again in scan test mode, the result is then shifted out through the scan chain. The test vectors are selected in order to exercise all the combinational logic in the circuit.

Test vectors for use with a scan path can be produced by an analysis of the logic functions implemented in the combinational logic blocks in the circuit. Alternatively, a pseudo-random binary sequence (PRBS) generator can be inserted at the start of the scan path, to produce a random test sequence. This can be combined with a signature analyser at the end of the scan path, which compresses the bit stream into a short, unique signature. Both the PRBS generator and the signature analyser are based on the technique of linear feedback shift registers. This method eliminates the need to develop a long set of test vectors, for a small overhead in silicon area. It produces an acceptably high level of fault coverage, and also permits in-service testing of devices.

As a further alternative, the software tools which perform scan path insertion also generate automatically a corresponding set of scan test vectors.

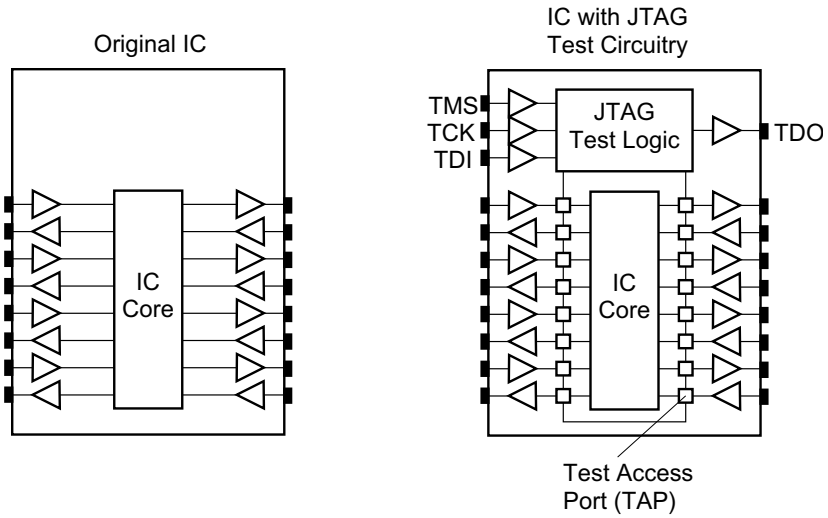
6. JTAG boundary scan path.

In addition to a scan path threading the internal logic of a circuit, a scan path may be constructed around the primary input/output cells. This technique follows the JTAG/IEEE 1149.1 standard, and provides a capability for board-level connectivity tests without the need to propagate test vectors through the core of a device. This is achieved by adding test circuitry and test pins to a device which enable all input and output pins to be connected together in a boundary scan path. The boundary scan path is a shift register with a parallel load facility which can be used to control and read the signal states on all input/output cells. See Figure 64.

In addition, the JTAG methodology allows the tester to interrogate an IC buried in the middle of the PCB, to run diagnostic checks, to identify the IC, or to sample the signal states of its pins during normal operation.

At PCB level, all such scan chains are connected together in series (parallel branches are permitted).

Figure 64. JTAG test circuitry



Test Vector Generation

Functional and Post-fabrication Tests

In the complete ASIC design and fabrication cycle, there are two main test phases:

- **functional testing**, to check whether the circuit design conforms to its functional specification
- **post-fabrication testing**, to ensure that each device has been fabricated without any faults.

It is essential to distinguish between the purposes of these tests: functional tests are concerned with the operation of the ASIC relative to its specification, post-fabrication tests are to check for differences between the operation of each individual die and an ideal, fault-free device (in practice a fault-free simulation of the circuit). This distinction is not always clearly drawn in practice, and in many cases the same simulation stimuli are applied to both types of test.

The fault-free simulation against which the post-fabrication tests are measured should (ideally) exercise all the nets in the circuit, in such a way that a fault in any net will show as a difference in an output signal. This simulation assumes (but does not verify) that the circuit is functioning according to specification.

This section gives some background information required for the generation of vectors intended for post-fabrication tests, their essential properties, and the specific requirements which they must satisfy. To an increasing extent, these test vectors are generated automatically as part of scan insertion. If this is not possible, then the guidelines in this section must be followed. These test vectors (both inputs and expected outputs) are supplied to Atmel, in a recognized format, with the design database for fabrication.

Static Test Vectors

An important concept which underlies the techniques for testing ASICs is that of static and dynamic circuits. These are defined as follows:

A **static** digital circuit is one which reaches a stable state a certain time after a set of inputs is applied, and then remains in that state for as long as it is powered up. For example, if the clock were to stop, the circuit would remain in a stable state as long as power were supplied.

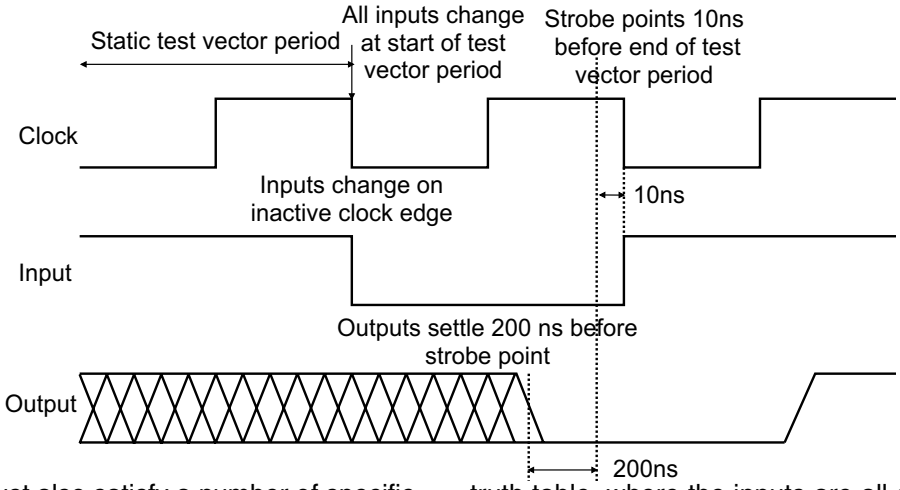
A **dynamic** digital circuit does not reach a stable state after a certain time, or the stable state represents a loss of data, such as leakage from a dynamic RAM. In dynamic circuits, a clock failure would lead to a loss of data.

Test vectors are designed on assumption that the device under test is a static circuit. They must also be compatible with the operation of the automatic test equipment used for post-fabrication tests. Test vectors which satisfy these conditions are called static test vectors. Their essential properties are as follows:

- For each test period, all inputs are applied simultaneously at the start of the period (1000ns, 10000ns or a multiple thereof).
- All outputs are strobed 10ns before the end of the test period
- All outputs and internal nodes must have reached a stable state 200ns before the strobe point.
- Input data must not change on the active edge of the clock which latches it.

The above three requirements for static test vectors are illustrated in Figure 65.

Figure 65. Static test vectors



- Static test vectors must also satisfy a number of specific requirements which are discussed in a later section.

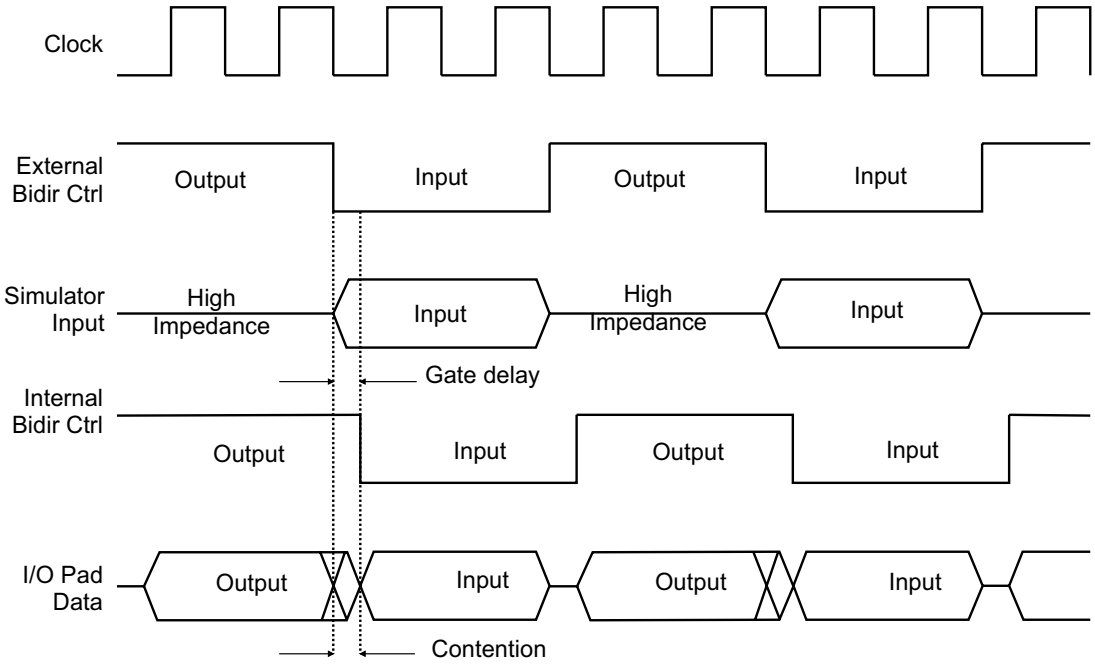
A consequence of the requirement for static circuits is that the behavior of a circuit under test can be described by a truth table, where the inputs are all applied together at the start of the static test vector period, and the outputs are at a stable state after the setting period.

Static Vectors for Bi-directional Input/Outputs

One circuit element which requires special attention in the design of static test vectors is the bi-directional input/output. *The problem is that there is almost always a short period of contention on the pad as it changes from output to input.* This is because the bi-directional control line is usually changed through internal logic, which produces a gate delay, whereas the simulated direction of the pad is determined externally, with no gate delay. Accordingly, the simulator has started to drive input data to the pad while it is still in its output state. The situation is illustrated in Figure 66.

Non-recommended protocol

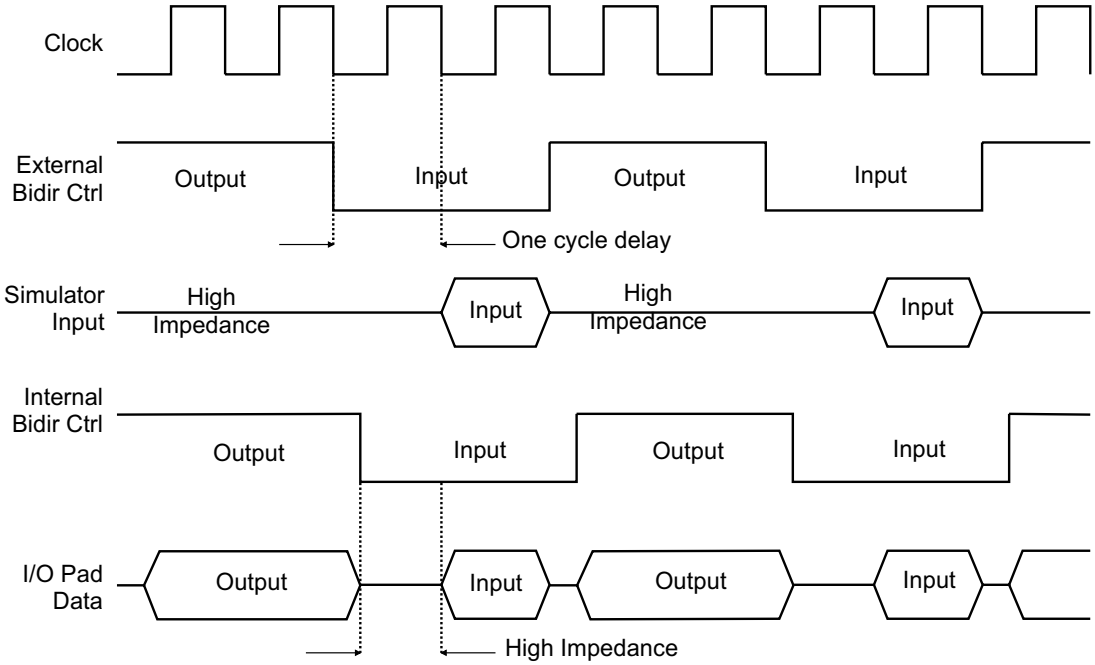
Figure 66. Contention on bi-directional input/output pad



The solution to this problem, to meet tester requirements, is to delay the application of input data by the simulator for a clock period after the transition from output to input. This is illustrated in Figure 67.

Recommended protocol

Figure 67. Delayed input on bi-directional input/output pad



ASIC Test Procedures

Post-fabrication tests are applied to ASICs using industry-standard automatic test equipment (ATE). The test vectors and pinout data submitted with a design are used to configure the ATE, and apply the test sequence. The overall procedure is as follows:

A truth table is extracted from the simulation vectors, and used to set up the ATE. The test equipment applies the test vectors at regular intervals (1000ns for digital designs and 10000ns or a multiple thereof for mixed analog/digital designs) and strobes the outputs from the ASIC 10ns before the end of this interval.

If any output differs from the one produced by fault-free simulation, then the particular device is rejected. A table is produced, showing the vector number at which each failure occurred, for the skew on each input.

In order to test the tolerance of the device to clock skew, the operational test cycles are repeated a number of times, each with one input advanced or retarded in steps of 10ns to a maximum skew of 80ns. (Note that the clock signal is not treated as a special case in the test process.)

If required, a variety of additional tests may be applied, including parametric tests and speed tests using functional vectors at full operational speed. These test such aspects as carry propagation.

Rules for Test Vectors

Design Rules

The following rules for test vectors are to ensure that they (and the associated design) are in general compliance with the design guidelines set out in this and previous sections of this document:

- An external master reset must be used (even if there is a POR cell) to ensure a complete device initialization.
- The package used must be in the Atmel ASIC Package Selector Guide.
- All I/O cell names must be from an Atmel Library or other known source, and not modified. If not, a Buffer Information Base (BIB) file must be created containing the name and details of the new I/O cell.
- Long counters and shift registers must have test access to intermediate stages.
- All internal feedback loops must be broken with test I/Os.
- Redundant logic must be avoided.
- Compiled megacells must either use the BIST option or have direct test access to input/outputs.
- Analog cells are peripherals. At most two analog cells may be connected in series.

- Different types of analog cells may not be placed in parallel, except if one is for input and the other for output.

Simulation Rules

Table 2 shows the nine possible simulation events or signal settings that can occur. The abbreviations shown are used in the list of checks that follows. I denotes either an input pad or a bi-directional pad configured as an input. O denotes an output pad, a bi-directional pad configured as an output, a tristate pad or an (internal) enable signal.

Table 2. Possible Simulation Events

Event value	Signal state		
Event type	0	1	X
I (input)	I:0	I:1	I:X
O (output)	O:0	O:1	O:X
Z (tristate)	Z:0	Z:1	Z:X

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

1205A–12/99/xM