VLSI Design: Challenges and Promise

An Overview

Dinesh Sharma

Electronic Systems, EE Department
IIT Bombay, Mumbai

September 11, 2015

Impact of Microelectronics

Microelectronics has transformed life styles in a very short time.

- Communications
- Personal Health
- Financial transactions
- Personal Computation
- Entertainment

Expectactions From the Technology

This wide reaching impact of microelectronics on ordinary life has raised expectations of the world from this field.

As VLSI designers,
we should know what the world expects
from what we are going to design

so that we may try to meet these expectations.

The Ideal VLSI: Asymptotic Limits

What does the world expect from an 'ideal' VLSI?

Complexity	$\rightarrow$	∞
Design Time	$\rightarrow$	0
Delay	$\rightarrow$	0
Power Consumption	$\rightarrow$	0
Cost	$\rightarrow$	0
Field Failures	$\rightarrow$	0
Testing Time	$\rightarrow$	0

A challenging prescription indeed!

(We wish we'd never asked!!)

∞ is defined as a quantity which has the property that given any arbitrarily large number G, $\infty > G$.

Major Challenges of VLSI Design

Complexity	$\rightarrow$	∞
Design Time	$\rightarrow$	0
Delay	$\rightarrow$	0
Power Consumption	$\rightarrow$	0
Cost	$\rightarrow$	0
Field Failures	$\rightarrow$	0
Testing Time	$\rightarrow$	0

1. Conquest over Complexity
2. Efficient Design process and manufacture
3. High Speed Design
4. Low Power Design
5. Handling mixed Digital, Analog and RF circuits
6. Efficient Test and Verification

Part I

Conquest over Complexity

Hierarchical Design

- Hierarchical Design Flow

Making Use of regularity

- Programmable Logic Arrays
- Sea of Gates

Taking A page out of the Software Designer's book

We must learn from the experience of software designers for handling complexity.

- We must use hierarchical Design,
- should prefer regular (repetitive) structures,
- should use text based, rather than pictorial descriptions. This is made possible by Hardware Description Languages.
- Re-use existing resources when possible.

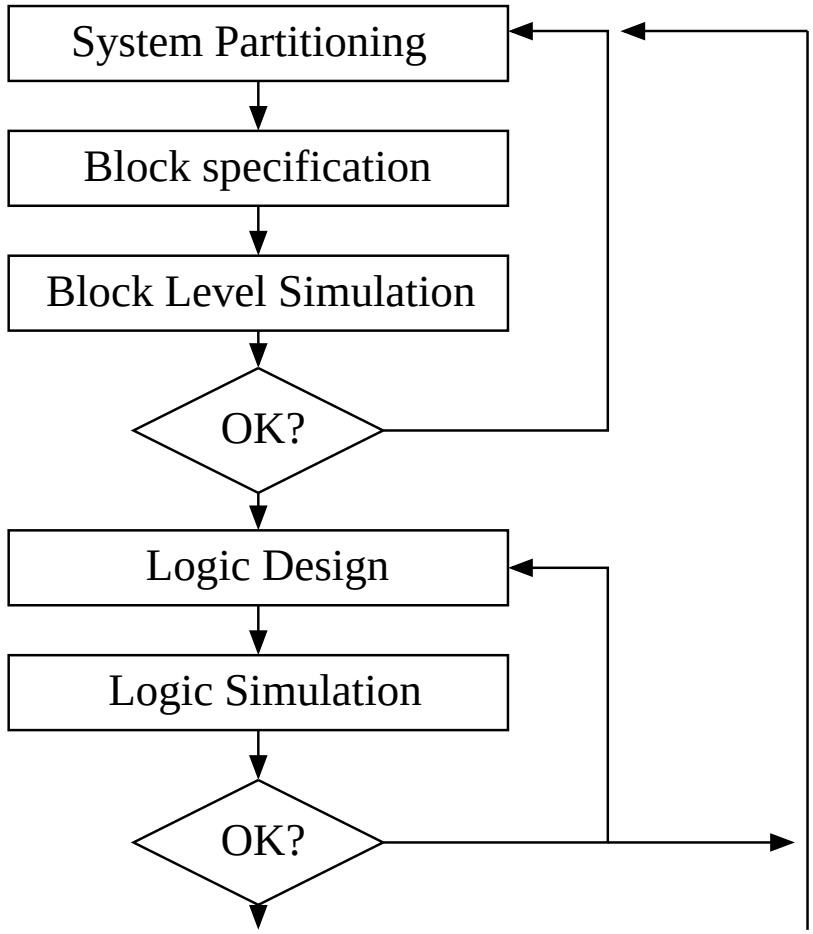
Hierarchical Design

- The main challenge for modern electronic design is that the circuits being designed these days are extremely complex.
-  While IC technology has moved at a rapid pace, capabilities of human brain have remained the same :-(
- The human mind cannot handle too many objects at the same time. So a complex design has to be broken down into a small number of ‘manageable’ objects.

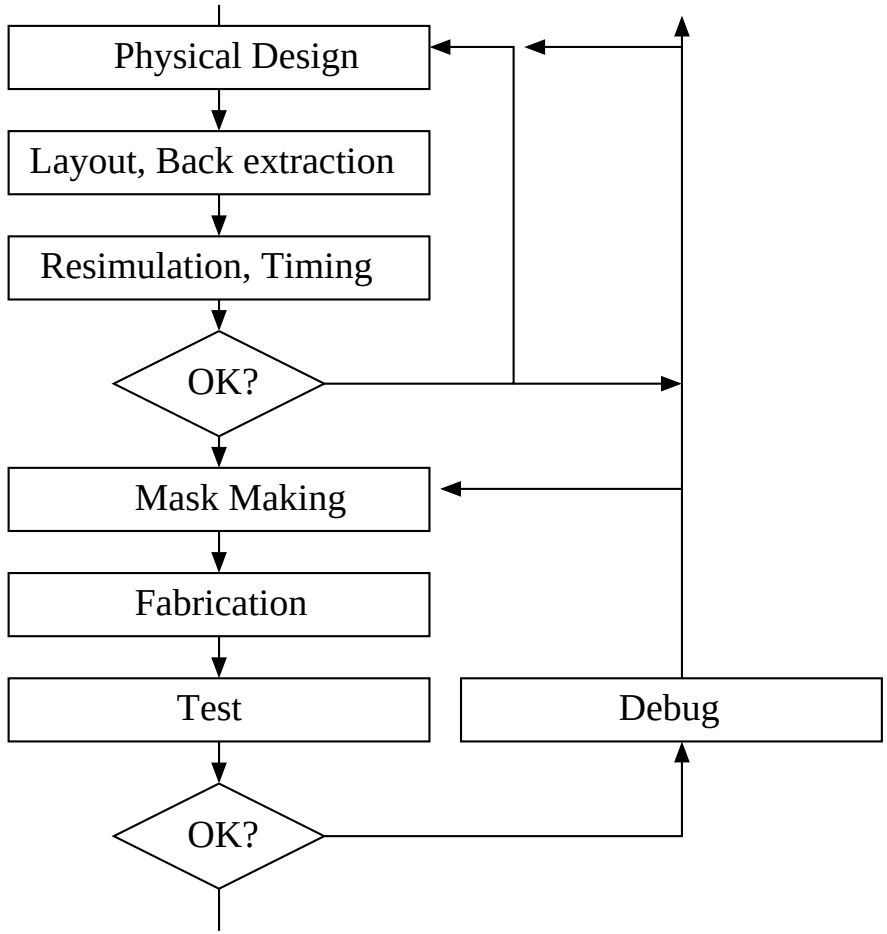
Hierarchical Design

- A complex design has to be broken down into a small number of ‘manageable’ objects.
- If each object is still too complex to handle, the above process has to be repeated recursively. This leads to hierarchical design.
- Systematic procedures have to be developed to handle complexity.

Design Flow: System and logic level



Design Flow: Physical level



Regularity in VLSI Design

If one looks at a VLSI structurally, it consists of active devices and interconnects.
Accordingly, we can classify designs in 4 classes:

Devices $\rightarrow$ $\downarrow$ Interconnects	Repetitive	Random
Repetitive	Memories	PLAs
Random	Sea of Gates	Processors

Only the random part needs to be designed separately. The repetitive part, after initial design, can just be replicated.

Making Use of regularity

By choosing to use design styles which make use of regularity, we can reduce the total design effort. In this approach, devices or interconnects are placed regularly and repetitively.

Semi-Custom design is one example of this approach. By choosing PLAs rather than random logic, design effort can be reduced considerably.

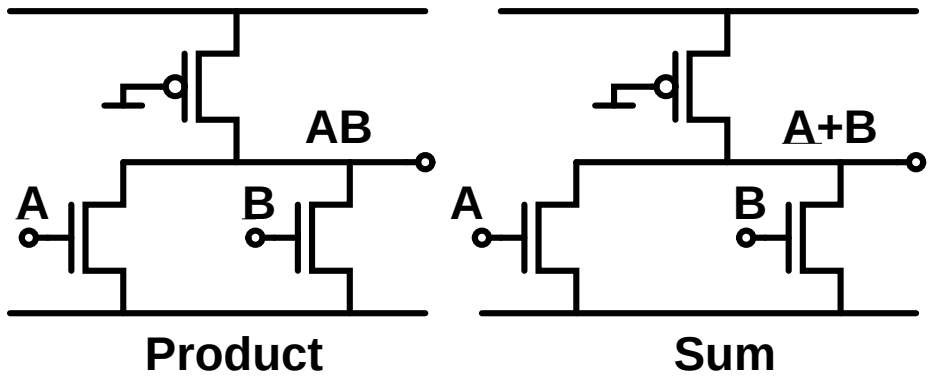
Re-configurable Logic with Pseudo NMOS gates

We want a regular circuit configuration which can be reconfigured to implement a general sum of products.

- It is inconvenient to reconfigure CMOS logic as configuration of the p channel pull up network as well as that of the n channel pull down network has to be changed.
- Pseudo NMOS gates are more suitable for reconfiguration as the pull up is just a single grounded gate PMOS.
- But Pseudo NMOS circuits are ratioed and the geometry of series connected devices depends on the number of devices in series and hence, on the logic.
- How do we implement the product function then?

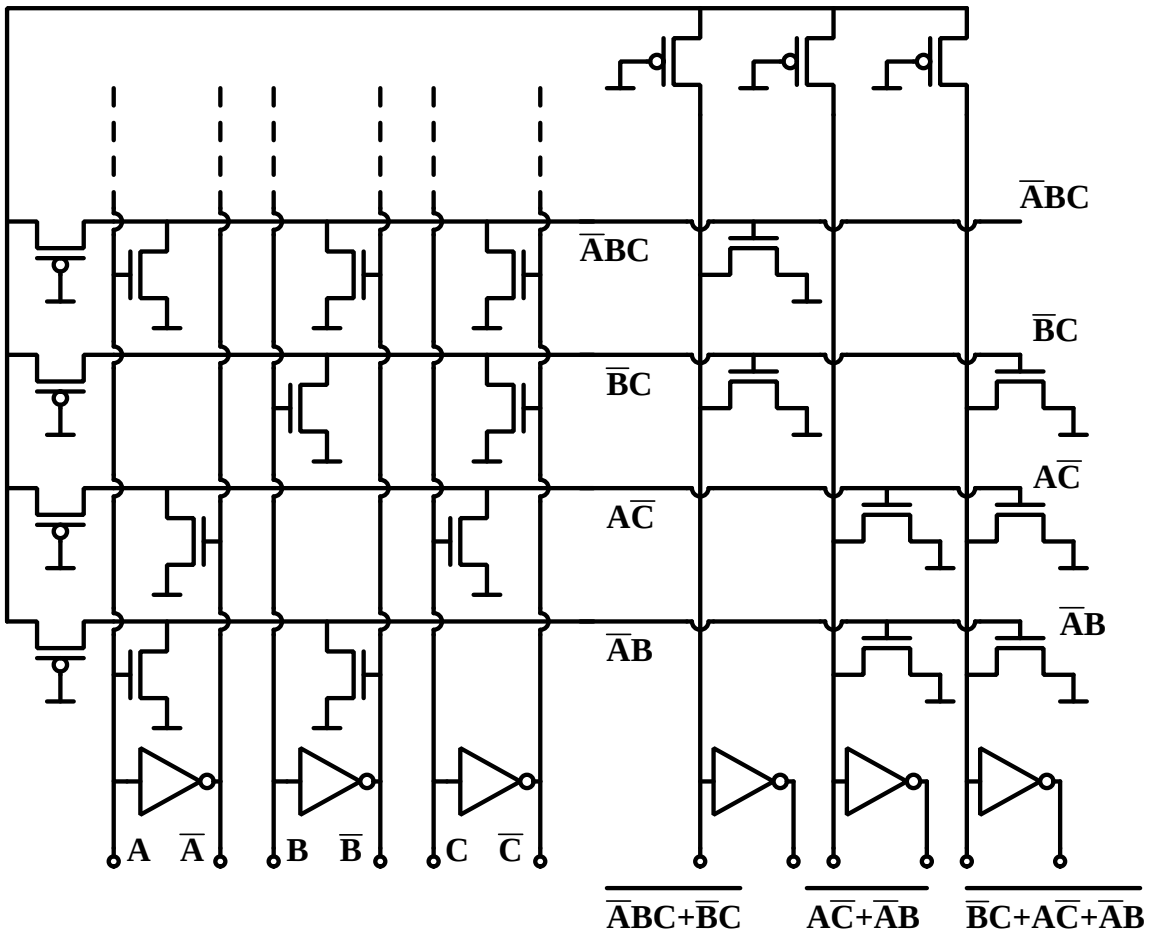
Pseudo NMOS NOR for Products as well as sums

- We use the expression : $A \cdot B = \overline{\overline{A} + \overline{B}}$.
- Now the product of A and B can be implemented as the NOR of $\overline{A}$ and $\overline{B}$ which does not use series connected transistors.



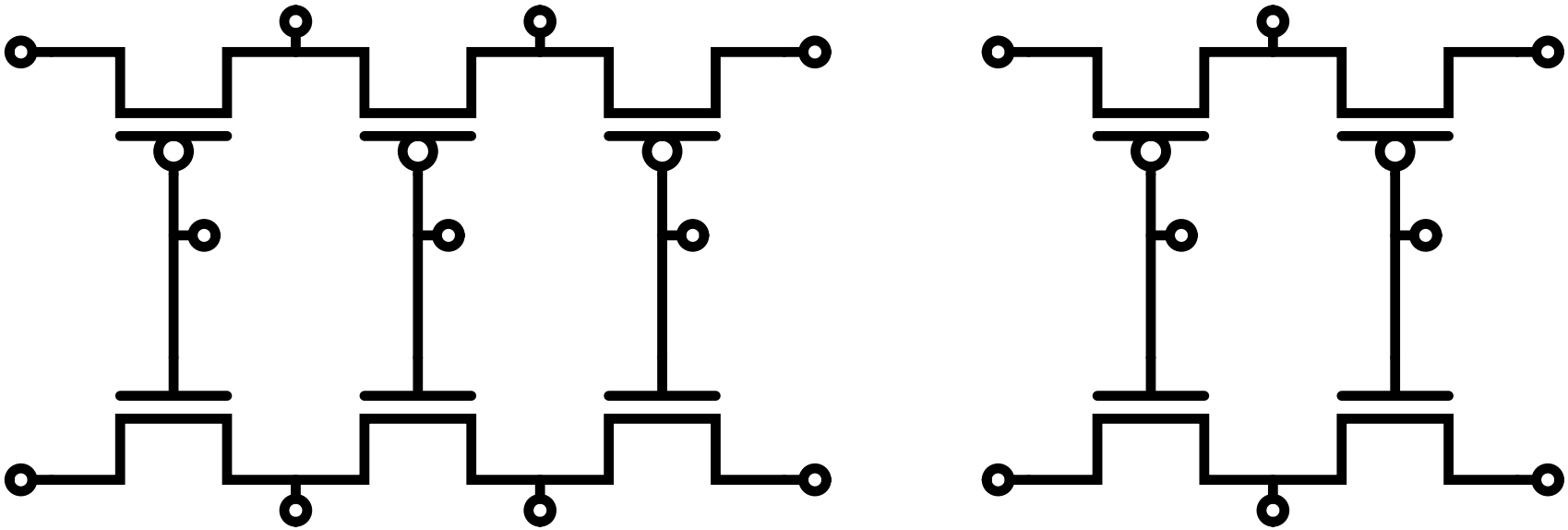
By adding inverters at the input and output as required, we can implement products or sums using only NOR type of circuits, which use constant geometry NMOS transistors in parallel.

Programmable Logic Arrays



Sea of Gates

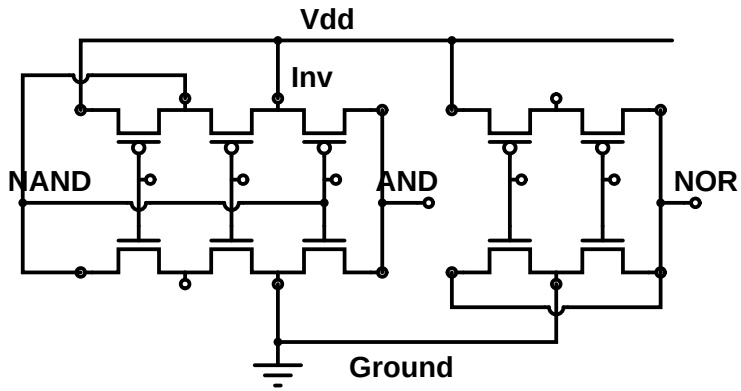
In this style of design, all transistors are pre-placed.
Interconnects determine what kind of logic will be implemented.



Both n and p channel transistors are in series here! How do we construct regular CMOS logic gates using these?

Logic from Sea of Gates

Actually, by wiring the apparently series connected devices, we can convert them to series or parallel as required.



The example above shows a NAND gate, whose output is fed to an inverter to form the AND function. The structure on the right forms a NOR gate.

Part II

Efficient Design Process

Efficient Design Process

Re-use of existing Resources

Text Based Descriptions

- Once the complexity becomes high, pictorial descriptions (like circuit diagrams) are inappropriate.
- We need text based descriptions.
- These are like programming languages used by software developers.
- Hardware description languages like VHDL and verilog provide this capability.

The Design Process

How do we fit in a programming language in electronic design?
So we ask our selves the question:
What is Electronic Design?

The Design Process

How do we fit in a programming language in electronic design?
So we ask our selves the question:
What is Electronic Design?

Given specifications, we want to develop a circuit by connecting *known* electronic devices, such that the circuit meets given specifications.

The Design Process

How do we fit in a programming language in electronic design?
So we ask our selves the question:
What is Electronic Design?

Given specifications, we want to develop a circuit by connecting *known* electronic devices, such that the circuit meets given specifications.

“Specifications” refer to the description of the desired behaviour of the circuit.

The Design Process

How do we fit in a programming language in electronic design?
So we ask our selves the question:
What is Electronic Design?

Given specifications, we want to develop a circuit by connecting *known* electronic devices, such that the circuit meets given specifications.

“Specifications” refer to the description of the desired behaviour of the circuit.

“Known” devices are those whose behaviour can be modeled by known equations or algorithms, with known values of parameters.

Electronic Design

Electronic Design is the process of converting
a behavioural description (What happens when ..)

to

a structural description (What is connected to what and how ..)

After conversion to a structural description, we may need to do "Physical Design" which involves choosing device sizes, placement of blocks, routing of interconnect lines etc.

This part is already done for us in FPGA based design.

But Hardware is different!

Hardware components are concurrent
(all parts work at the same time).

Whereas (traditional) software is sequential -
(executes an instruction at a time).

Description of hardware behaviour has timing as an integral part.

Traditional software is not real time sensitive.

Therefore, design of complex hardware involves many more basic concepts beyond those of programming languages.

Hardware Description Languages

Hardware description languages provide the ability to

- Describe
- Simulate at
  - Behavioural
  - Structural
  - and mixed

  level.
- and to synthesize (structure from behaviour).

HDL Uses

Hardware Description Languages are used for:

- Description of
  - Interfaces
  - Behaviour
  - Structure
- Test Benches
- Synthesis

Re-use of existing Resources

When a VLSI is conceptualized, we do not design the whole of VLSI from scratch.

The designer can ‘import’ other designs, which one need not understand in full detail. Only the interface has to be designed and understood

This is IP based design. (IP stands for Intellectual Property). Many modern designs incorporate designs from other companies as a component. For example, many communication circuits use the ARM processor as a component.

Part III

The High Speed Challenge

Impact of Scaling

Interconnect Delay
- Buffer Insertion
- Current Signaling

Asynchronous Design

High Frequency Models

Power Distribution

High Speed Design

- As devices are scaled down, not only do we get more dense and complex circuits, but also circuits which operate at higher speeds.
- This brings its own challenges.
  - Interconnect delays
  - Clock jitter
  - High frequency modeling for devices and interconnects
  - Substrate noise
  - Power Delivery

Consequences of Scaling

All dimensions and voltages divided by the factor S(>1).

Device area	$\propto W \times L : (\downarrow S)(\downarrow S)$	$\downarrow S^2$
C_{ox}	ϵ_{ox}/t_{ox} : const$/(\downarrow S)$	$\uparrow S$
C_{total}	$\epsilon A/t : (\downarrow S^2)/(\downarrow S)$	$\downarrow S$
V_{DS}, V_{GS}, V_T	Voltages : $(\downarrow S)$	$\downarrow S$
I_d	$\mu C_{ox}(W/L)(\propto V^2)$: $(\uparrow S)(\text{const})(\downarrow S^2)$	$\downarrow S$
Slew Rate $\frac{dV}{dt}$	$I/C_{total} : (\downarrow S)/(\downarrow S)$	*const.*
Delay	$V/\frac{dV}{dt} : (\downarrow S)/(const)$	$\downarrow S$
Static Power	$V \times I : (\downarrow S)(\downarrow S)$	$\downarrow S^2$
dynamic power	$C_{total} V^2 f : (\downarrow S)(\downarrow S^2)(\uparrow S)$	$\downarrow S^2$
Power delay product	delay $\times$ power$(\downarrow S)(\downarrow S^2)$	$\downarrow S^3$
Power density	power/area : $(\downarrow S^2)/(\downarrow S^2)$	*const.*

Impact of scaling

- Improved packing density: $\uparrow S^2$
- Improved speed: delay $\downarrow S$
- Improved power consumption: $\downarrow S^2$

However ...
The above improvements apply to active circuits.

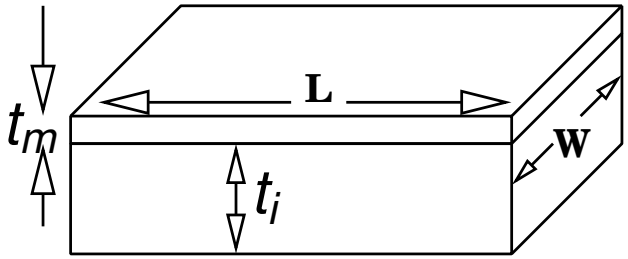
What about passive components?

Also, reduced voltages imply a lower signal to noise ratio.

Interconnect Design

- The chip designer has no control over material properties such as the resistivity of metal or the dielectric constant of insulator between metal layers.
- Structural parameters – such as the thickness of metal layer or the thickness of inter-metal dielectric are also chosen by the process designer.
- The length of the interconnect is also not a design parameter.
- The only parameters that a chip designer may choose are the layer in which to run the interconnect and the width of the wire.
- Typically, the upper layers offer thicker metal, but lower packing density.

Concern: Interconnect Delay

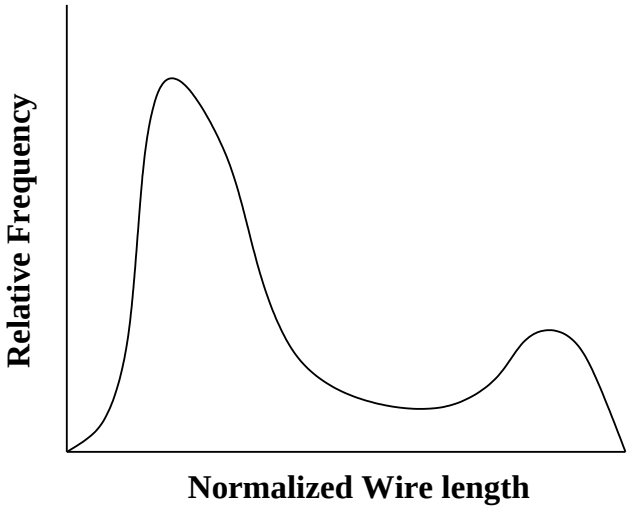


$$R = \rho \frac{L}{W t_m}, \qquad C = \epsilon \frac{LW}{t_i}$$

$$\text{Charge Time} \approx RC = \rho\epsilon \frac{L^2}{t_m t_i}$$

- To first order, delay is independent of W.
  This is because increasing W reduces resistance but increases capacitance in the same ratio.
- Unfortunately W is about the only parameter that the circuit designer can decide! (L is fixed by the distance between the points to be connected, ρ, ϵ, t_m and t_i are decided by the technology).

Concern: Interconnect Delay



- Local interconnects scale with device size.
- Global interconnects scale with die size.

Interconnect Delay = $\frac{\rho\epsilon}{t_m t_i} L^2 \equiv AL^2$

For local interconnects, L scales the same way as t_m, t_i, so delay is invariant.

For Global Interconnects, L goes *up* with die size, while t_m and t_i scale down. This leads to a sharp increase in delay.

Buffer Insertion

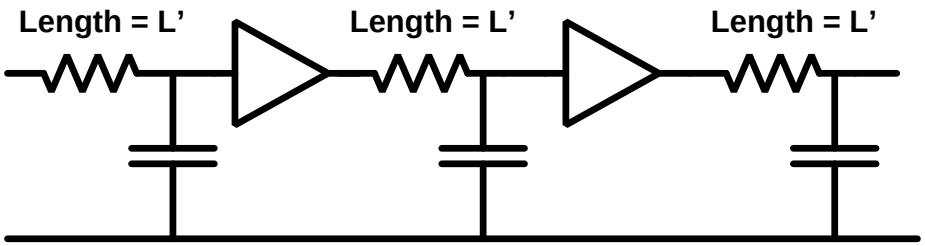
Global Interconnect delay can be the determining factor for the speed of an integrated system.

The L^2 dependence of interconnect delay is a source of particular concern.

This problem can be somewhat mitigated by buffer insertion in long wires.

We define some critical wire length and when a wire segment exceeds this length, we insert a buffer.

Repeater Insertion in Voltage Mode



Let the total wire length be L, with $n-1$ repeaters. Hence each segment has a length $L' = \frac{L}{n}$
Segment wire delay = AL'^2.
Let buffer delay = τ

For n segments, there will be n-1 buffers, and L = nL' .

$$\begin{aligned} \Delta &= (n-1)\tau + nAL'^2 \\ &= (n-1)\tau + \frac{AL^2}{n} \end{aligned}$$

What is the optimum number of segments in which we should divide this wire?

Optimal Buffer Insertion

We can differentiate the delay expression with respect to n and equate it to zero.

$$\begin{aligned} \Delta &= (n-1)\tau + \frac{AL^2}{n} \\ \frac{d\Delta}{dn} &= \tau - \frac{AL^2}{n^2} = 0 \\ \text{So } \tau &= \frac{AL^2}{n^2} = AL'^2 \\ n &= \sqrt{\frac{A}{\tau}}L \end{aligned}$$

This optimum choice of n corresponds to $AL'^2 = \tau$.
L' should be so chosen that the wire segment delay = τ.
Total delay is proportional to n and so, is linear in L.

Difficulties with Buffer Insertion

Currently, buffer insertion is the most widely used method to control interconnect delay.
However, there are several difficulties with buffer insertion.

- Buffers consume power and silicon area.
- Typically, we do floor planning and layout first and then put in the interconnects. When the wire length reaches L', we need to put in a buffer. However, it is quite possible that there is active circuitry underneath, and there is no room to put in a buffer!
- We either live with buffer insertion at non-optimal wire lengths or create space by pushing out existing cells and modifying the layout.

Problem with bi-directional data transmission

- Global interconnects often include data buses, which may require bidirectional data transmission. (For example, a bus connecting a processor and memory).
- However, buffer insertion fixes the direction of data flow!
- We need to replace buffers with bidirectional transceivers.
- These require a direction signal, which will enable a buffer in the desired direction.
- This direction signal must also be routed with the bus and should have its own buffers. It should reach the bidirectional buffers ahead of the data.

Promise of current mode signaling

- Why not signal with current rather than voltage?
- Current rise time is limited by inductance rather than capacitance. Typically, inductive effects are much smaller than capacitive effects.
  (After all, $\epsilon \simeq 4, \mu = 1$ for insulators used in IC's).
  So electromagnetic coupling is lower than electrostatic coupling.
- Signal voltage swings are limited by scaled down supply voltages: this does not restrict current swings.
- In fact, we could even use multiple current values to send more than one bit down the same wire!

Promise of current mode signaling

If we hold the Voltage on the interconnect nearly constant

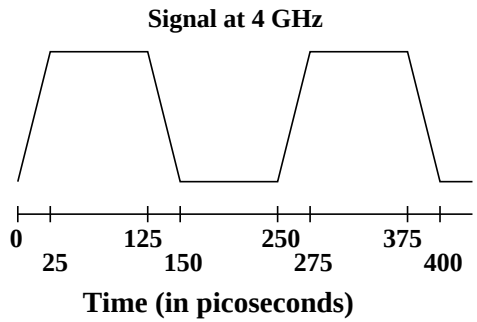
- Dynamic power is negligible.
- Latency is much lower.
- We also have the option of using multiple current levels to transmit multiple bits simultaneously.
  This can lead to:
  - Higher Throughput.
  - Lower interconnect area.

Possibility for improving Latency, Throughput and Power **simultaneously!**

Since $\Delta V \rightarrow 0$, while $\Delta I \neq 0$
$\Rightarrow$ We need a low (near 0) input impedance receiver.

“Synchronous” Design?

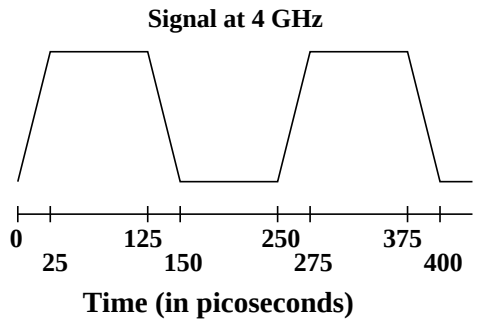
For a 4 GHz clock, rise and fall times of about 25 picoseconds would be expected.



- How far does light travel in a clock period?

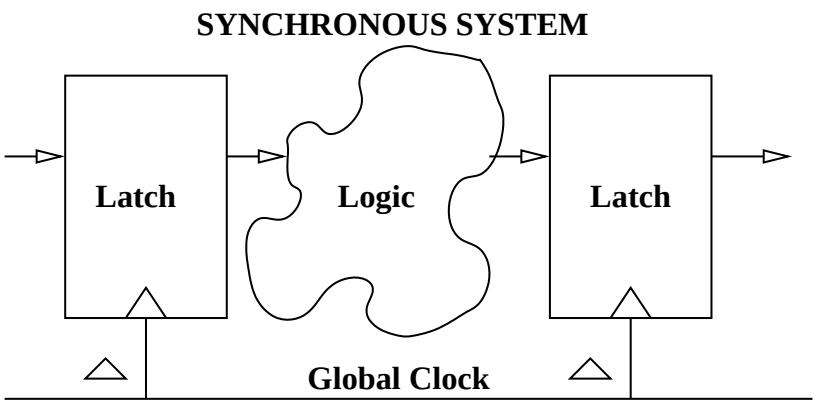
“Synchronous” Design?

For a 4 GHz clock, rise and fall times of about 25 picoseconds would be expected.



- How far does light travel in a clock period?
- Light travels just 7.5 cm - (comparable to the die size for most VLSIs!) in a whole clock period at today’s speeds.

Synchronous Design



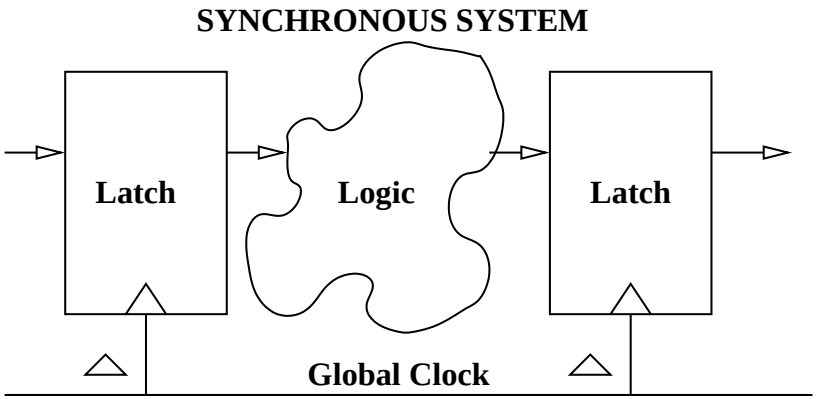
Synchronous design uses a global clock.
Clock Distribution introduces Skew and Jitter in clock arrival times.
Worst cases occur
For setup: Ck1 late and Ck2 early,
For hold: Ck1 early and Ck2 late.

Thus the worst cases for setup and hold violation are:

$$\text{Setup} : \Delta_{1max} = S_{max} + J_{max} \quad \text{and} \quad \Delta_{2min} = S_{min} - J_{max}$$

$$\text{Hold} : \Delta_{1min} = S_{min} - J_{max} \quad \text{and} \quad \Delta_{2max} = S_{max} + J_{max}$$

Synchronous Design



Worst cases occur
when Ck1 is late and Ck2 early,
and when Ck1 is early and Ck2 late.

To meet setup and hold requirements:

$$S_{max} + J_{max} + \text{Ck-to-Q} + \text{Logic}_{max} < T + S_{min} - J_{max} - \text{Setup}$$
$$S_{min} - J_{max} + \text{Ck-to-Q} + \text{Logic}_{min} > S_{max} + J_{max} + \text{Hold}$$

This puts a two sided constraint on logic delay, which may be difficult to meet for large skew and jitter values.

Synchronous Design

$$\begin{aligned} S_{max} + J_{max} + \text{Ck-to-Q} + \text{Logic}_{max} &< T + S_{min} - J_{max} - \text{Setup} \\ S_{min} - J_{max} + \text{Ck-to-Q} + \text{Logic}_{min} &> S_{max} + J_{max} + \text{Hold} \end{aligned}$$

This gives

$$\begin{aligned} \text{Logic}_{max} &< T - (S_{max} - S_{min}) - 2J_{max} - \text{Setup} \\ \text{Logic}_{min} &> (S_{max} - S_{min}) + 2J_{max} + \text{Hold} \end{aligned}$$

$S_{max} - S_{min}$ is the worst case variation due to skew. $2J_{max}$ is the worst case variation due to a jitter of $\pm J_{max}$ We can define the total variation in clock arrival times as

$$\Delta T \equiv (S_{max} - S_{min}) + 2J_{max}$$

Synchronous Design

$$\begin{aligned} \text{Logic}_{max} &< T - (S_{max} - S_{min}) - 2J_{max} - \text{Setup} \\ \text{Logic}_{min} &> (S_{max} - S_{min}) + 2J_{max} + \text{Hold} \end{aligned}$$

This can be expressed as

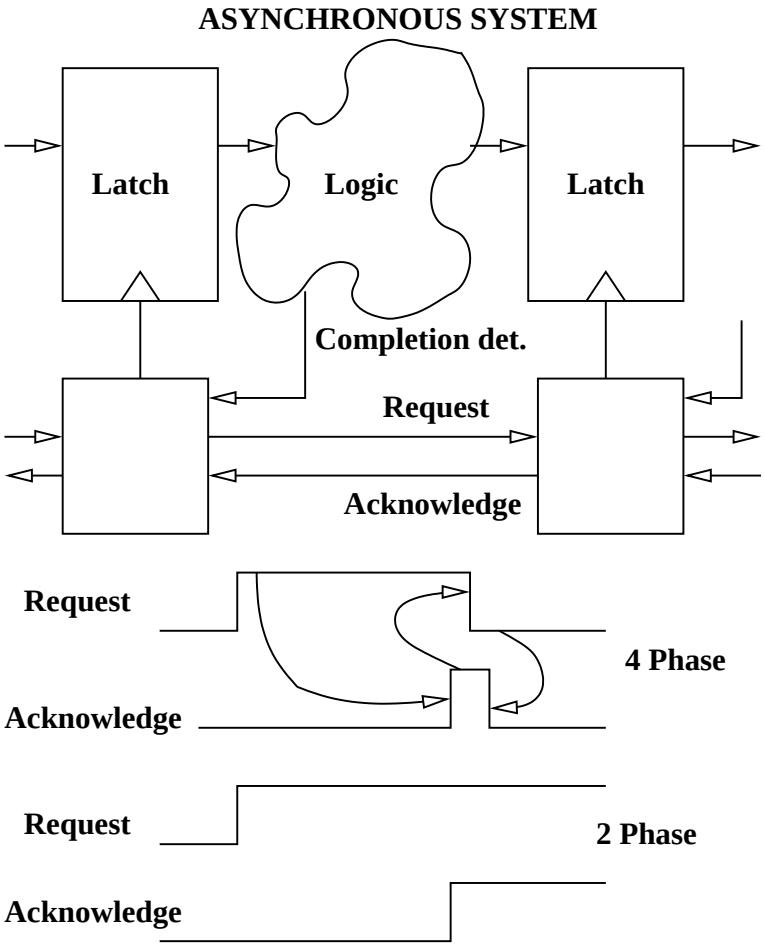
$$\begin{aligned} \text{Logic}_{max} &< T - \Delta T - \text{Setup} \\ \text{Logic}_{min} &> \Delta T + \text{Hold} \end{aligned}$$

where ΔT is the worst case variation in arrival time of the clock. The maximum logic delay condition can be met by slowing down the clock (increasing T). However, the minimum logic delay condition can only be met by controlling skew variation and jitter values.

Difficulties with Synchronous Design

- Scaling laws for global interconnects predict that that interconnect delays will become much worse in the future.
- Obviously signals, (propagating much slower than the speed of light), will take multiple clock periods to go from one end of the chip to the other.
- Synchronous design will thus become increasingly more difficult. (It is already quite hard to control clock skews to acceptable values!).
- Asynchronous design (which is difficult) will have to be used in the future.

Asynchronous Design



- Asynchronous design uses **local** request and acknowledge signals to forward the data.
- These signals ensure that the data is transferred only when it is safe to forward it.
- 4 phase signaling has 4 transitions per transfer, but is easy to design.
- 2 phase signaling has just 2 transitions per transfer, but needs event logic - which is more complex.

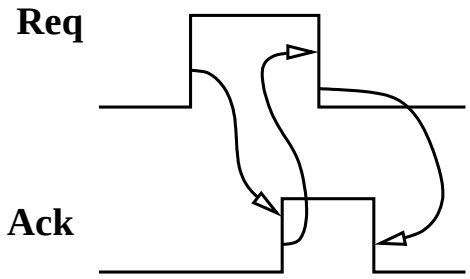
Asynchronous Design

- Asynchronous design uses **local** request and acknowledge signals to forward the data.
- All stages do not forward data at the same time.
- Throughput of synchronous systems is dominated by worst case delays. This is because the clock period is determined by the worst case and the same clock period applies to all stages.
- Throughput of asynchronous systems is determined by average delays.

Four Phase Request-Acknowledge Protocol

The standard Request Acknowledge protocol uses 4 phases.

- The initiator puts up Req to send/receive data.
- The responder puts up Ack when it has received/sent data. This can take any amount of time after Req has been asserted.
- The initiator removes Req on seeing Ack.
- The responder removes Ack on removal of Req.

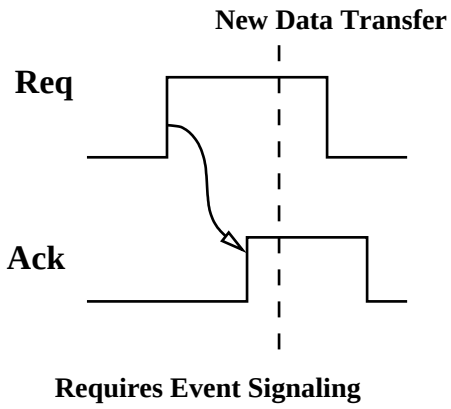


The last two phases are used just to return the signals to their passive state.

Two Phase Request-Acknowledge Protocol

The two phase protocol uses event signaling. Any change in the state of Req or Ack is considered an event.

- The initiator toggles Req to send/receive data. (Could be $0 \rightarrow 1$ or $1 \rightarrow 0$).
- The responder toggles Ack when data has been received/sent. (Could be $0 \rightarrow 1$ or $1 \rightarrow 0$).



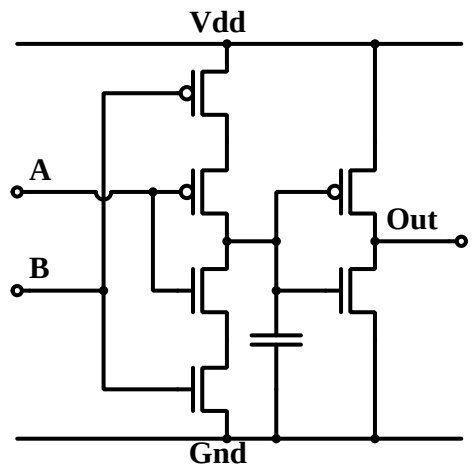
There is no need to have states where signals return to their passive states.

Now faster data transfer is possible.
However, we need to have logic which senses events rather than levels.

The C element

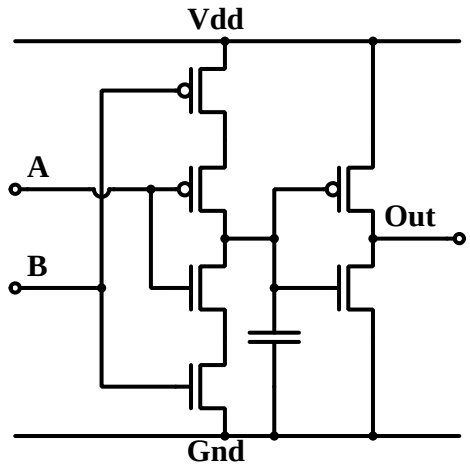
To implement event logic rather than level sensitive logic, we need special hardware elements.

- The C element is one such hardware structure. It uses 2 PMOS and 2 NMOS transistors, all in series.
- The output of this 4 transistor structure goes to a capacitor, which stores state.
- The logic level on the capacitor is inverted to generate the final output.



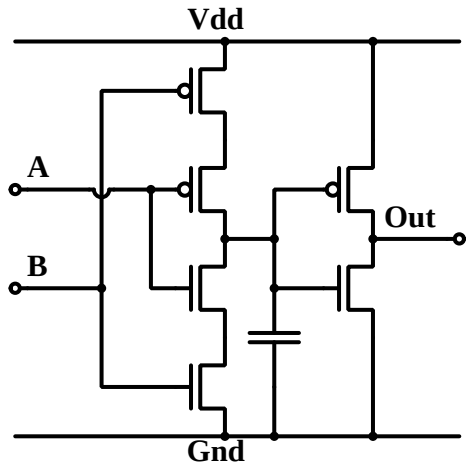
C element: Unequal Inputs

- When A and B are unequal, one of these should be ‘0’ and the other ‘1’.
- Thus, one out of the two series connected NMOS transistors is off.
- Similarly, one out of the two series connected PMOS transistors is off.
- In this case both the pull up and pull down are disabled and the the capacitor holds its previous value.
- So the output remains at its previous value.



C element: Equal Inputs

- When both inputs are ‘0’, the P channel transistors are ON while the N channel transistors are OFF in the first stage.
- The capacitor charges up to ‘1’ and so the output is ’0’.
- When both inputs are ‘1’, the N channel transistors are ON while the P channel transistors are OFF.
- The capacitor is discharged and the output goes to ‘1’.

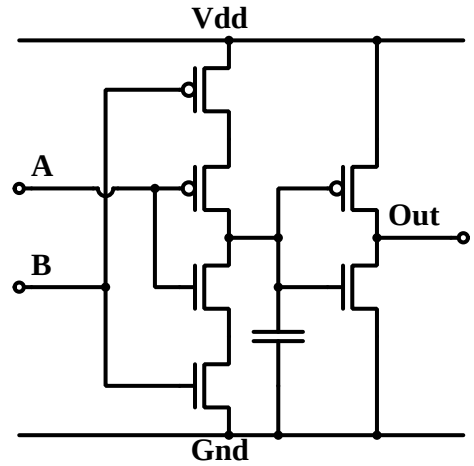


Thus when the inputs are equal, the output is the same as inputs. When inputs are unequal, the C element holds its previous state.

C element: AND of Events

Assume that both inputs are at ‘0’ initially. So the output is also at ‘0’.

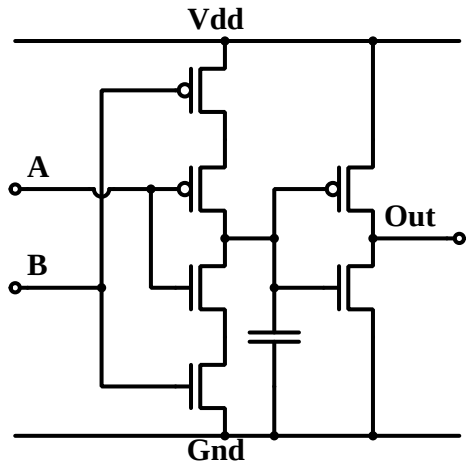
- If either input goes to ‘1’, the inputs become unequal and the output holds its previous value of ‘0’
- Subsequently, if the other input also goes to ‘1’, the output will be driven to ‘1’.
- If the other input does not go to ‘1’ but the first one returns to ‘0’, the inputs are equal to ‘0’ and the output remains ‘0’.
- Thus the output has an “event” only when *both* inputs have had an event.



C element: AND of Events

Assume that both inputs are at ‘1’ initially. So the output is also at ‘1’.

- If either input goes to ‘0’, the inputs become unequal and the output holds its previous value of ‘1’
- Subsequently, if the other input also goes to ‘0’, the output will be driven to ‘0’.
- Thus the output again has an “event” only when *both* inputs have had an event.

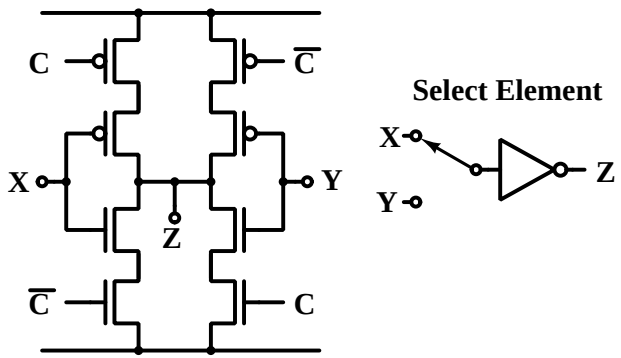


Thus the C element performs an “AND” on events. It is therefore represented as an AND gate with a C inside the symbol.

The selector element

The figure on the right shows a select element.

- When C = '0', the PMOS and NMOS connected to supply and ground in the left half are ON while the right half is OFF. So $Z = \overline{X}$.
- When C = '1', the PMOS and NMOS connected to supply and ground in the right half are ON while the left half is OFF. So $Z = \overline{Y}$.

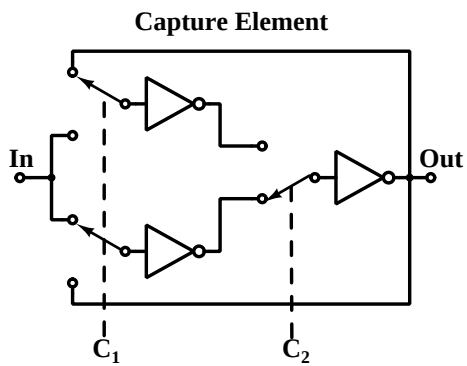


Thus, the behaviour of the circuit is that of a two way switch followed by inversion as shown in the figure above.

Capture Element

We can construct an event sensitive latch using selector elements.

- The circuit has two control inputs C_1 and C_2.
- Assume that C_1 = '1' puts the selectors in the up position, while C_2 = '1' puts its selector in the down position.
- C_1 and C_2 can be either equal or unequal.
- The circuit behaviour is the same when $C_1 = 0, C_2 = 0$ or when $C_1 = 1, C_2 = 1$.
- Similarly, the circuit behaviour is the same for the two unequal combinations $C_1 = 0, C_2 = 1$ or $C_1 = 1, C_2 = 0$.
- This makes it event sensitive (similar to C element).

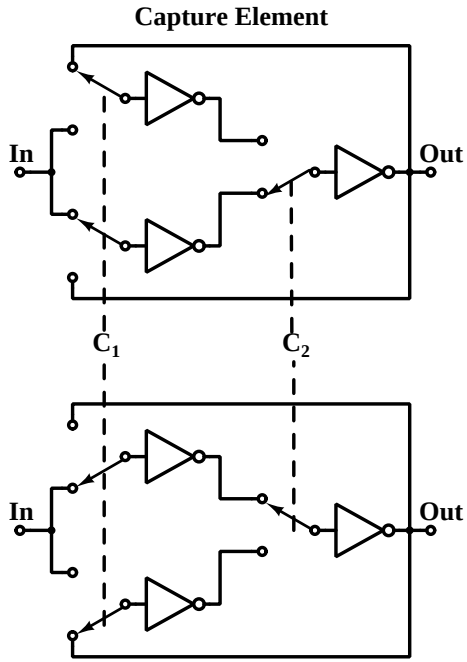


Buffer Condition

When $C_1 = C_2$, both can be '1' or both can be '0'.

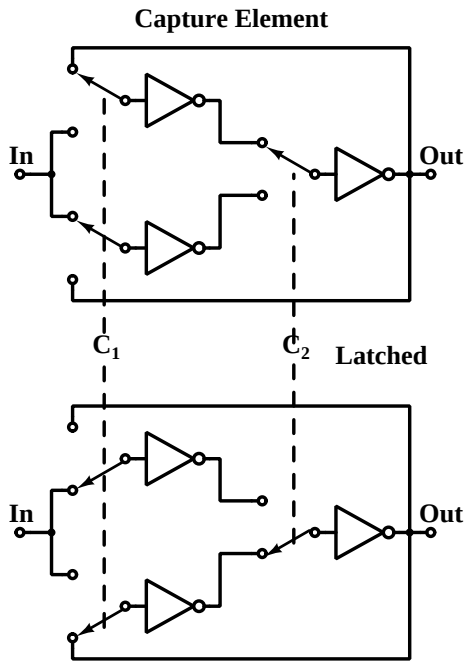
- When both are '1', the two switches on the left will be up and the right switch will be down. Data will flow through the lower inverter on the left and output inverter.
- When both are '0', the two switches on the left will be be down and the switch on the right will be up. Data will flow through the upper inverter on the left and the output inverter.

In either case, The input is buffered to the output



Capture Condition

When $C_1 \neq C_2$, either $C_1 = 1, C_2 = 0$ and all switches are up; or $C_1 = 0, C_2 = 1$ and all switches are down;

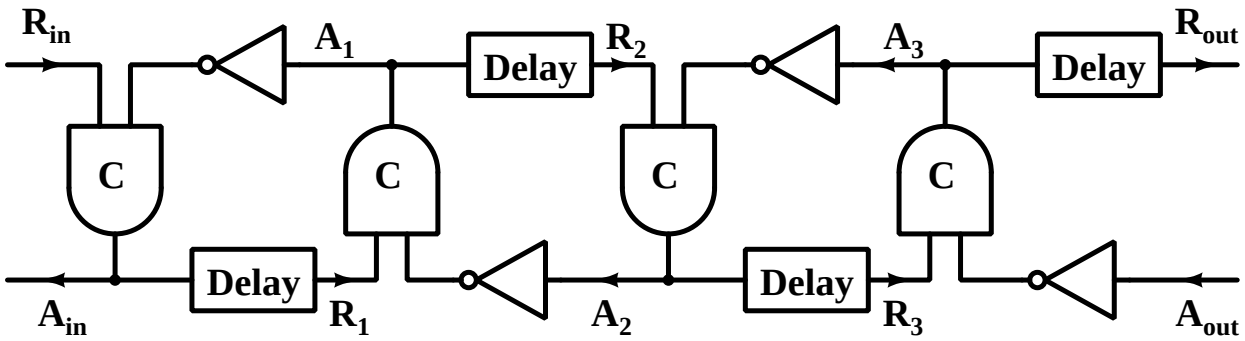


- When all switches are in the up position, the upper inverter on the left and the output inverter will form a latch.
- When all switches are in the down position, the lower inverter on the left and the output inverter will form a latch.

In either case the data present at the input when this condition occurred will be latched and the output will be isolated from the input.

Two Phase Pipeline

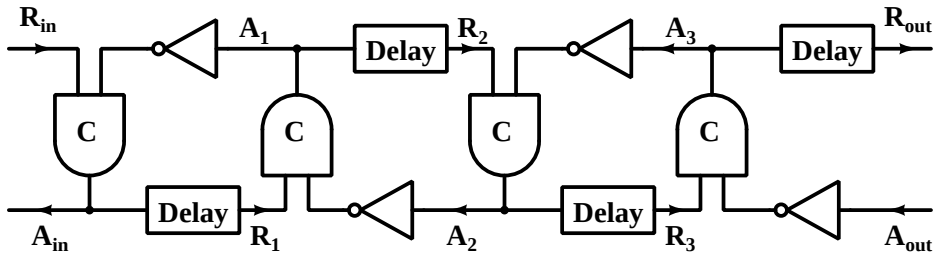
Assume initially that all Req and Ack signals are ‘0’ and the FIFO is empty.



- When the external source of data on the left has data available, it will toggle the Rin line.
- Now Rin = ‘1’, A1 = ‘0’. Both inputs of first C element are ‘1’, so output Ain = ‘1’.
- This latches the data and acknowledges it to the source on the left. This stage is now full.

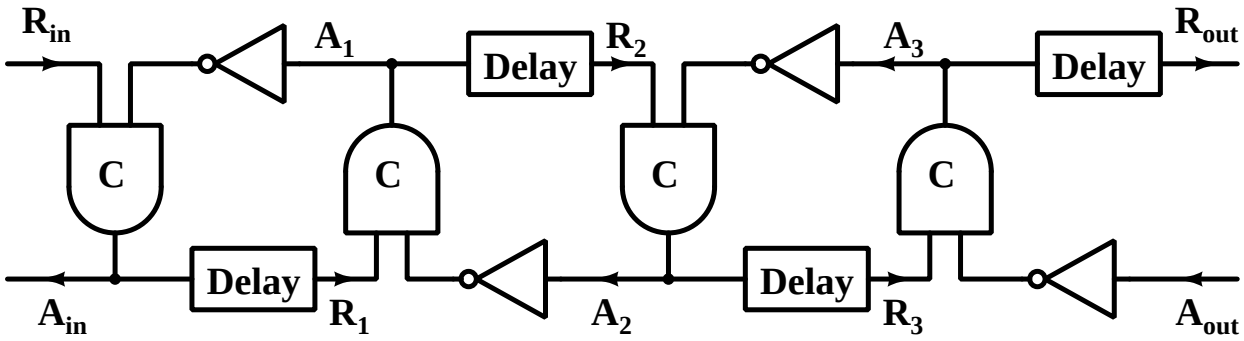
Two Phase Pipeline

- If another request comes when the first stage is full, (Rin is toggled now to 0, while A1 is still 0), the two inputs to the left most C element are unequal, so it holds its output.



Since no acknowledge event is generated, the previous stage will hold its data and Rin lines.

Two Phase Pipeline



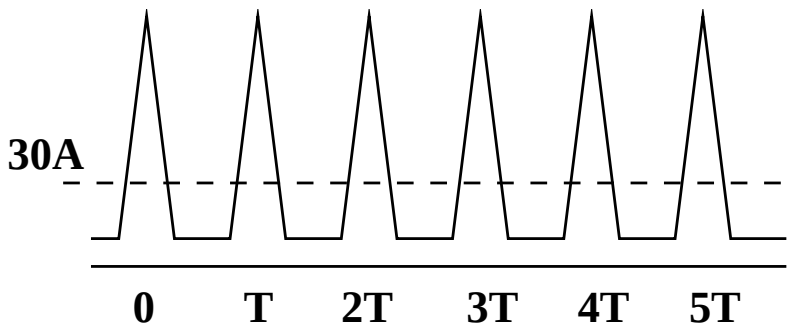
- After generation of Ain, it appears as request R1 to the next stage with some delay.
- R1 = '1', A2 = '0'. Both inputs of C element are '1', so output A1 = '1'.
- Thus the second stage has accepted data and the first stage is empty again.

High Frequency Models

- Accurate device models are difficult to develop because of non-equilibrium effects.
- At these high frequencies, even a wire behaves like a strip line and becomes difficult to model.
- Isolation between circuits becomes difficult because of substrate coupling effects.

Power Distribution

- CMOS circuits draw current only when they switch.
- This happens simultaneously in millions of gates when new inputs arrive at the clock transitions.
- This results in a huge peak to average ratio.



This huge value of peak I and $\frac{dI}{dt}$ results in large RI and L$\frac{dI}{dt}$ drops. This can even result in a total power rail collapse.

Part IV

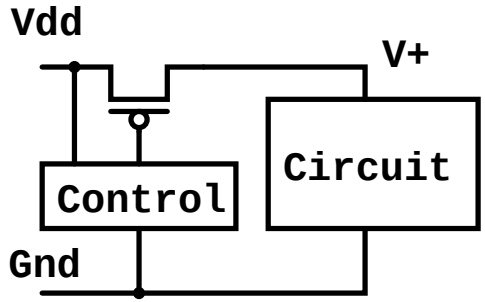
Low Power Design

Power Reduction methods

- Turn Off Modules which are not in use
- Reduce Voltage Swing on High Capacitance Lines
- Use power efficient codes

Turning off Power

If a part of the VLSI is not active, we can turn off its power supply.



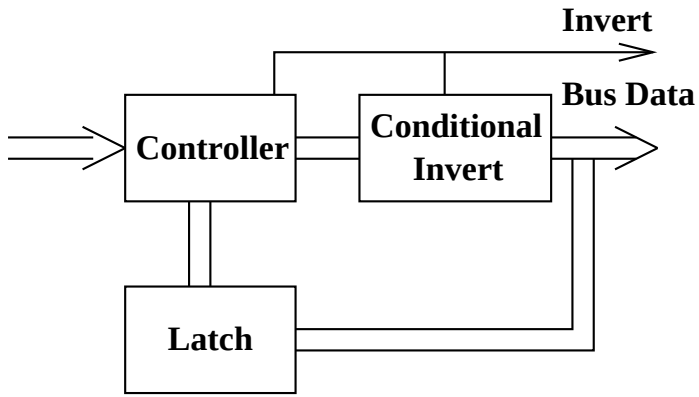
- Turning off power requires low dropout power switches.
- A look ahead circuit is required, which will decide when to turn the power on again. (Turning the power on would introduce a delay).
- State information in latches could be lost.

Stopping the clock

- Instead of turning the power off, we could just stop the clock.
- Stopping the clock saves dynamic power - which is a major component of total power consumption.
- However, introducing logic in the clock path to control it will add to clock jitter.
- Enable type Flip-Flops could be used instead of gating the clock.

Using Power Efficient Codes

Bus Invert coding:



- Controller compares the prev. output with the current one.
- If inverting the bus would cause fewer toggles, it generates a bus invert signal.
- The invert block (containing XOR gates and buffers) inverts the bus if necessary.
- The bus carries an extra wire to indicate whether the bus has been inverted.

Using Power Efficient Codes

- If the pattern of signals on a bus is not random, one can use special codes for these buses to reduce power. For example, address buses generally carry sequential values. Use of Gray code would ensure that there is only one toggle per cycle on this bus.
- In future technologies where leakage power is expected to be high, the inputs of idle circuits can be held to bit patterns which minimize leakage power.
- Similarly, magnitude comparators need not load less significant bits if MSB's are different.

Part V

Mixed Signal Design

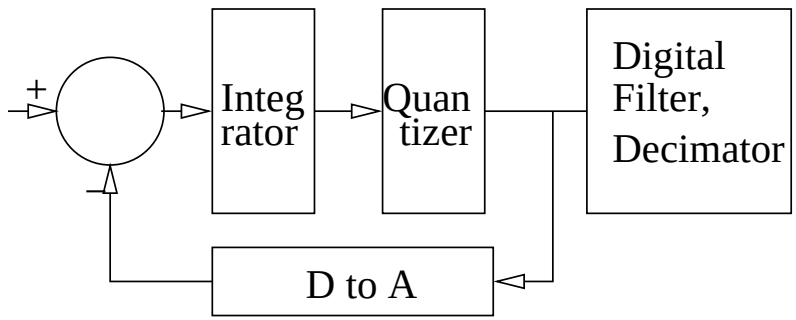
Mixed Signal Design

- Most large VLSIs will include some analog circuitry.
- Unfortunately, requirements of analog circuits are different from those of digital circuits.
- Since most of the circuit is digital, the technology is optimized for digital applications.
- This means that analog circuits need to be designed with non-optimal technology and device structures.

Handling Digital and analog circuits together

- Power supply and ground connections for analog and digital parts of the design are kept separate.
- Analog circuits are laid out in a separate area and isolated using grounded moats.
- Typical channel lengths in the analog portion of the design are much longer than those used in the digital portion.

A Sigma Delta Modulator



Part VI

Test and Verification

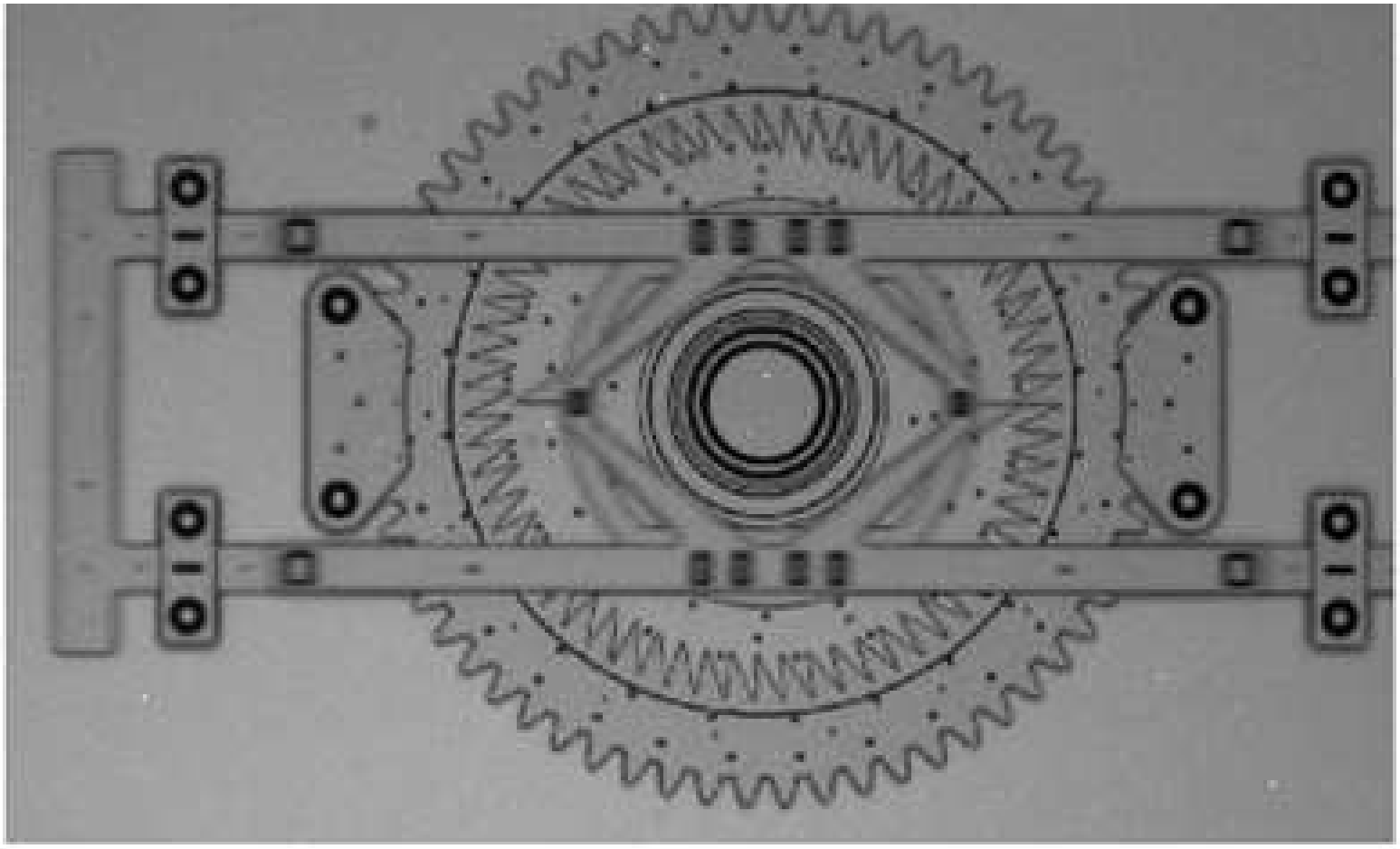
Test and Verification

- As die sizes increase, the number of pads increases linearly with chip dimensions.
- The die area and therefore the number of devices on the chip increases as the square of the chip dimension.
- As a result, it is difficult to reach internal nodes for testing.
- Test signals can be carried in and node values brought out using serial communication (JTAG).
- This, however, increases test time.
- We need to incorporate self test circuits on chip.

Part VII

What does the future hold?

What does the future hold?



What does the future hold?

- Greater complexity: MEMS, motors and the like!
- Different devices: MOS and Bipolar + optical?
- (Hopefully) More powerful design aids
- Error tolerant designs
- Reconfigurable design/Platform based design
- Even more expectations by the market!!

Impact of Microelectronics

Communications and personal computations are the two fields where most achievements have been made.

Now these two will become enabling agents for other fields to be revolutionized.

The next two ‘frontiers’ are likely to be:

- Telemedicine and Personal Health
- Finance and Trade

Other fields will continue to see an ‘evolutionary’ improvement.