MODULE- 2

ASIC Library Design: Logical effort: predicting delay, logical area and logical efficiency, logical paths, multi stage cells, optimum delay, optimum no. of stages, library cell design. (3.3 & 3.4)

Low-Level Design Entry: Schematic Entry: Hierarchical design. The cell library, Names, Schematic, Icons & Symbols, Nets, schematic entry for ASIC'S, connections, vectored instances and buses, Edit in place, attributes, Netlist screener, Schematic-Entry tools Back annotation. (9.1)

by,

Dr.P.Vimala
Department of ECE
Dayananda Sagar College of Engineering
Bangalore

Figures from M.J.S .Smith, - "**Application - Specific Integrated Circuits**"

Logical Effort

- Explore a delay model- basis for time constant analysis
- All nonideal component of delay, tq, includes

  (1)delay due to internal parasitic capacitance;

  (2)the time for the input to reach the switching threshold of the cell;

  (3)the dependence of the delay on the slew rate of the input waveform.

- With these assumptions we can express the delay as follows:

$$t_{PD} = R\,(C_{out} + C_p) + t_q$$

- Logic cell by a scaling factor, s
  - pull resistance R will decrease to R / s
  - parasitic capacitance Cp will increase to sCp
  - assume that tq scales linearly with s for all cells (stq)

Total cell delay $t_{PD} = (R/s)\cdot(C_{out} + sC_p) + st_q$

- Rewrite above Eq. using the input capacitance of the scaled logic cell $C_{in} = sC$,

$$t_{PD} = RC\frac{C_{out}}{C_{in}} + RC_p + st_q$$

- Finally we normalize the delay using the time constant formed from the pull resistance Rinv and the input capacitance Cinv of a minimum-size inverter: $d = \frac{(RC)(C_{out}/C_{in}) + RC_p + st_q}{t}$ where, $t = R_{inv}C_{inv}$
- The delay equation is the sum of three terms with special names as follows: $d = f + p + q$

  delay = effort delay + parasitic delay + nonideal delay

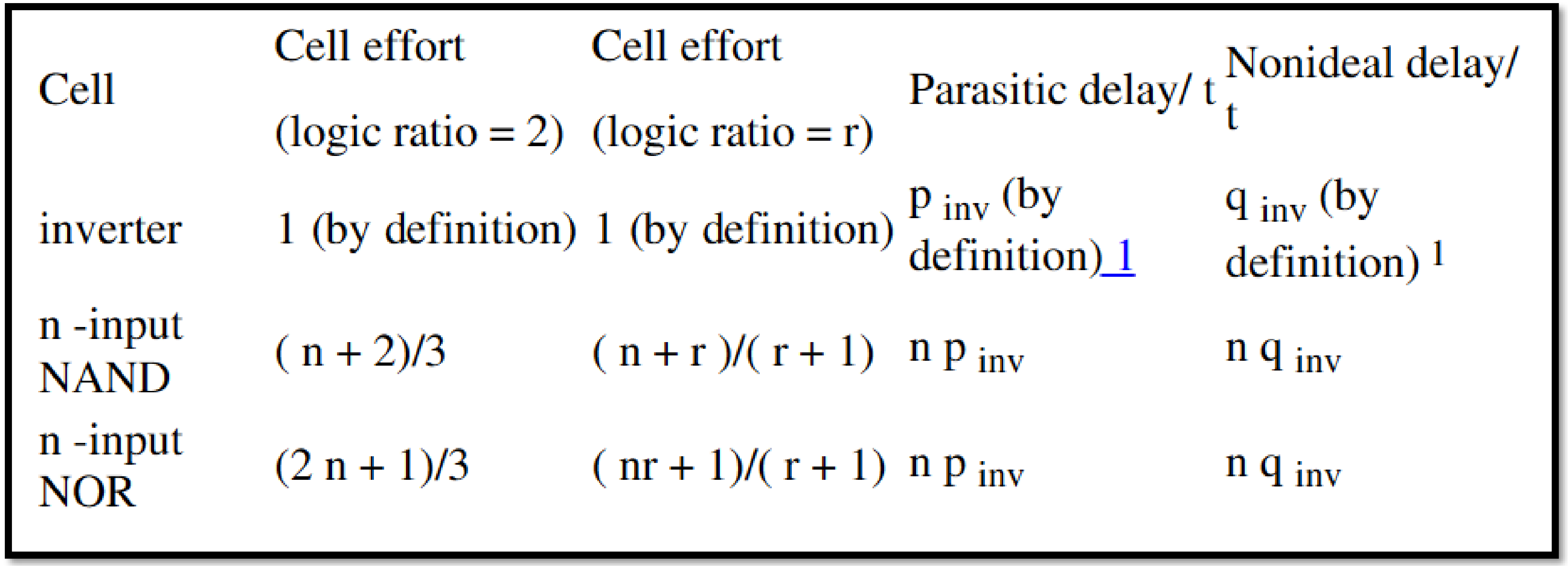
  - Effort delay , $f = gh$

    where, g-logical effort (g = RC/ t) & h-electrical effort ($h = C_{out}/C_{in}$)
  - Parasitic delay, $p = RC_p/t$
  - Nonideal delay, $q = st_q/t$
- delay = (logical effort X electrical effort)+ parasitic delay + nonideal delay

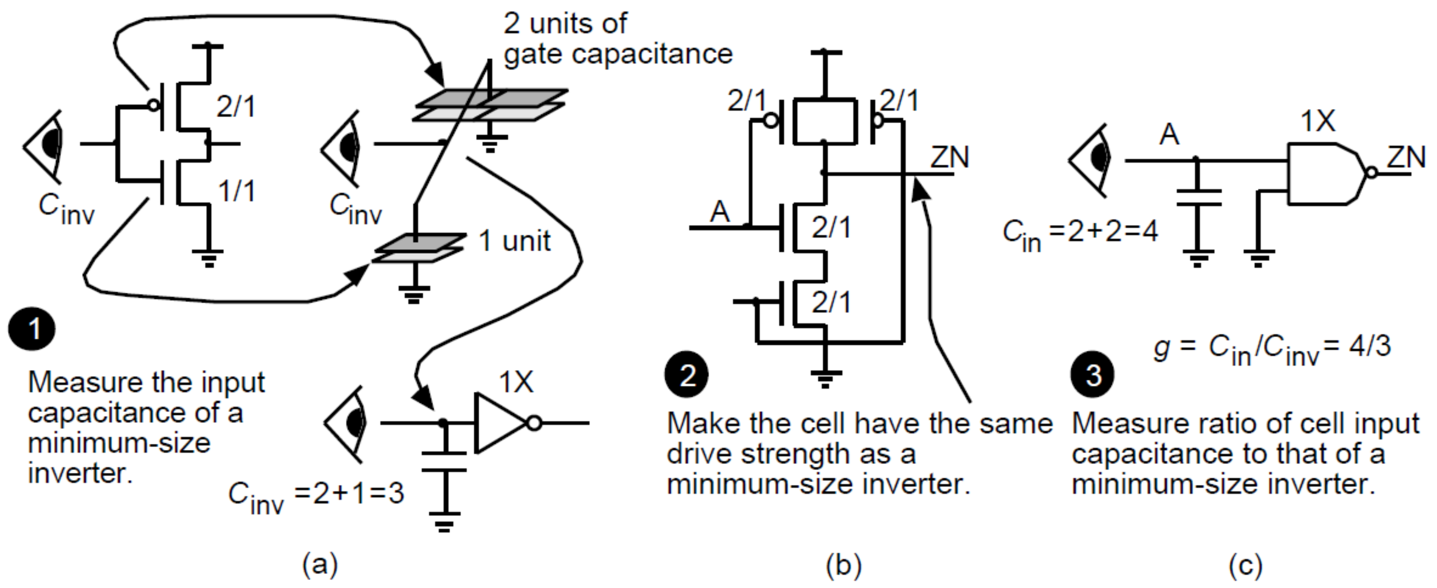
logical effort , g = RC/ t

- R and C will change as we scale a logic cell, but the RC product stays the same
- Logical effort is independent of the size of a logic cell
- We can find logical effort by scaling a logic cell to have the same drive as a 1X minimum-size inverter
- Then the logical effort, g, is the ratio of the input capacitance, Cin, of the 1X logic cell to Cinv.

The logical effort of a cell (g) = Cin/ Cinv



Cell	Cell effort (logic ratio = 2)	Cell effort (logic ratio = r)	Parasitic delay/ t	Nonideal delay/ t
inverter	1 (by definition)	1 (by definition)	p_{inv} (by definition) 1	q_{inv} (by definition) 1
n -input NAND	$(n + 2)/3$	$(n + r)/(r + 1)$	$n\,p_{inv}$	$n\,q_{inv}$
n -input NOR	$(2n + 1)/3$	$(nr + 1)/(r + 1)$	$n\,p_{inv}$	$n\,q_{inv}$


2 units of gate capacitance
2/1
1/1
C_{inv}
C_{inv}
1 unit
1
Measure the input capacitance of a minimum-size inverter.
1X
C_{inv} =2+1=3
(a)
2/1
2/1
ZN
A
2/1
2/1
2
Make the cell have the same drive strength as a minimum-size inverter.
(b)
A
1X
ZN
C_{in} =2+2=4
$g = C_{in}/C_{inv} = 4/3$
3
Measure ratio of cell input capacitance to that of a minimum-size inverter.
(c)

- For a two-input NAND cell, the logical effort, g=4/3

(a) Find the input capacitance, Cinv, looking into the input of a minimum-size inverter in terms of the gate capacitance of a minimum-size device

(b) Size a logic cell to have the same drive strength as a minimum-size inverter (assuming a logic ratio of 2). The input capacitance looking into one of the logic-cell terminals is then Cin

(c) The logical effort of a cell is Cin/ Cinv

We can size a logic cell using these basic rules:

- Any string of transistors connected between a power supply and the output in a cell with 1X drive should have the same resistance as the n -channel transistor in a 1X inverter.
- A transistor with shape factor W 1 /L 1 has a resistance proportional to L 1 /W 1 (so the larger W 1 is, the smaller the resistance).
- Two transistors in parallel with shape factors W 1 /L 1 and W 2 /L 2 are equivalent to a single transistor (W 1 /L 1 + W 2 /L 2)/1. For example, a 2/1 in parallel with a 3/1 is a 5/1.
- Two transistors, with shape factors W 1 /L 2 and W 2 /L 2 , in series are equivalent to a single 1/(L 1 /W 1 + L 2 /W 2) transistor.

Predicting Delay

Example: Let us predict the delay of a three-input NOR logic cell with 2X drive, driving a net with a fan out of four, with a total load capacitance of 0.3 pF.

Where $p = 3\, p_{inv}$ and $q = 3\, q_{inv}$

For C5 technology, Cinv and is approximately 0.036 pF and qinv=1.7

We can calculate Cin from the fact that the input gate capacitance of a 1X drive, three-input NOR logic cell is equal to gCinv and for a 2X logic cell, Cin= 2 gCinv

$$gh = g \frac{C_{out}}{C_{in}} = \frac{g \cdot (0.3\ \text{pF})}{2\, g\, C_{inv}} = \frac{(0.3\ \text{pF})}{(2) \cdot (0.036\ \text{pF})}$$

The delav of the NOR logic cell. in units of t .

$$d = gh + p + q = \frac{0.3\ \mathbf{X}\ 10^{12}}{(2) \cdot (0.036\ \mathbf{X}\ 10^{12}\ )} + (3) \cdot (1) + (3) \cdot (1.7) \quad = 12.266667\ t$$

For C5 technology, t= 0.06ns

Absolute delay ,t_{PD} = 12.3 * 0.06 ns = 0.74 ns

In the C5 library,

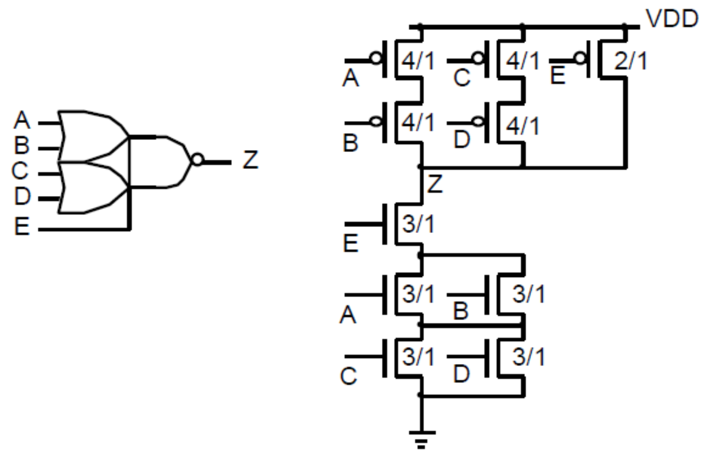
- The delay for a 2X drive, three-input NOR logic cell is t_{PD} = (0.03 + 0.72Cout + 0.60) ns
- With Cout=0.3pF, tPD = 0.03 + (0.72)·(0.3) + 0.60 = **0.846 ns** compared to our prediction of **0.74ns**

Error comes here due to,

- Inaccuracy in predicting the nonideal delay
- Logical effort gives us a method to examine relative delays and not accurately calculate absolute delays.
- More important is that logical effort gives us an insight into why logic has the delay.

Logical Area and Logical Efficiency

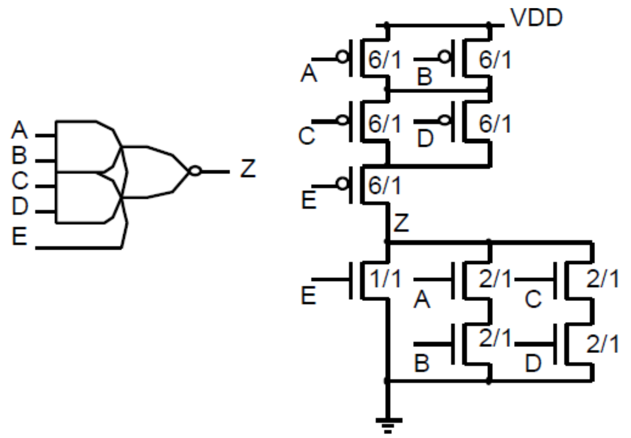
Example -1 (OAI221)



An **OAI221** logic cell

- Logical-effort vector g=(7/3, 7/3,5/3)
- The logical area is 33 logical squares

Example -2 (AOI221)



An **AOI221** logic cell

- g=(8/3, 8/3, 7/3)
- Logical area is 39 logical squares
- Less logically efficient than OAI221

Logical Paths

- We can extend the idea of logical effort to a chain of logic cells or logical path .
- Consider the logic path when we use a minimum-size inverter $(g_0 = 1, p_0 = 1, q_0 = 1.7)$ to drive one input of a 2X drive, three-input NOR logic cell with

  $g_1 = (nr + 1)/(r + 1), p_1 = 3, q_1 = 3,$ and a load equal to four standard loads.

  If the logic ratio is r = 1.5,then g1 = 5.5/2.5 = 2.2.
- The delay of the inverter is,

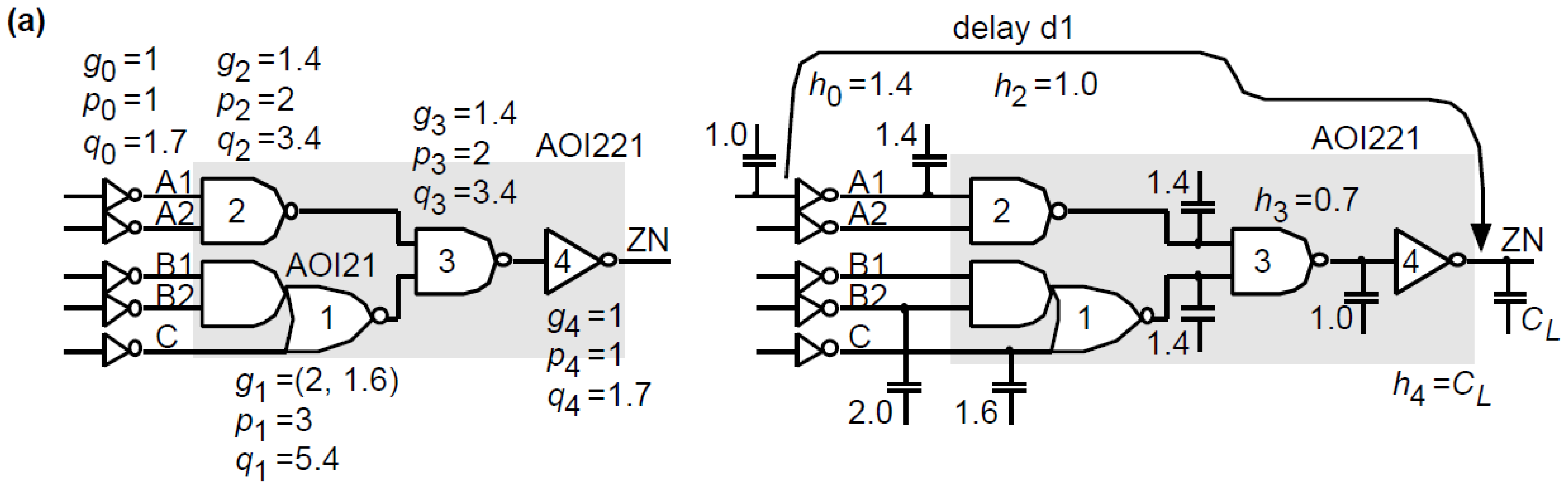
$$\begin{aligned} d &= g_0 h_0 + p_0 + q_0 = (1) \cdot (2g_1) \cdot (C_{inv} / C_{inv}) + 1 + 1.7 \\ &= (1)(2)(2.2) + 1 + 1.7 \\ &= 7.1 . \end{aligned}$$

- In this, 4.4 t to the loading of the NOR logic cell input capacitance.
- The delay of the NOR logic cell is, $d_1 = g_1 h_1 + p_1 + q_1 = 12.3,$
- total delay 7.1 + 12.3 = 19.4 and the absolute delay is (19.4)(0.06 ns) = 1.164 ns, or about 1.2 ns.

$$\textbf{path delay } D = \sum_{i \in \text{path}} g_i h_i + \sum_{i \in \text{path}} (p_i + q_i)$$

Multistage Cells

Consider the function AOI221

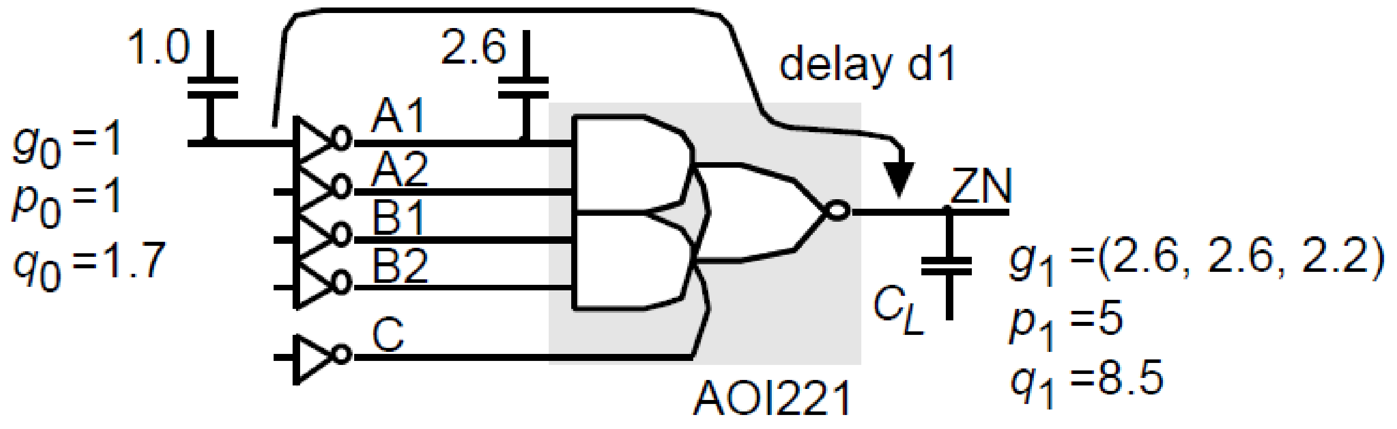


$$d1=(g_0h_0+p_0+q_0)+(g_2h_2+p_2+q_2)+(g_3h_3+p_3+q_3)+(g_4h_4+p_4+q_4)$$

$$=(1\times 1.4+1+1.7)+(1.4\times 1+2+3.4)+(1.4\times 0.7+2+3.4)+(1\times C_L+1+1.7)=20+C_L$$

$$g_0=g_4=g(\text{NOT})=1,$$
$$g_1=g(\text{AOI21})=(2,(2r+1)/(r+1))=(2,4/2.5)=(2,1.6)$$
$$g_2=g_3=g(\text{NAND2})=(r+2)/(r+1)=(3.5)/(2.5)=1.4.$$

$$d1=(1)\cdot(1.4)+1+1.7+(1.4)\cdot(1)+2+3.4$$
$$+(1.4)\cdot(0.7)+2+3.4+(1)\cdot C_L+1+1.7$$
$$=(20+C_L).$$



$d1 = (1 \times 2.6 + 1 + 1.7) \quad + (1 \times C_L + 5 + 8.5) = 18.8 + \quad C_L \longleftarrow$

$$\boxed{\begin{aligned} g\,(\mathrm{AOI221}) &= ((3r + 2)/(r + 1), (3r + 2)/(r + 1), (3r + 1)/(r + 1)) \\ &= (6.5/2.5, 6.5/2.5, 5.5/2.5) \\ &= (2.6, 2.6, 2.2)\ . \end{aligned}}$$

$$\boxed{\begin{aligned} d1 &= ((1)\cdot(2.6) + 1 + 1.7 + (1)\cdot C_L + 5 + 8.5\ ) \\ &= 18.8 + C_L\ . \end{aligned}}$$

- The single-stage delay is very close to the delay for the multistage version of this logic cell.
- In some ASIC libraries the AOI221 is implemented as a multistage logic cell instead of using a single stage.

Can we make the multistage logic cell as faster???

Optimum Delay

- The **path logical effort, G** is the product of logical efforts on a path

$$G = \prod_{i \in \text{path}} g_i$$

- The **path electrical effort, H** is the product of the electrical efforts on the path,

$$H = \prod_{i \in \text{path}} h_i \quad \frac{C_{out}}{C_{in}}$$

where, Cout is the last output capacitance on the path (the load)
Cin is the first input capacitance on the path.

- The **path effort, F** is the product of the path electrical effort and logical efforts

$$F = GH$$

- The optimum effort delay for each stage is found by minimizing the path delay D by varying the electrical efforts of each stage hi, while keeping H fixed.

$$\hat{f}_i = g_i h_i \qquad = F^{1/N}$$

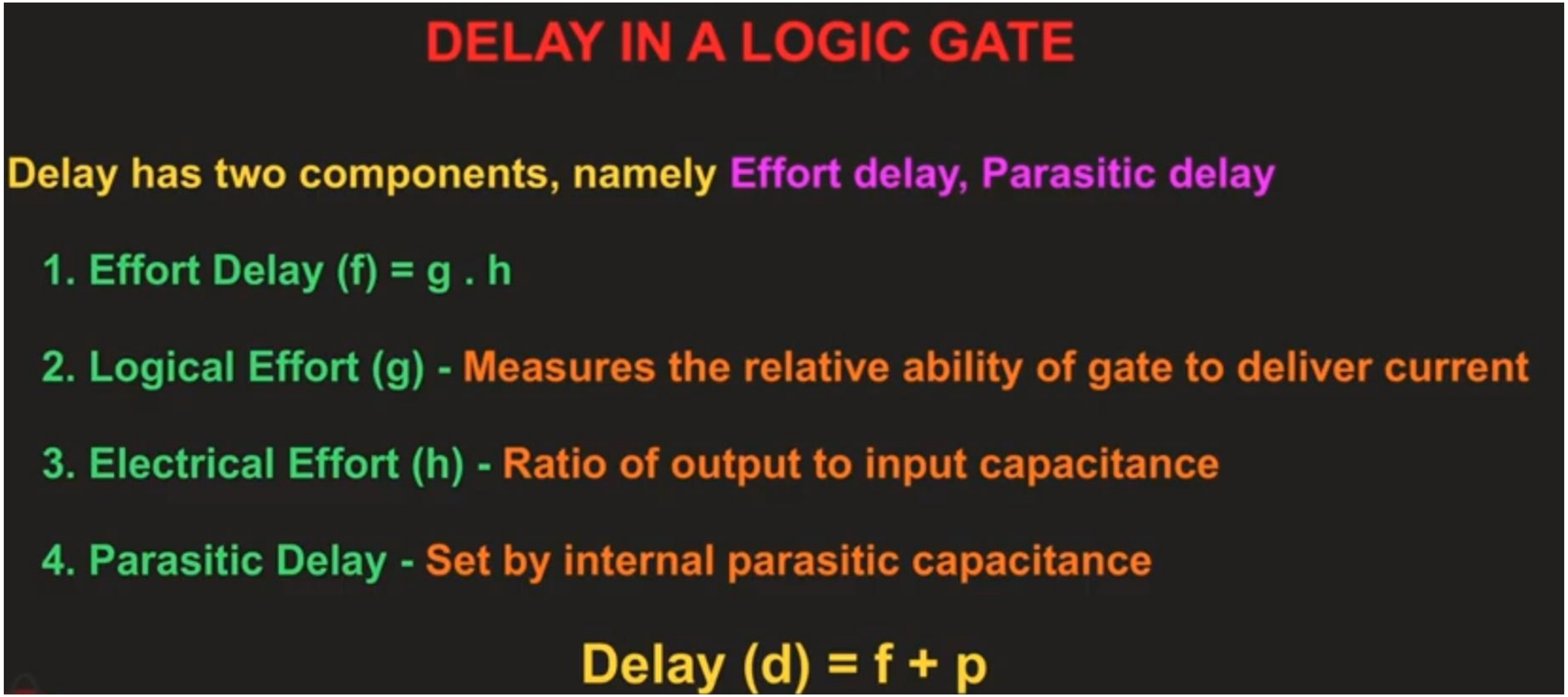
- The optimum effort delay is achieved when each stage operates with equal effort,

$$\hat{D} = NF^{1/N} \qquad = N(GH)^{1/N} + P + Q$$

- The optimum path delay is then,

$$P + Q = \sum_{i \in \text{path}} p_i + h_i$$

where P + Q is the sum of path parasitic delay and nonideal delay

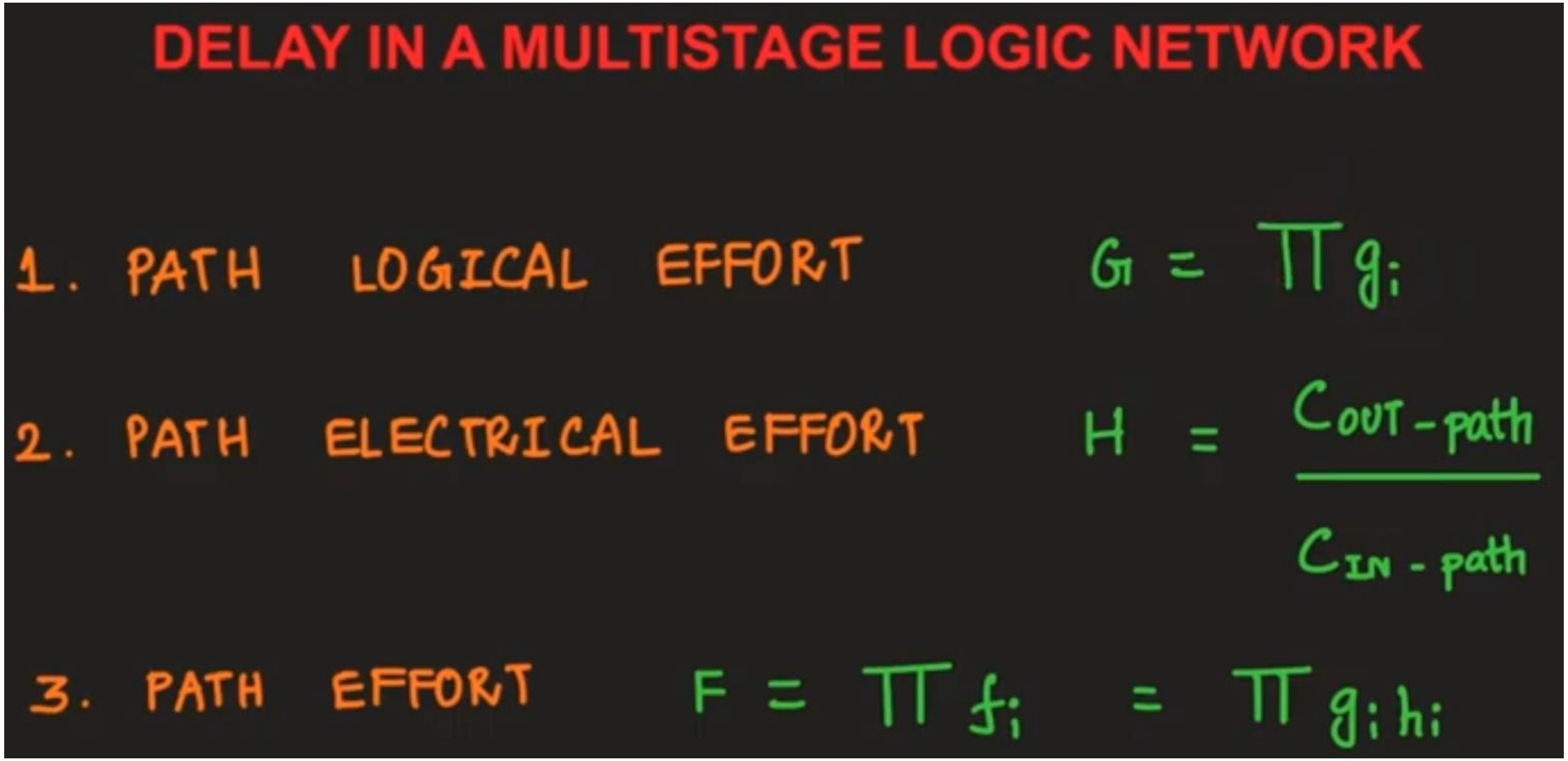


DELAY IN A LOGIC GATE

Delay has two components, namely Effort delay, Parasitic delay

1. Effort Delay (f) = g . h
2. Logical Effort (g) - Measures the relative ability of gate to deliver current
3. Electrical Effort (h) - Ratio of output to input capacitance
4. Parasitic Delay - Set by internal parasitic capacitance

Delay (d) = f + p



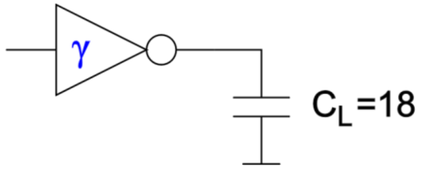
DELAY IN A MULTISTAGE LOGIC NETWORK

1. PATH LOGICAL EFFORT $G = \prod g_i$

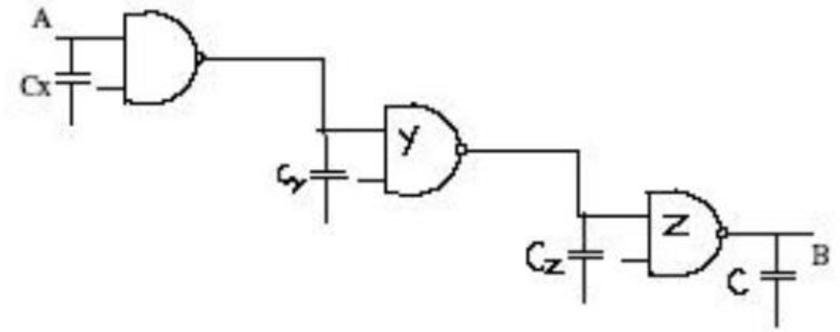
2. PATH ELECTRICAL EFFORT $H = \frac{C_{OUT\text{-}path}}{C_{IN\text{-}path}}$

3. PATH EFFORT $F = \prod f_i = \prod g_i h_i$

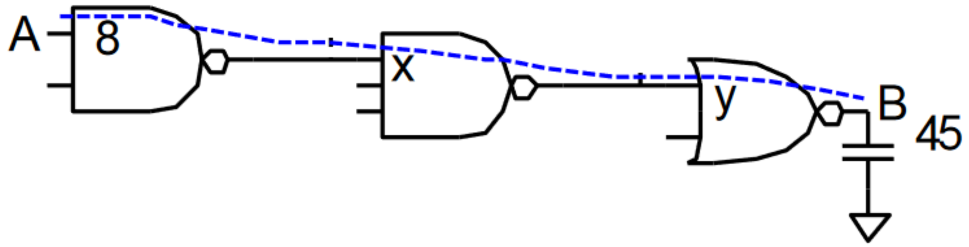
1) Compute the delay of the reference inverter



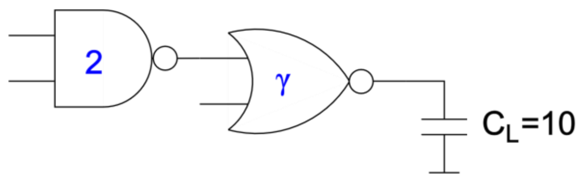
2) Consider the path from A to B involving three two input NAND gates as in fig 23.22. The input capacitance of first gate is C and the load capacitance is also C . Find the least delay in this path and how should the transistors be sized to achieve least delay?



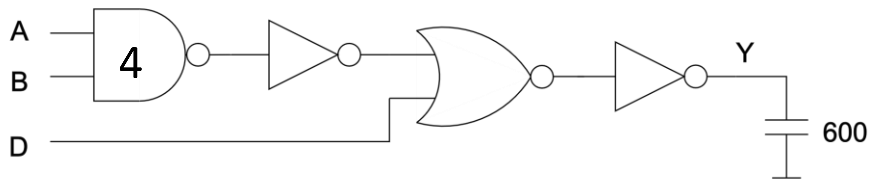
3) Find the gate sizes x and y for least delay from A to B



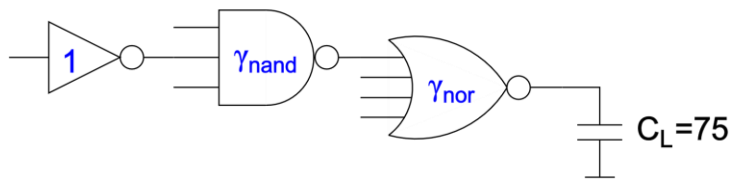
4) Compute minimum path delay



5) Compute the optimum delay for the path A to Y. Also find the gate sizes for cells available between A to Y.



6) Minimize the delay of the 3-stage path using the method of logical effort.



Optimum Number of Stages

- Suppose we have a chain of N inverters each with equal stage effort, f = gh
- Neglecting parasitic and nonideal delay,

  The total path delay is Nf = Ngh = Nh ,since g = 1 for an inverter.

- Suppose we need to drive a path electrical effort H ;

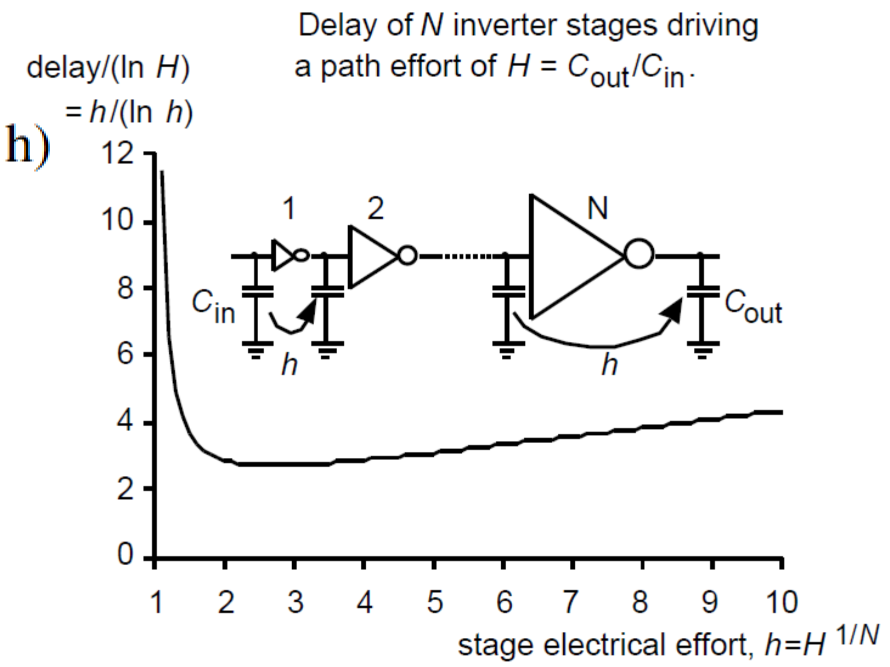
  $h^N = H$, or $N \ln h = \ln H$

  $N = \ln H / \ln h$

  $Nh = h * \ln H / \ln h$

- Since ln H is fixed, we can only vary h /ln (h)

h	h/(ln h)
1.5	3.7
2	2.9
2.7	2.7
3	2.7
4	2.9
5	3.1
10	4.3



- Diagram shows, How to minimize the delay regardless of area or power

Library-Cell Design

- A big problem in library design is dealing with design rules
  - The **optimum cell layout for each process generation changes** because of the **different design rules for each ASIC vendors** process
- **All vendors to agree on a common set of design rules.**
- Sometimes we can **waive** design rules (very large customer)
- **Symbolic layout**, **sticks** or **logs** can decrease the library design time.
- Symbolic layout performed in two ways: **1)graphics 2) a text layout language.**

Graphics-Shapes are represented by simple lines or rectangles, known as sticks or logs.

Text layout language - similar to a programming language such as C, that directs a program to assemble layout.

- Mapping symbolic layout uses 10–20 percent more area than hand-crafted layout.

Design Entry

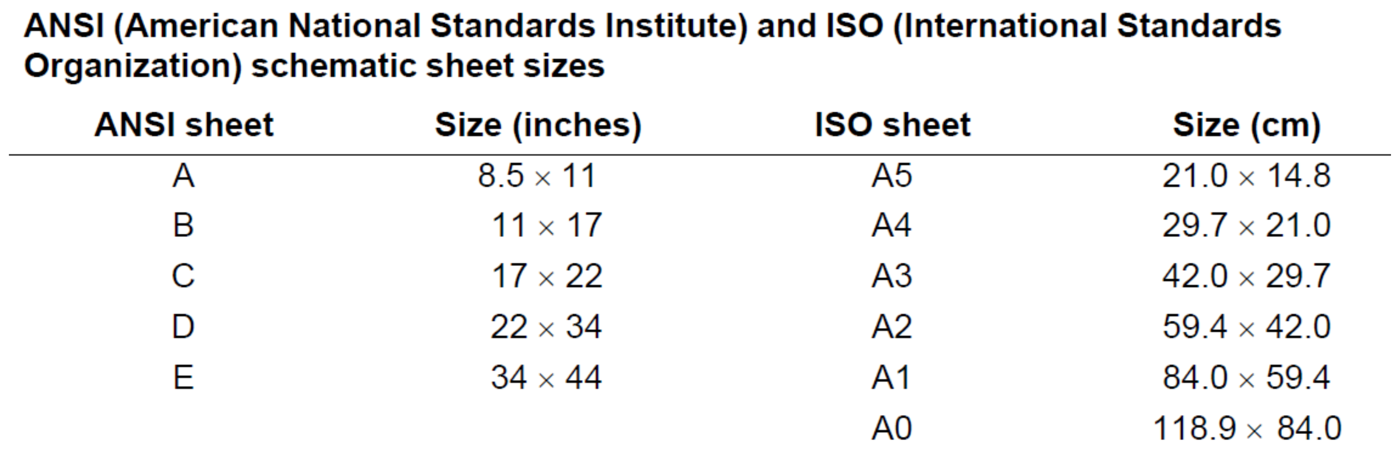
- The purpose of design entry is **to describe a microelectronic system to a set of electronic-design automation (EDA) tools.**
- A circuit schematic describes an ASIC in the **same way an architects plan describes a building**.

Schematic Entry

- Graphical design entry
- Transforms an idea to a computer file
- An “old” method that periodically regains popularity
- Schematic sheets

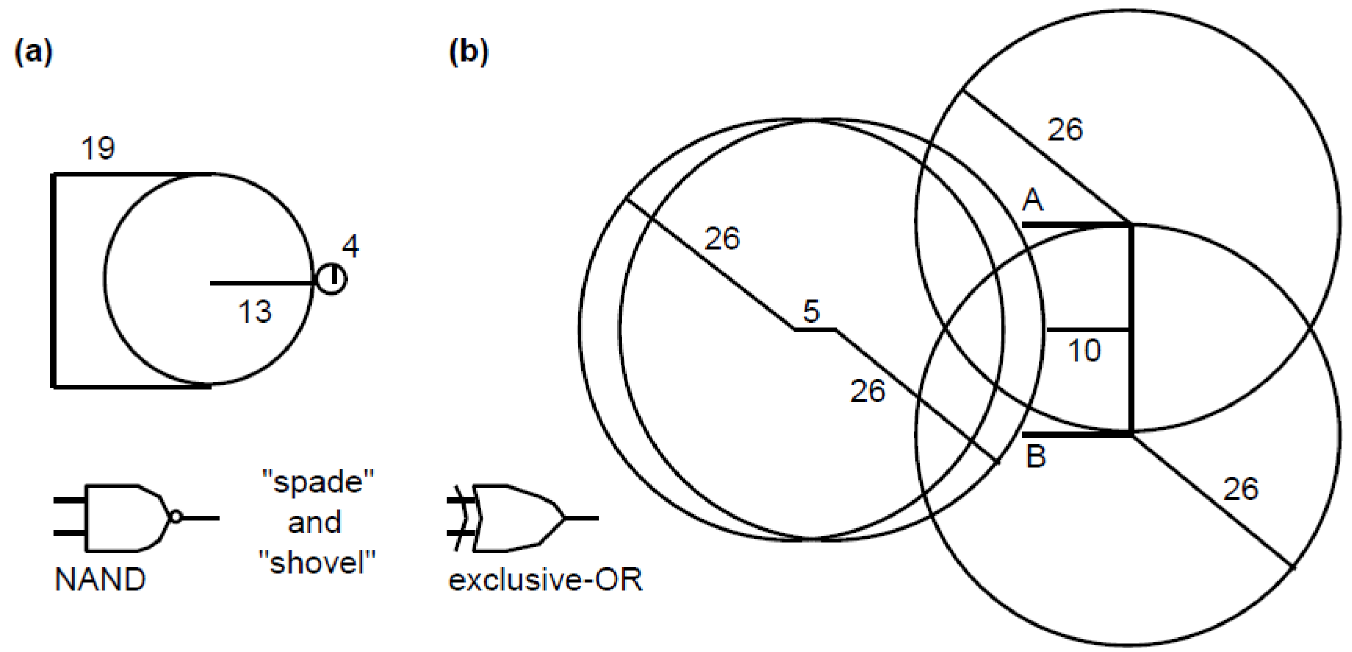
Schematic boxes containing :

- Name and number of schematic page
- Designer
- Date of drawing
- List of any modification



ANSI (American National Standards Institute) and ISO (International Standards Organization) schematic sheet sizes

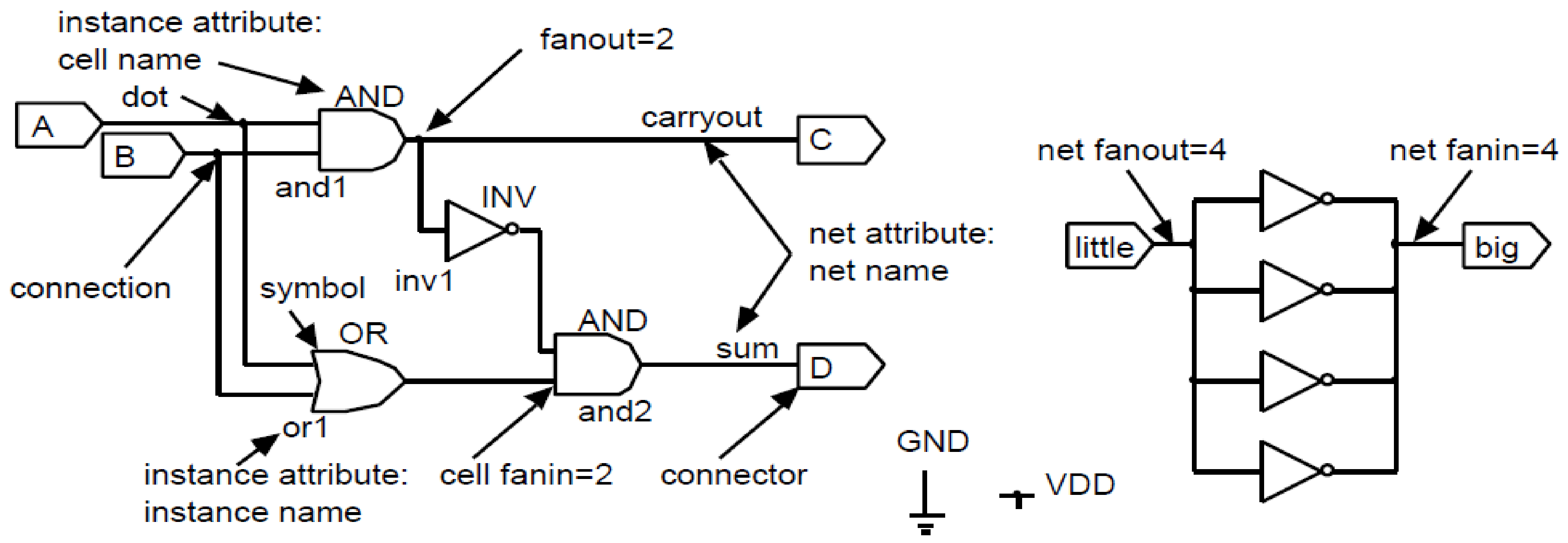
ANSI sheet	Size (inches)	ISO sheet	Size (cm)
A	8.5 × 11	A5	21.0 × 14.8
B	11 × 17	A4	29.7 × 21.0
C	17 × 22	A3	42.0 × 29.7
D	22 × 34	A2	59.4 × 42.0
E	34 × 44	A1	84.0 × 59.4
		A0	118.9 × 84.0



IEEE-recommended dimensions and their construction for logic-gate symbols

(a) NAND gate

(b) exclusive-OR gate (an OR gate is a subset)



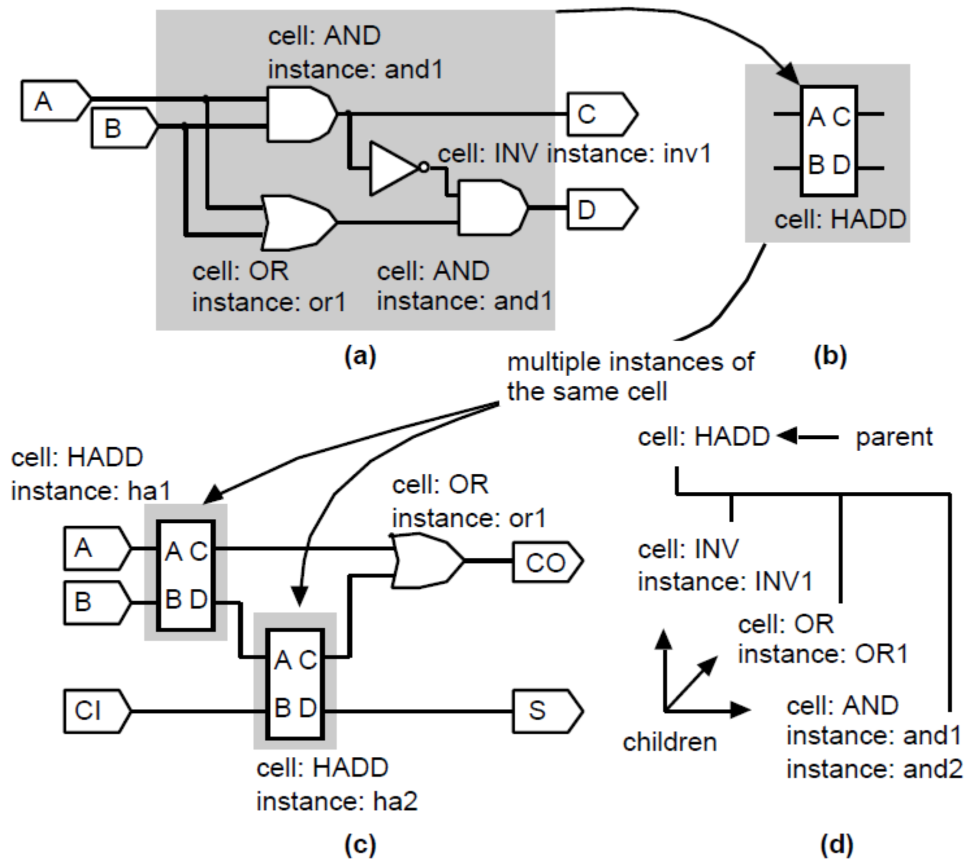
Terms used in circuit schematics

Schematic entry tools for ASIC design are similar to PCB design.

- Object used in PCB: Component / device
  Objects used in ASIC: Logic Gates
- Draw every component on schematic sheets for PCB
  Drawing every component on an ASIC schematic is impractical

Hierarchical Design

- Hierarchy reduces the size and complexity of a schematic.
- To clarify the relationship between different levels of hierarchy we say that a sub schematic (an office) is a child of the parent schematic (the floor containing offices).
- An electrical schematic can contain sub schematics. The sub schematic, in turn, may contain other sub schematics.
- Alternative to hierarchical design is to draw all of the ASIC components on one giant schematic, with no hierarchy , in a flat design.
- For modern ASIC-flat design is impractical



Schematic example showing hierarchical design.
(a) The schematic of a half-adder, the sub schematic of cell HADD.
(b) A schematic symbol for the half adder.
(c) A schematic that uses the half-adder cell.
(d) The hierarchy of cell HADD.

The Cell Library

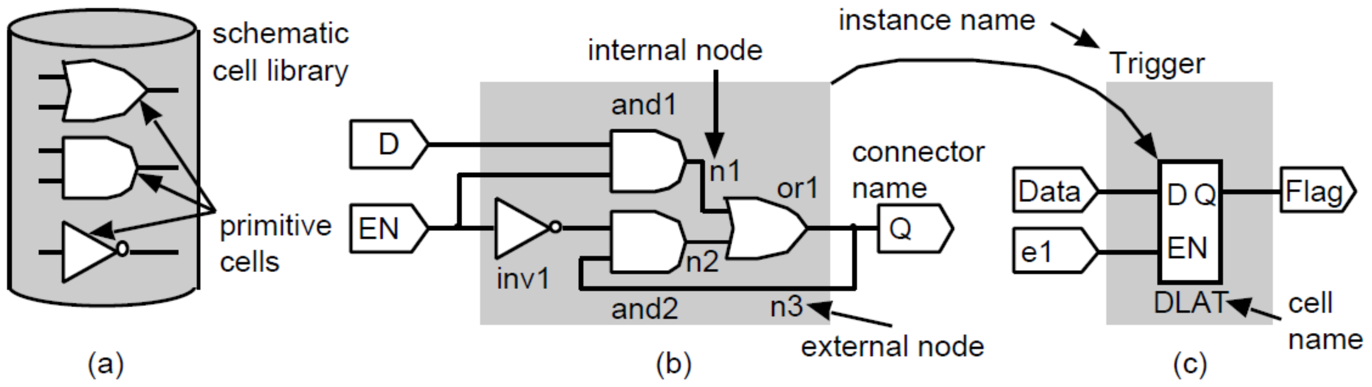
- Components in an ASIC schematic are chosen from **a library of cells**.
- Library elements for all types of ASICs are sometimes also known as **modules** .
- Library cells are equivalent to the offices in our office building.
- Different ASIC companies provide a schematic library
  1. no naming conventions. Example- two-input NAND gate in a Xilinx FPGA library does not have the same name as in an LSI Logic gate-array library.
  2. no standards for cell behavior. Example- a two-input MUX in an Actel library- input A is selected when the MUX select input S = '0'. In a VLSI Technology library operates in the reverse fashion, input B is selected when S = '0’. (**porting a design**)
- **Primitive cells:** logic cells that represent basic logic gates are known as primitive cells
- Cell is used to represent both primitive cells and sub schematic.
  - In a hierarchical ASIC design a cell may be a NAND gate, a flip-flop, a multiplier, or even a microprocessor
- Two types of MACROs for MGAs and Programmable ASICs: Hard Macro and Soft Macro
  - **Hard Macro**: includes placement information- change in position and orientation
  - **Soft Macro** : contains only connection information- Placement and wiring can vary
- Gates, FFs and so on are hard macros (the timing parameters for a soft macro can only determined after complete the place and route step)

Names

- Each of the cells, that we place on an ASIC schematic has a **cell name .**
- Each use of a cell is a different instance of that cell, and we give each instance a **unique instance name**.
- We represent each cell instance by a **picture or icon also known as symbol**.
- In the office building example, We will have icons to represent the ground floor and the plan for the other floors. We shall give them instance names: Floor Two , Floor Three , ... , Floor Ten .
- The library designer needs to work with libraries at the level the primitive cells are transistors.
- Three different office types is a corner office, the second office type has a window, and a third office type is without a window. We shall call these office cells: Corner Office ,Window Office , and No Window Office .

Schematic Icons and Symbols

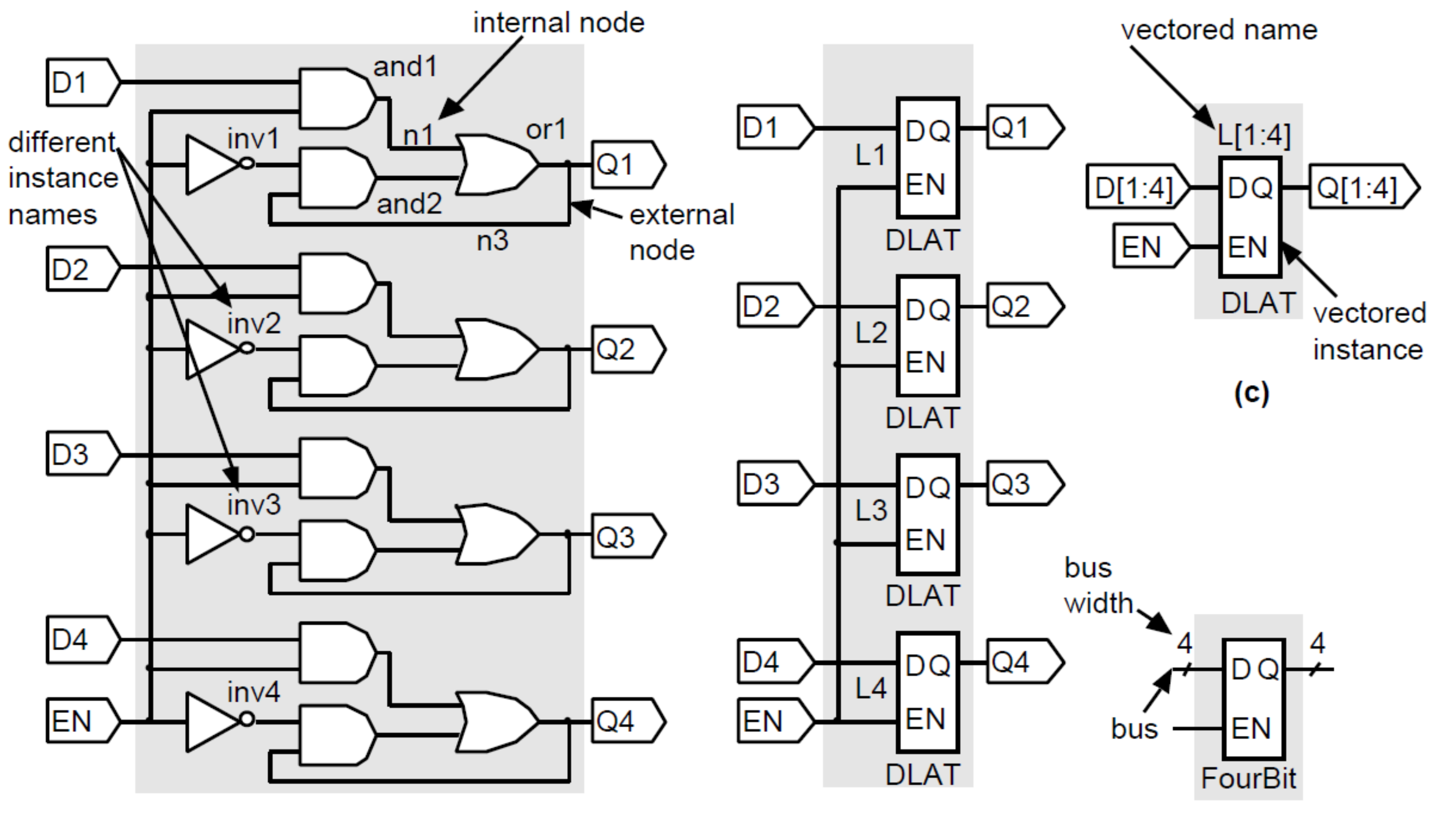
- schematic-entry programs allow the designer to draw special or custom icons.
- schematic-entry tool will usually create an icon automatically for a sub schematic used in higher- level schematic known as a **derived icon , or derived symbol** .
- The **external connections** of the sub schematic are automatically attached to the icon, usually a **rectangle.**



- Fig(c) shows what a derived icon for a cell, DLAT
- The sub schematic for DLAT is shown in Fig(b). We say that the inverter with the instance name inv1 in the sub schematic is a sub cell (or submodule) of the cell DLAT . Alternatively we say that cell instance inv1 is a child of the cell DLAT , and cell DLAT is a parent of cell instance inv1 .

A cell and its sub schematic. (a)A schematic library containing icons for the primitive cells. (b)A sub schematic for a cell, DLAT, showing the instance names for the primitive cells. (c)A symbol for cell DLAT.

Vectored instance



(a) (b) (d)

A 4-bit latch:

(a) drawn as a flat schematic from gate-level primitives

(b) drawn as four instances of the cell symbol DLAT

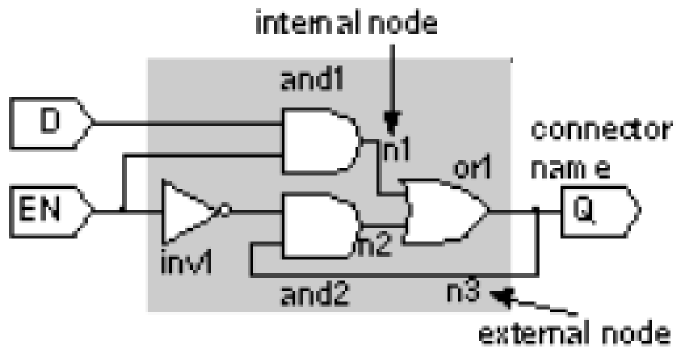
(c) drawn using a vectored instance of the DLAT cell symbol with cardinality of 4

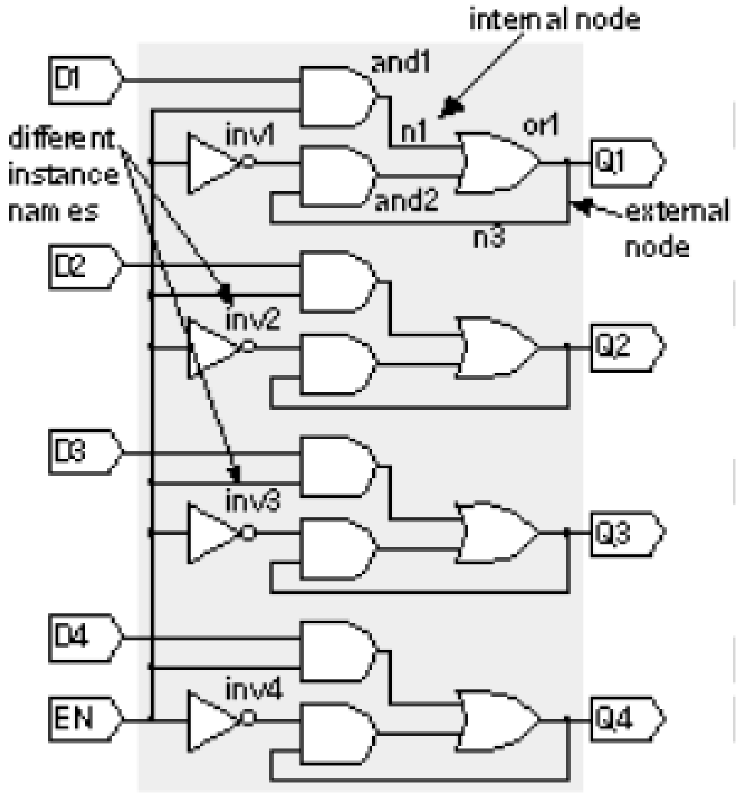
(d) drawn using a new cell symbol with cell name FourBit

Nets

Local /internal nets : Ex: n1

External Nets: Ex: D,EN and Q





When 4 copies of the circuit is placed in parent cell 4bit, 4 copies of net n1 created. All four nets are having the name as n1.

A special character (':' '/' '$' '#' for example) that is not allowed to appear in names is used as a delimiter to separate the net name from the cell instance name.

the four different nets labeled n1 might then become: . .

FourBit .L1:n1 FourBit .L2:n1 FourBit .L3:n1 FourBit .L4:n1

This naming is usually done automatically by the schematic-entry tool.

Schematic Entry for ASICs and PCBs

- A symbol on a schematic may represent a component, which may contains component parts.
- A component in PCBs is slightly different from an ASIC library cell.

Example: TTL gate (SN74LS00N), contains four 2-input NAND gates. SN74LS00N a component and each of the individual NAND gates inside is a component part.

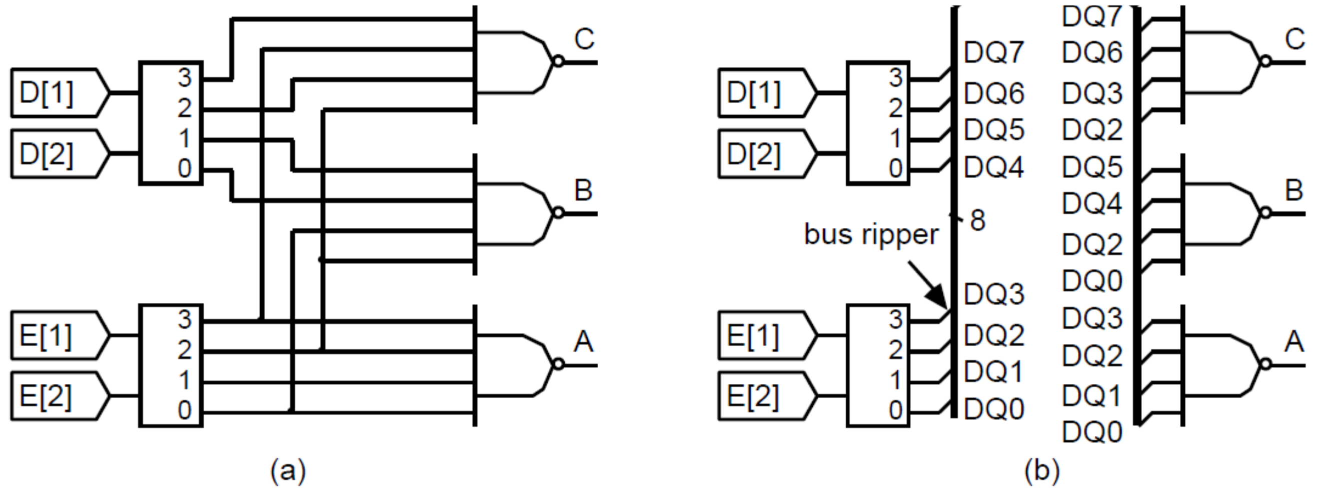
- PCBs usually contain packaged ASICs and other ICs that have pins that are soldered to a board.
- IC symbols have a pin number for each part in the package.

Example: TTL 74174 hex D flip-flop, contains six parts: six identical D flip-flops.

- The IC symbol representing this device has six Pin Number attribute entries for the D input corresponding to the six possible input pins. They are pins3, 4, 6, 11, 13, and 14.

Connections

- Cell instances have **terminals** that are the inputs and outputs of the cell.
- Terminals are also known as **pins , connectors , or signals** .
- The term **pin** is used in schematic entry and routing programs for PCB design.
- Electrical connections between cell instances **use wire segments or nets** . We can group closely related nets, such as the 32 bits of a 32-bit digital word, together into a bus or into buses.



An example of the use of a bus to simplify a schematic. (a)An address decoder without using a bus. (b)A bus with bus rippers (breakout) simplifies the schematic and reduces the possibility of making a mistake in creating and reading the schematic

- If signals on a bus are not closely related, we usually use the term **bundle or array** instead of bus.

Example: SCSI disk system: containing handshake and control signal with data bits

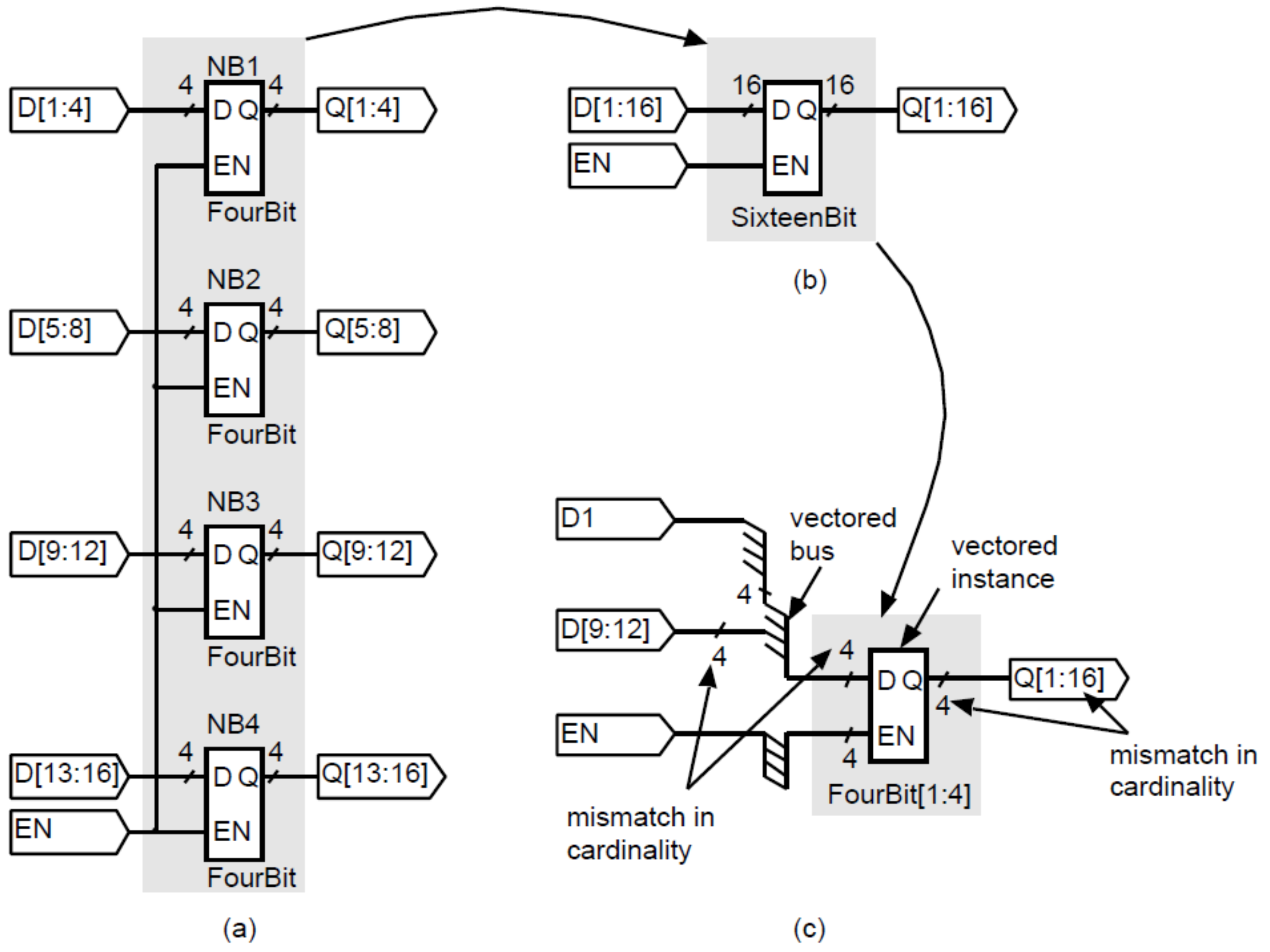
- If we need to access individual nets in a bus or bundle, we can use breakout known as **ripper, an EDIF term or extractor**
  - Electronic Design Interchange Format (EDIF) is a text format used to convey information between different EDA tools.
  - A bus with bus rippers simplifies the schematic and reduces the possibility of making a mistake in creating and reading the schematic

Example: If we need to access bits 0-7 of 32bits bus.

- If we need to rearrange bits on a bus , we can use **swizzle**

Example: to reorder the bits on an 8-bit bus (MSB becomes LSB and so on down to the LSB)

Vectored Instances and Buses



(a) shows a schematic for a 16-bit latch that uses multiple copies of the cell Four Bit . The buses are labeled with the appropriate bits.

(b) shows a new cell symbol for the 16-bit latch with 16-bit wide buses for the inputs, D, and outputs, Q

(c) shows an alternative representation of the 16-bit latch using a vectored instance of Four Bit with cardinality 4. Suppose we wish to make a connection to expressly one bit, D1 We also wish to make a connection to bits D9-D12, represented as D[9:12].

Problems arise when we have buses of buses because the numbers for the bus widths do not match. For this reason it is best to use the single-bus approach shown in (b) rather than the vectored-bus approach of (c).

Edit in Place

- we use the schematic-entry program **to edit the sub cell** NB1.L1 , (instance of DLAT inside NB1).

Example: To change the D latch to a D latch with a reset.

If the schematic editor supports edit-in-place , we can edit a cell instance directly. After we edit the cell, the program will update all the DLAT subcells in the cell that is currently loaded to reflect the changes that have been made.

To see how edit-in-place works,

- To change some of the offices on each floor from offices without windows to offices with windows. We select the cell instance FloorTwo (instance of cell Floor)
- edit the cell Floor- making changes to all of the floors that use cell name Floor
- cell instance FloorTwo - the second floor will become different from all the other floors.

If we want to edit just one instance of a cell,

- You must create a new cell with a new symbol and new unique cell name

Some tools have the ability to alias nets.

Aliasing creats a net name from the highest level in the design using dictionary.

This greatly speeds tracing of signals through a design containing many levels of hierarchy.

Attributes

- Attach an **attribute , or property**, which describes some aspect of the component, cell instance, net, or connector.
- **Each attribute has a name, and some attributes also have values**.
- The most common problems in working with schematics and netlists, especially when we try to exchange schematic information between different tools, are problems in naming.
- Problems of naming conventions including:
  - case sensitivity
  - name-collision resolution
  - Dictionaries
  - handling of common special characters and so on

Netlist Screener

- A program that analyzes a schematic netlist for simple errors is called a **schematic screener or netlist screener** .
- schematic or netlist screener catches errors at an early stage
- Errors that can be found by a netlist screener include:
  - unconnected cell inputs
  - unconnected cell outputs
  - nets not driven by any cells
  - too many nets driven by one cell
  - nets driven by more than one cell
- The screener can work continuously as the designer is creating the schematic or can be run as a separate program independently from schematic entry.
- The designer provides attributes that give the screener the information necessary to perform the checks.

- A screener usually generates a list of errors together with the locations of the problem.
- Some editors associate an identifier, or handle , to every piece of a schematic including comments and every net.
- Most of the schematic entry programs work on a grid. Drawing a schematic is like drawing on graph paper.
    - It simplifies the internal mechanics of schematic entry program
    - It makes the transfer of schematics between different EDA systems more manageable.
    - It allows designer to produce schematic diagrams are cleaner in appearance and easier to read.

Schematic-Entry tools

- Some editors offer icon edit-in-place in a similar fashion as schematic edit-in-place for cells.
- Often you have to toggle editing modes in the schematic-entry program to switch between editing cells and editing cell icons.
- A schematic-entry program must keep track of when cells are edited. Normally this is done by using a timestamp or date stamp for each cell.
  - a text field with in the data file for each cell that holds the date and time that the cell was last modified.
- When a new schematic or cell is loaded, the program needs to compare its timestamp with the timestamps of any subcells.
- If any of the subcell timestamps are more recent, then the designer needs to be alerted. Usually a message appears to inform the changes.
- versions , version number

Back-Annotation

- After a schematic need to simulate the design - to make sure it works as expected. This completes the logical design.
- Next is to ASIC physical design and complete the layout –to know the parasitic capacitance and therefore the delay associated with the interconnect.
- **This post route delay information must be returned to the schematic in a process known as back-annotation .**
- Then you can complete a final, post layout simulation to make sure that the specifications for the ASIC are met.