Design Representation

Mahit Kumar Paul
Assistant Professor, Dept. of CSE
RUET, Rajshahi-6204

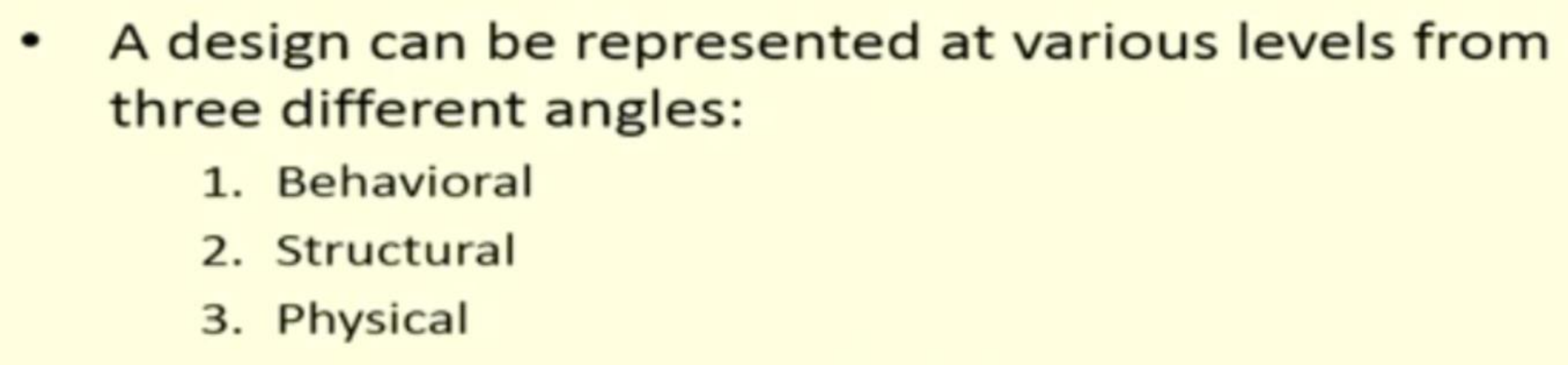
mahit.cse@gmail.com
mahit@cse.ruet.ac.bd

Design

Design can be of any level:

- High level design
- Intermediate design
- Low level detailed design

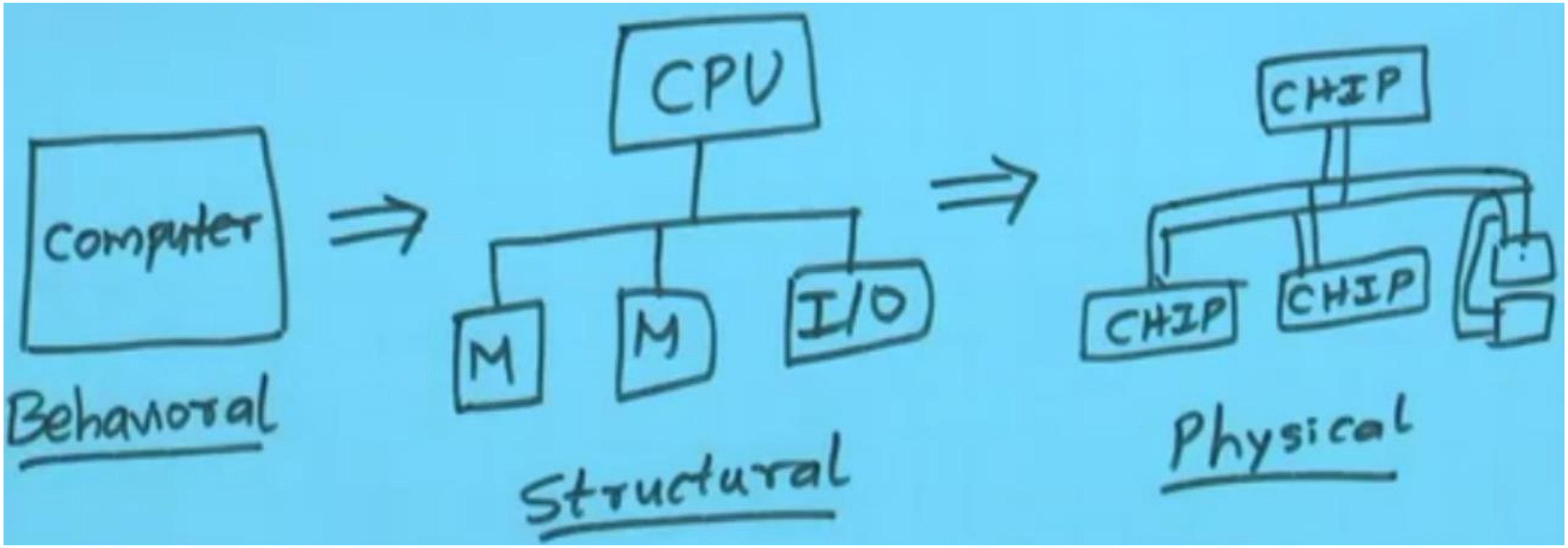
Design Representation



- A design can be represented at various levels from three different angles:
  1. Behavioral
  2. Structural
  3. Physical

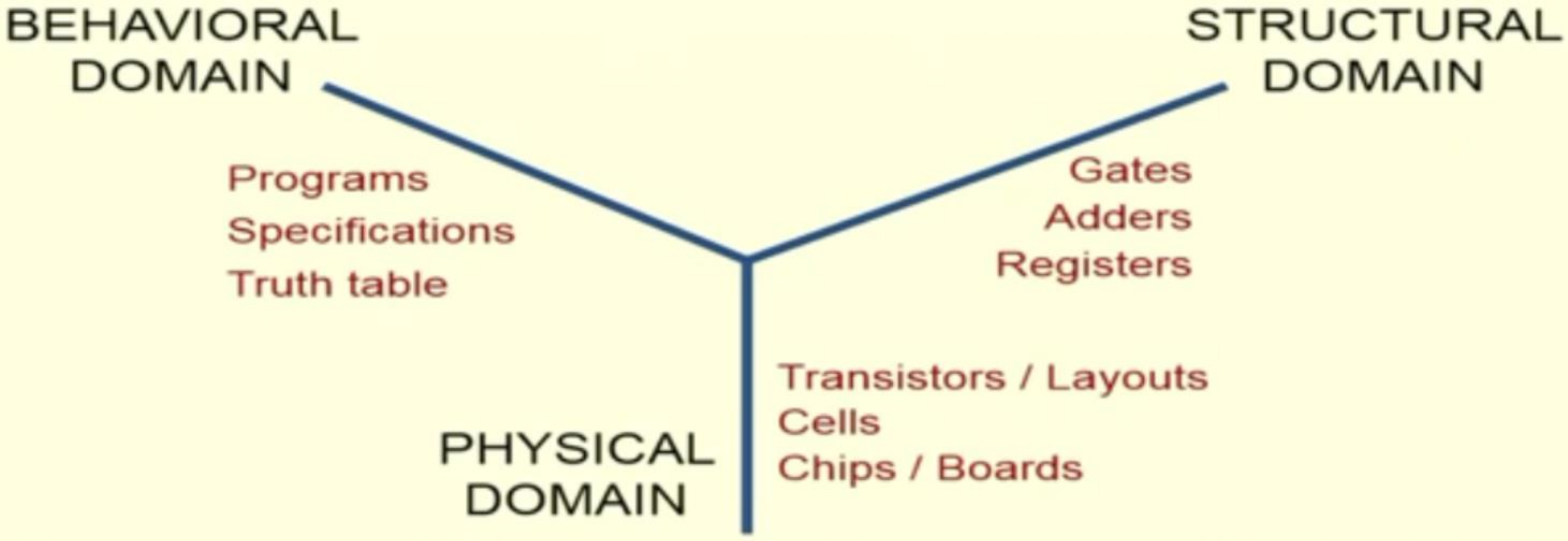
- Behavioral means what the design supposed to do.
- Structural means how it is implemented in terms of the circuitry or the net list.
- Physical means the actual implementation whether it is implemented as a chip/module/cell.

Design Representation…



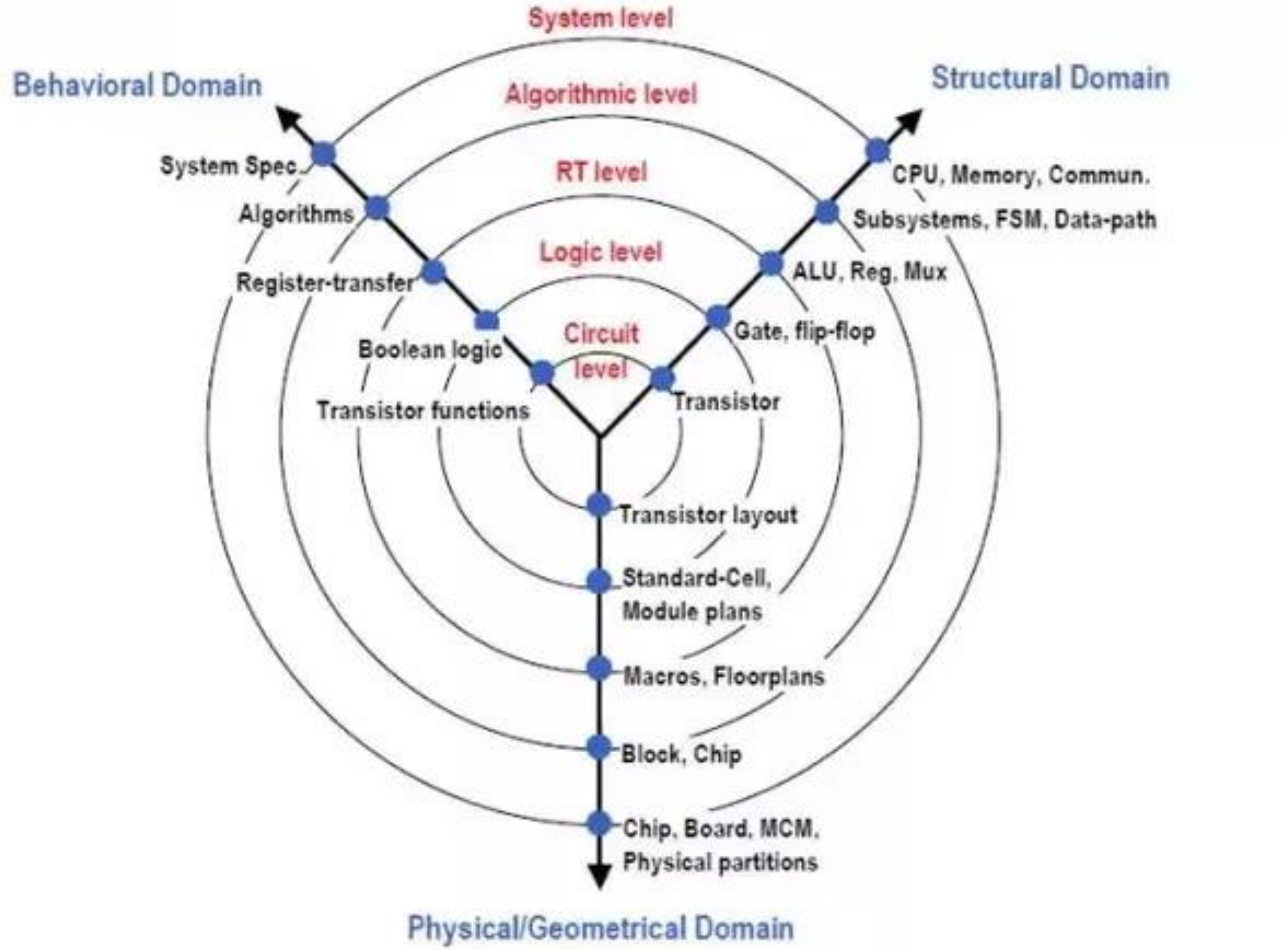
Design Representation…

- Can be represented by **Y-diagram**.



Interpretation of Y-diagram

- ❑ **Gajski-Kuhn Y-chart**



*https://www.google.com/search?q=y+diagram+in+vlsi&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiu69WAh_7bAhXEfSsKHZq2AC0Q_AUICigB&biw=1280&bih=566#imgrc=0ailg0IqvTaKNM:

Interpretation of Y-diagram…

❑ Gajski-Kuhn Y-chart…

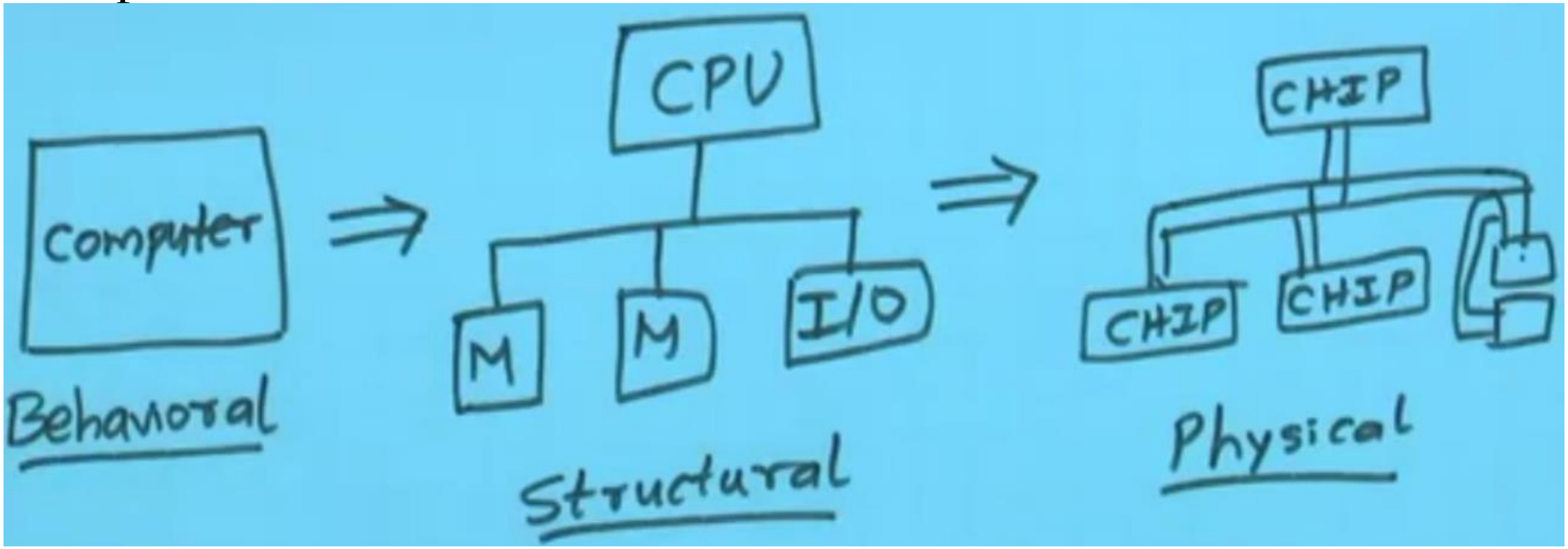
- Gajski-Kuhn Y-chart depicts the different perspectives in VLSI hardware design.
- Mostly, it is used for the development of integrated circuits.
- The outer shells are generalizations, the inner ones refinements of the same subject.
- The issue in hardware development is most often a top-down design problem. This is perceived by the three domains of behavior, structure, and the layout that goes top-down to more detailed abstraction levels.
- The designer can select one of the perspectives and then switch from one view to another.

https://en.wikipedia.org/wiki/Gajski-Kuhn_chart#cite_ref-grout_1-0

Interpretation of Y-diagram…

❑ Gajski-Kuhn Y-chart…

- On the **system level**, basic properties of an electronic system are determined. For the behavioral description, **block diagrams** are used by making abstractions of signals and their time response. Blocks used in the structure domain are CPUs, memory etc. Physical domain uses chip implementation.



https://en.wikipedia.org/wiki/Gajski-Kuhn_chart#cite_ref-grout_1-0

Interpretation of Y-diagram…

❑ Gajski-Kuhn Y-chart…

- The **algorithmic level** is defined by the definition of concurrent algorithms (signals, loops, variables, assignments). In the structural domain, blocks like ALUs are in use.

- The **register-transfer level** (RTL) is a more detailed abstraction level on which the behavior between communicating registers and logic units is described. Here, data structures and data flows are defined. In the geometric view, the design step of the floorplan is located.

https://en.wikipedia.org/wiki/Gajski-Kuhn_chart#cite_ref-grout_1-0

Interpretation of Y-diagram…

❑ Gajski-Kuhn Y-chart…

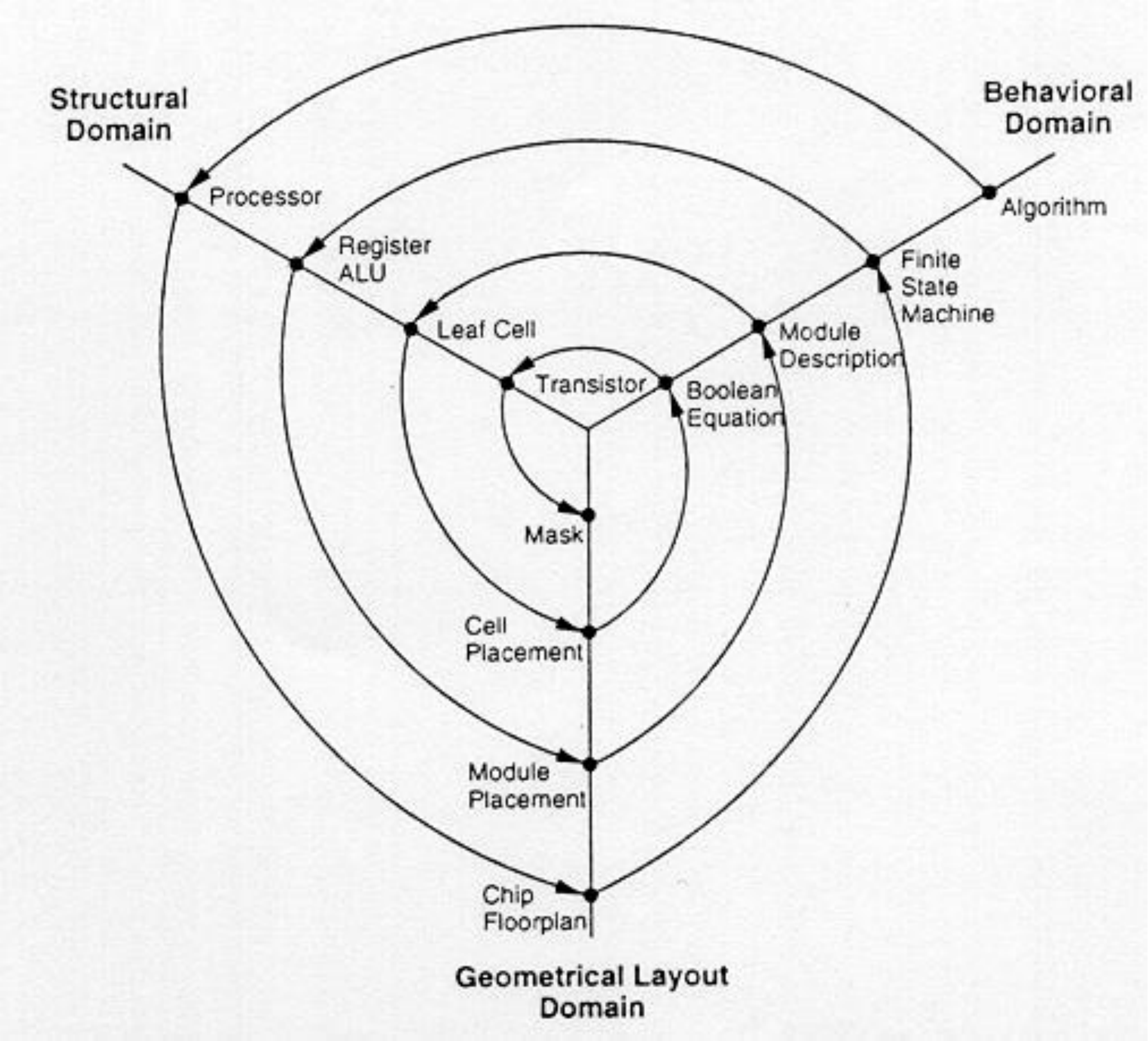
- The **logical level** is described in the behavior perspective by Boolean equations. In the structural view, this is displayed with gates and flip-flops. In the geometric domain, the logical level is described by standard cells.

- The behavior of the **circuit level** is described by mathematics using differential equations or logical equations. This corresponds to transistors and capacitors up to crystal lattices.

https://en.wikipedia.org/wiki/Gajski-Kuhn_chart#cite_ref-grout_1-0

Interpretation of Y-diagram…

- Alternative Representation of Y-chart



Interpretation of Y-diagram…

❑ Alternative Representation of Y-chart…

- This is in helix shape and actually a top-down approach.
- Starts with behavioral domain like **Algorithm** that describes how a circuit will work.
- The algorithmic behavior is then decomposed into structural domain such as some interconnections of processors, memory, I/O etc.

Interpretation of Y-diagram…

- ❑ **Alternative Representation of Y-chart…**

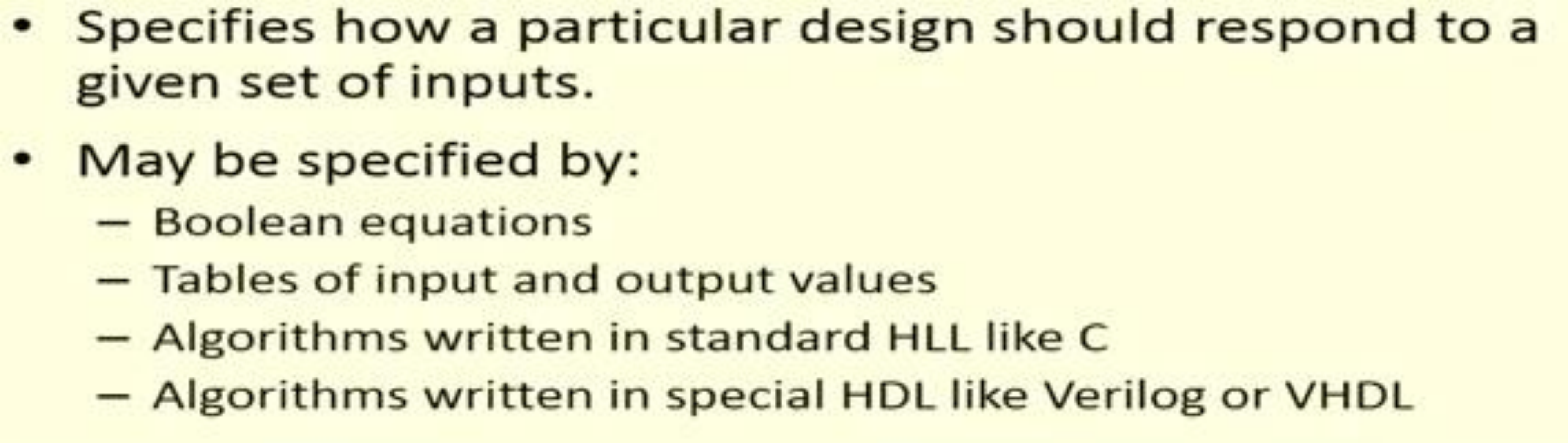
- Then geometrical layout represents the floorplan specifications of the chips i.e. their tentative layout.

- Again shifted to behavioral domain that specifies finite state machine i.e. how CPU, memory etc. works.

- And so on…

- Finally moved to transistor and mask level implementation i.e. fabrication.

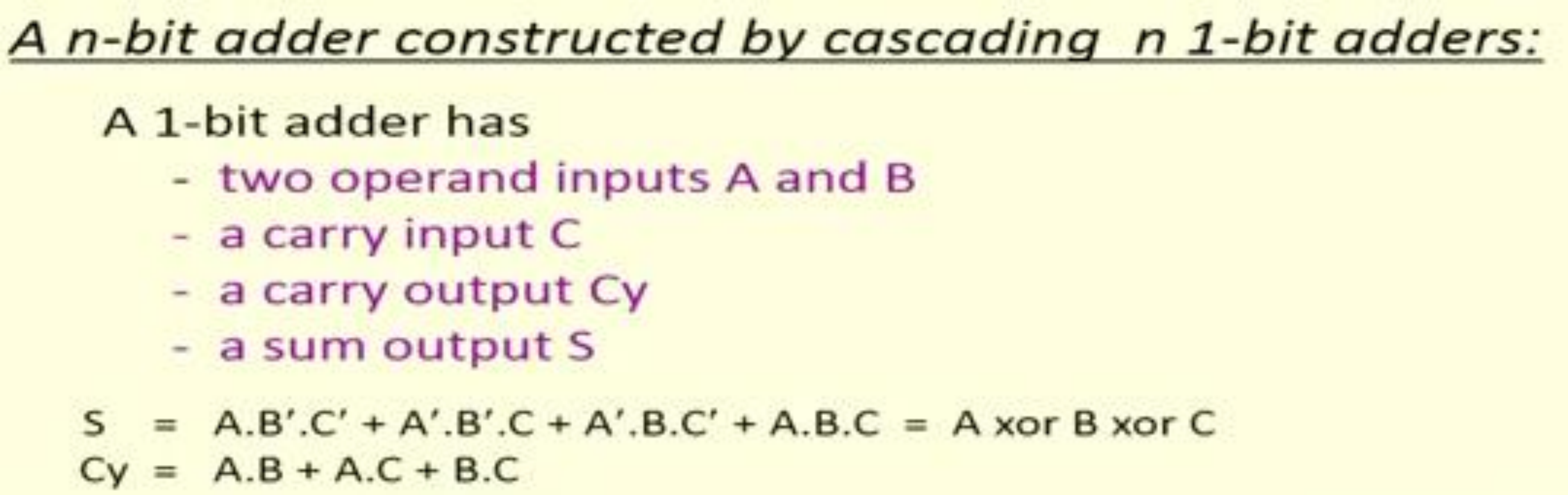
Behavioral Representation



- Specifies how a particular design should respond to a given set of inputs.
- May be specified by:
  - Boolean equations
  - Tables of input and output values
  - Algorithms written in standard HLL like C
  - Algorithms written in special HDL like Verilog or VHDL

VHDL is an **acronym** which **stands for** VHSIC Hardware Description Language. VHSIC is yet another acronym which **stands for** Very High Speed Integrated Circuits.

Behavioral Representation-Example



A n-bit adder constructed by cascading n 1-bit adders:

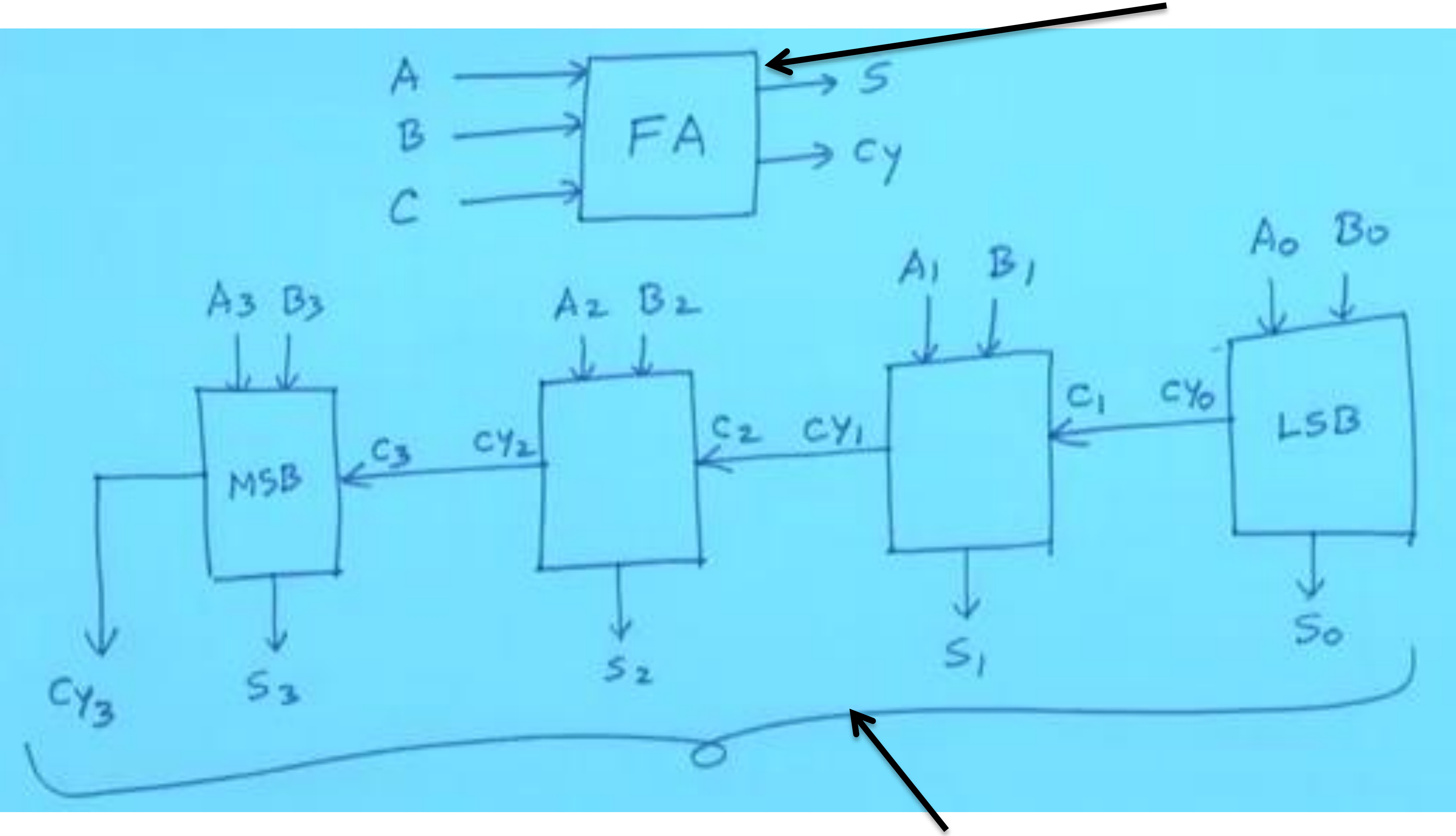
A 1-bit adder has

- two operand inputs A and B
- a carry input C
- a carry output Cy
- a sum output S

$S \;=\; A.B'.C' + A'.B'.C + A'.B.C' + A.B.C \;=\; A \text{ xor } B \text{ xor } C$

$Cy \;=\; A.B + A.C + B.C$

Behavioral Representation-Example…

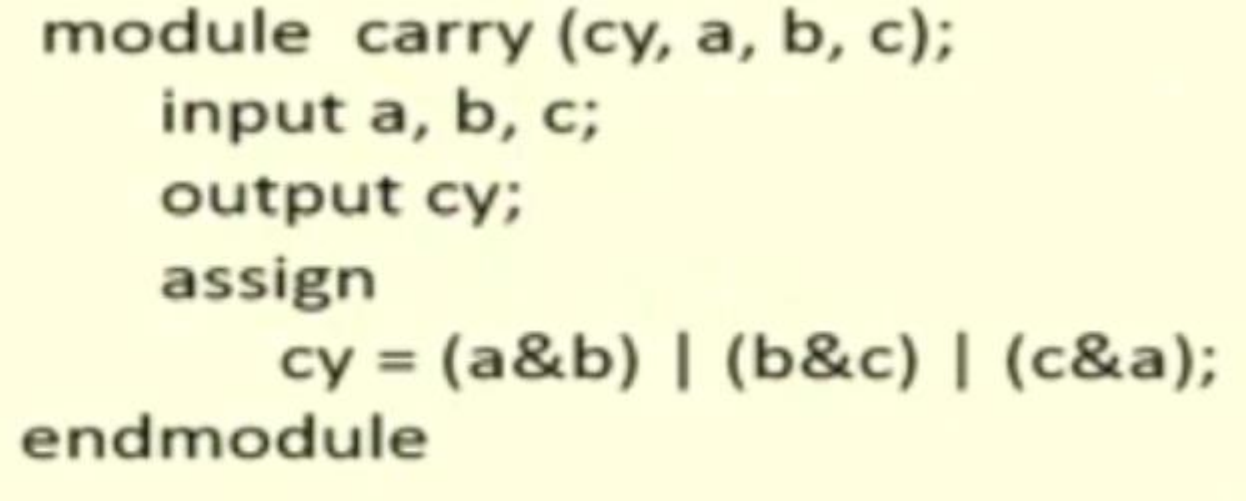


Behavioral

Structural

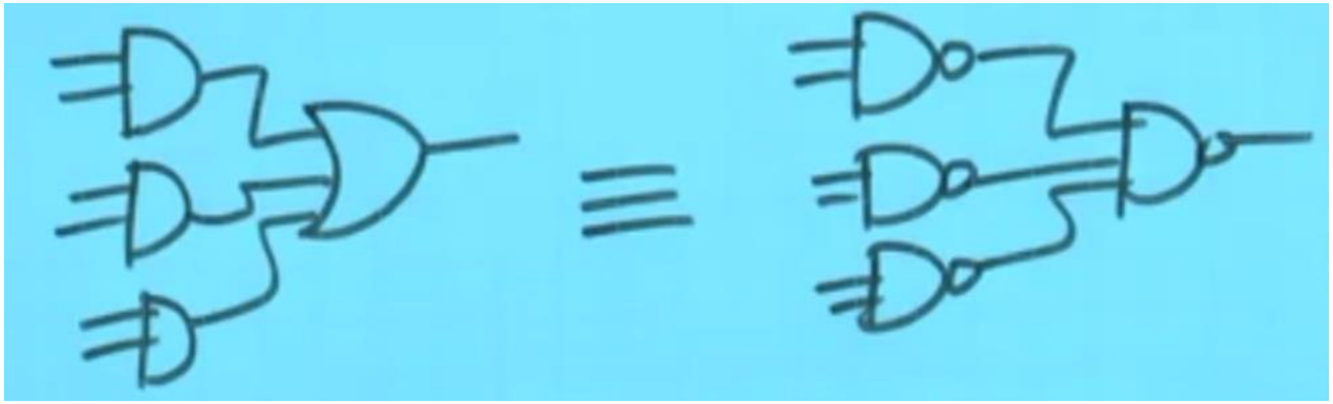
Behavioral Representation-Example…

An algorithmic level description of Cy (Verilog) : Behavioral Versus Structural



```
 module  carry (cy, a, b, c);
      input a, b, c;
      output cy;
      assign
           cy = (a&b) | (b&c) | (c&a);
endmodule
```

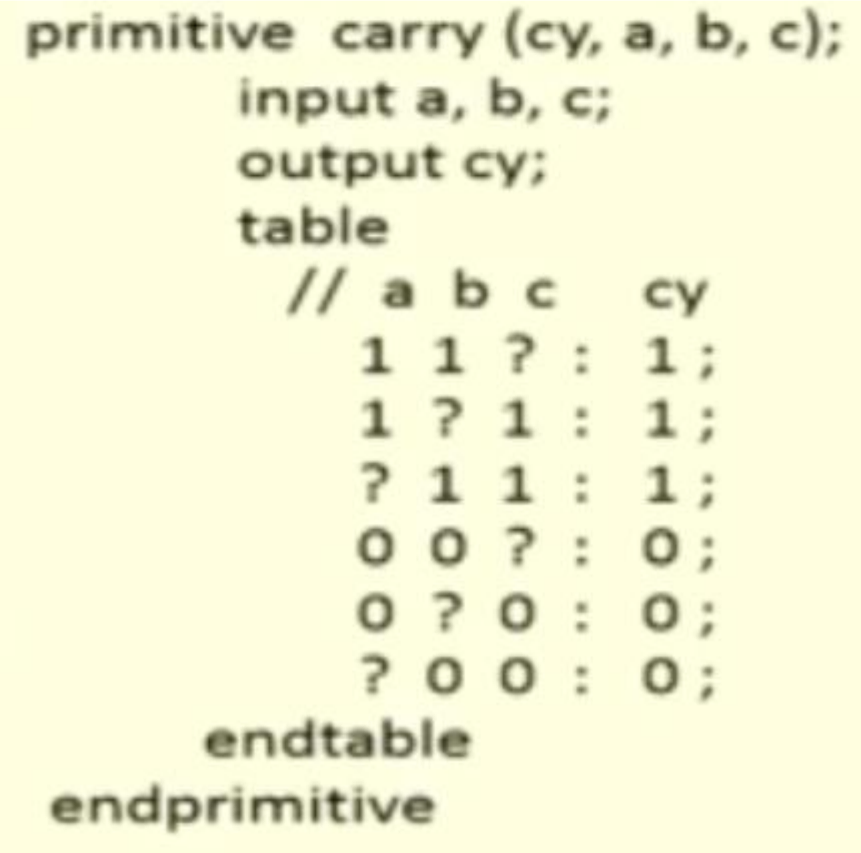
Behavioral



Two structural representation of carry Cy

Behavioral Representation-Example…

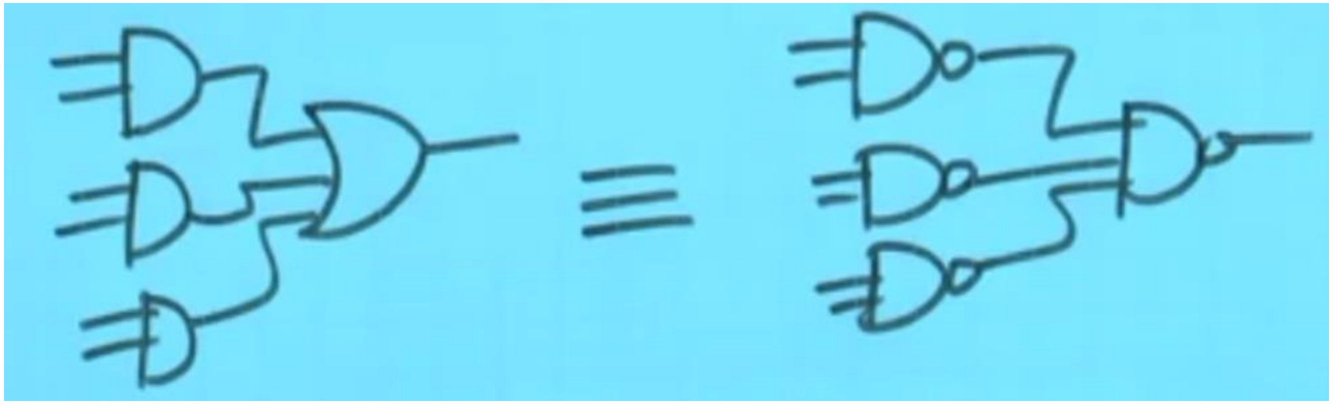
Boolean behavioral specification for Cy



```
primitive  carry (cy, a, b, c);
         input a, b, c;
         output cy;
         table
           // a  b  c     cy
              1  1  ?  :  1 ;
              1  ?  1  :  1 ;
              ?  1  1  :  1 ;
              0  0  ?  :  0 ;
              0  ?  0  :  0 ;
              ?  0  0  :  0 ;
        endtable
 endprimitive
```

Structural Representation

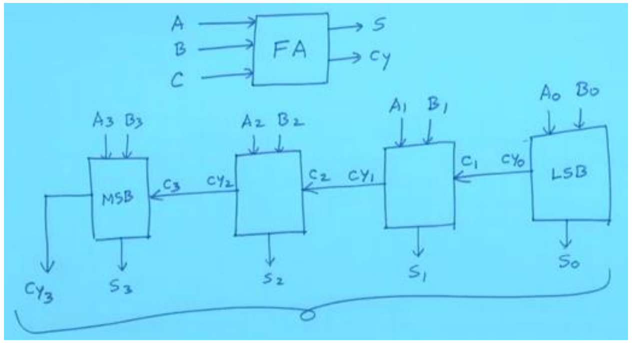
- Specifies how components are interconnected.
- In general, the description is a list of modules and their interconnects:
  - called *netlist*.
  - can be specified at various levels.



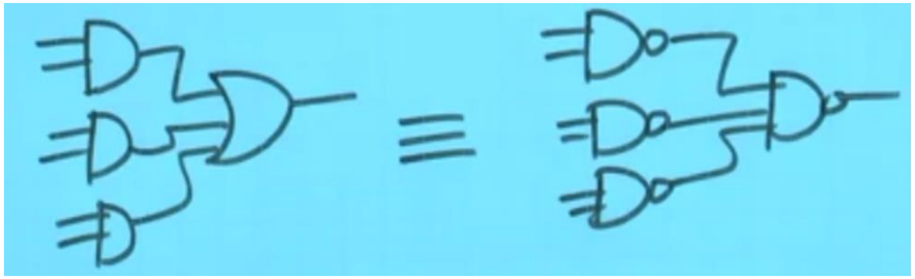
Modules and their interconnections through netlist

Structural Representation…

- At the structural level, the levels of abstraction are:
  - the module (functional) level
  - the gate level
  - the switch level
  - the circuit level
- In each successive level more detail is revealed about the implementation.

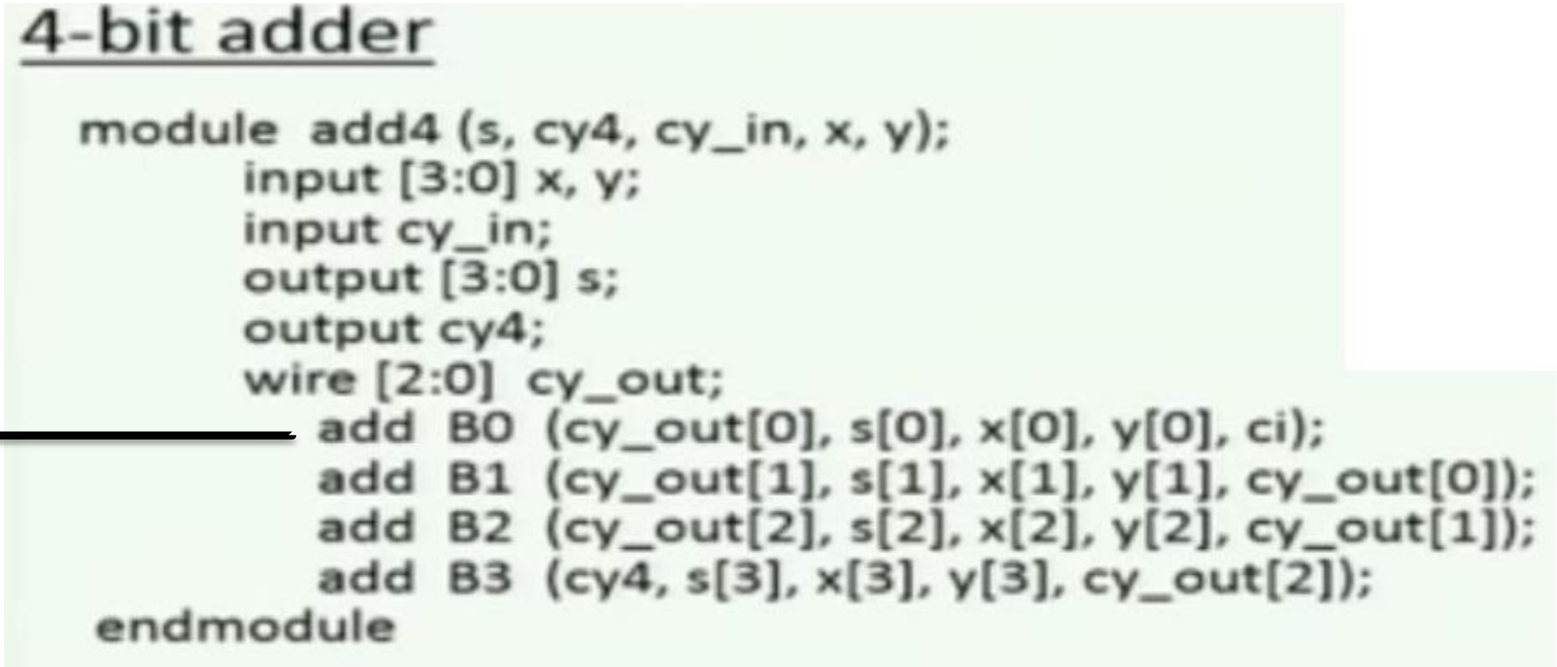


Functional Level



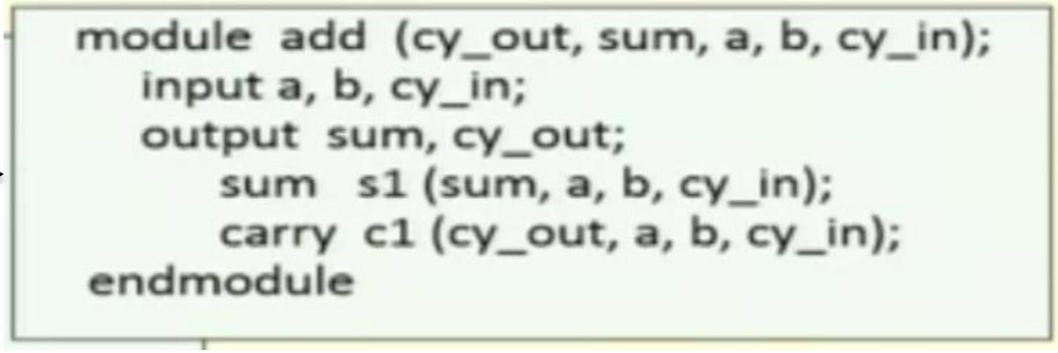
Gate Level

Structural Representation-Example



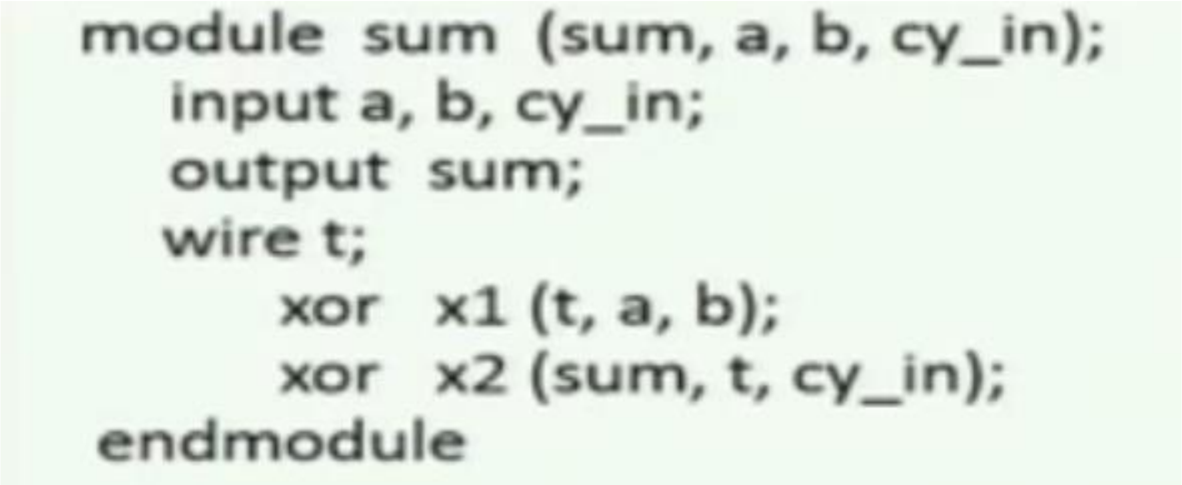
4-bit adder

```
module  add4 (s, cy4, cy_in, x, y);
        input [3:0] x, y;
        input cy_in;
        output [3:0] s;
        output cy4;
        wire [2:0]  cy_out;
            add  B0  (cy_out[0], s[0], x[0], y[0], ci);
            add  B1  (cy_out[1], s[1], x[1], y[1], cy_out[0]);
            add  B2  (cy_out[2], s[2], x[2], y[2], cy_out[1]);
            add  B3  (cy4, s[3], x[3], y[3], cy_out[2]);
 endmodule
```

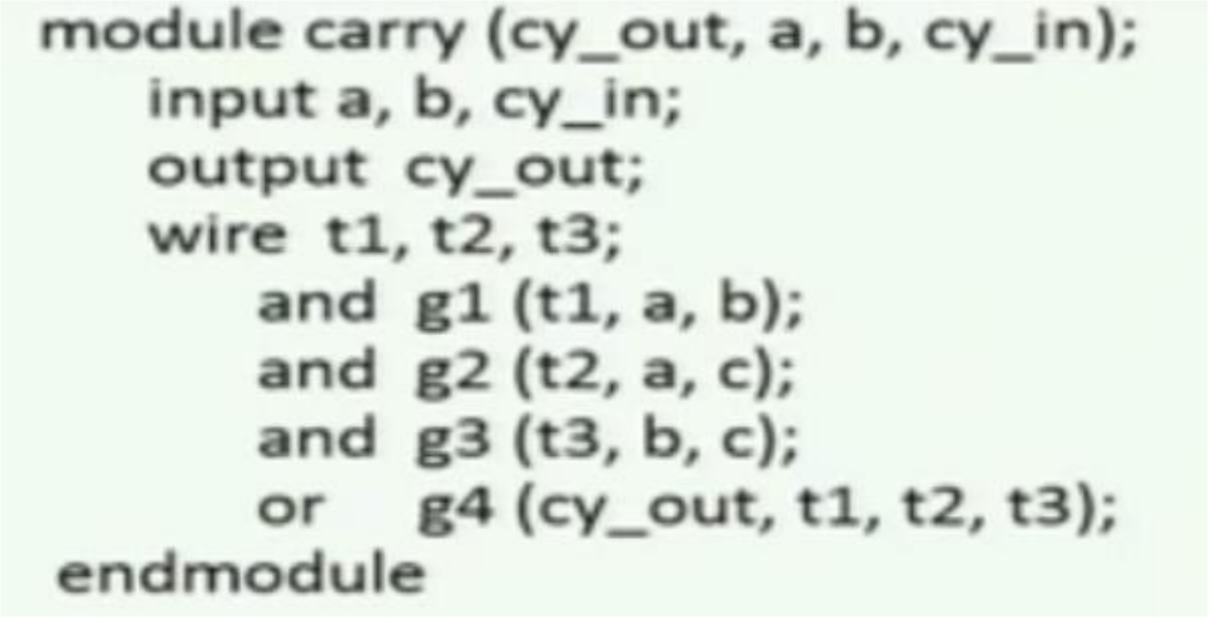


```
module  add  (cy_out, sum, a, b, cy_in);
    input a, b, cy_in;
    output  sum, cy_out;
        sum   s1 (sum, a, b, cy_in);
        carry  c1 (cy_out, a, b, cy_in);
 endmodule
```

Structural Representation-Example…

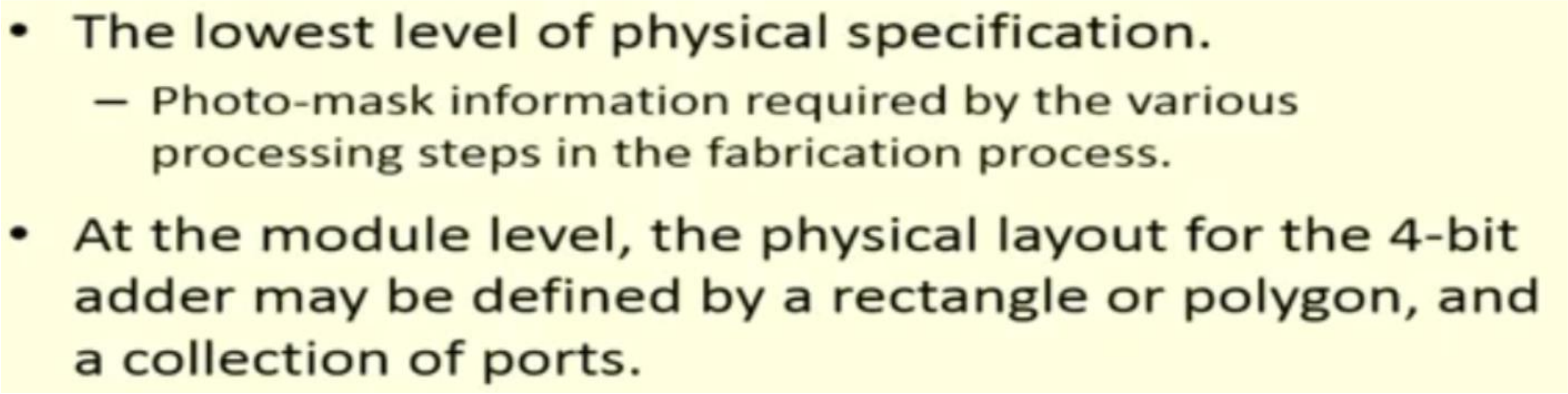


```
module  sum  (sum, a, b, cy_in);
    input a, b, cy_in;
    output  sum;
    wire t;
         xor   x1 (t, a, b);
         xor   x2 (sum, t, cy_in);
 endmodule
```



```
module carry (cy_out, a, b, cy_in);
    input a, b, cy_in;
    output  cy_out;
    wire  t1, t2, t3;
         and  g1 (t1, a, b);
         and  g2 (t2, a, c);
         and  g3 (t3, b, c);
         or     g4 (cy_out, t1, t2, t3);
 endmodule
```

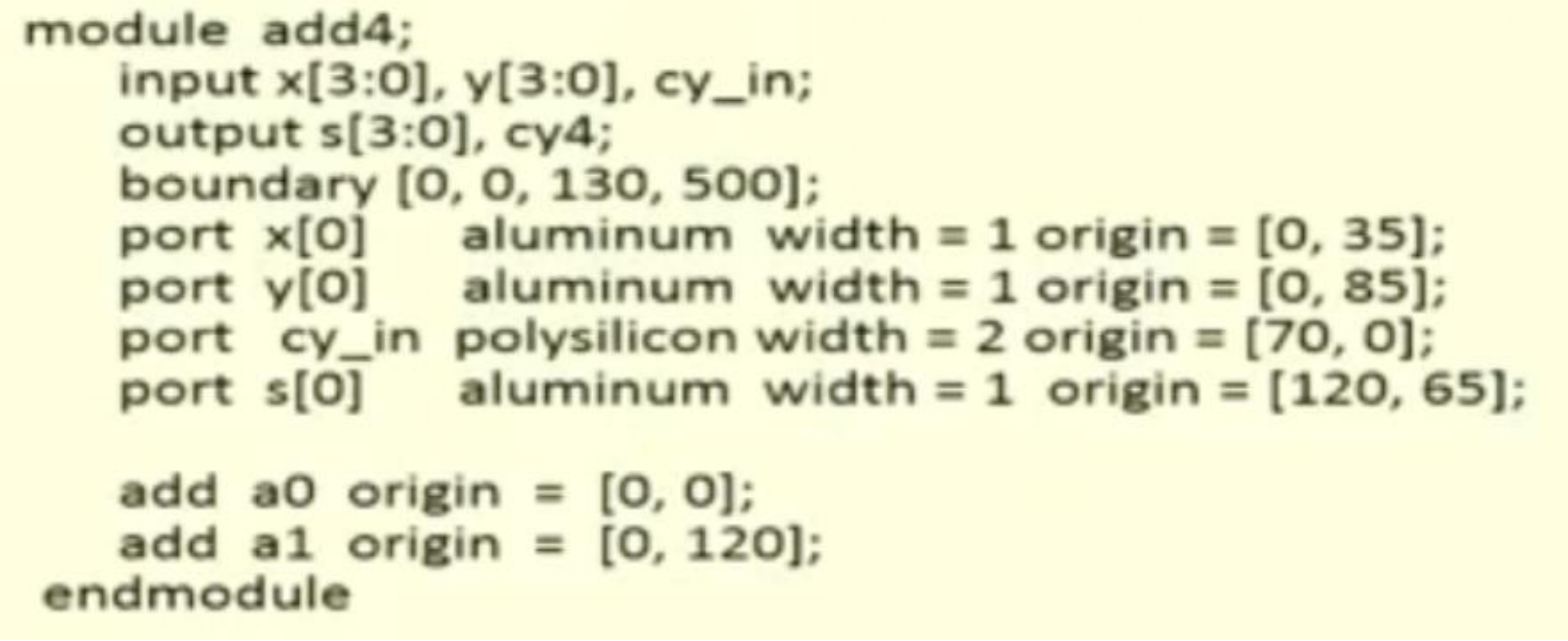
Physical Representation



- The lowest level of physical specification.
  - Photo-mask information required by the various processing steps in the fabrication process.
- At the module level, the physical layout for the 4-bit adder may be defined by a rectangle or polygon, and a collection of ports.

Physical Representation-Example

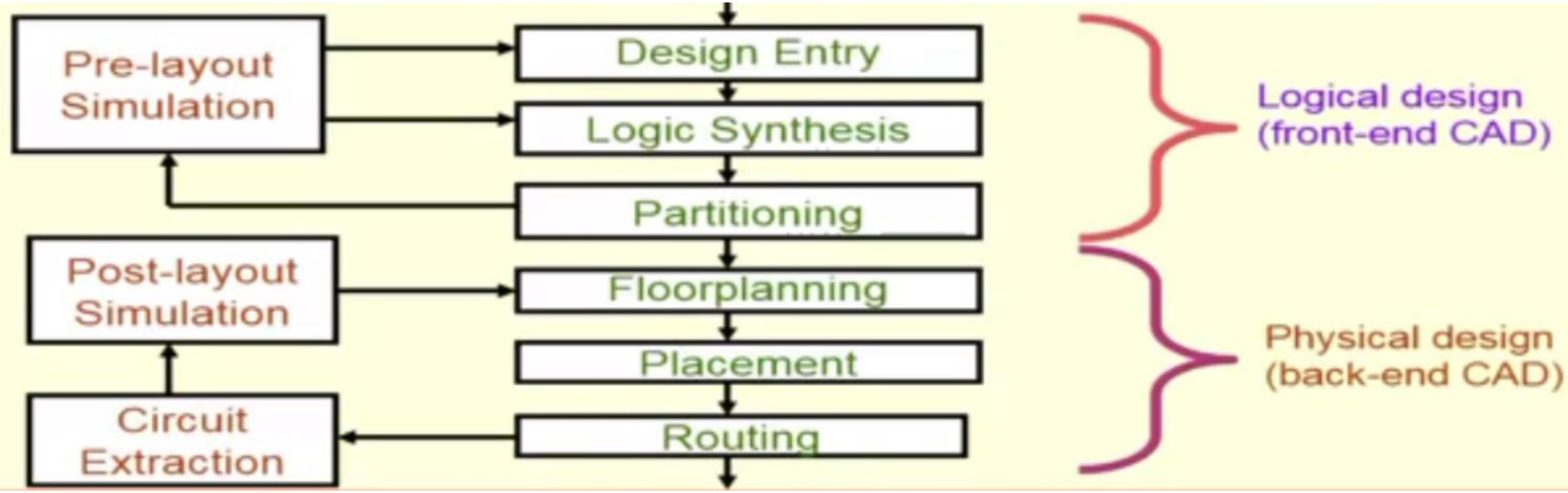
A possible (Partial) physical description of 4-bit adder:



```
module  add4;
     input x[3:0], y[3:0], cy_in;
     output s[3:0], cy4;
     boundary [0, 0, 130, 500];
     port  x[0]      aluminum  width = 1 origin = [0, 35];
     port  y[0]      aluminum  width = 1 origin = [0, 85];
     port   cy_in  polysilicon width = 2 origin = [70, 0];
     port  s[0]      aluminum  width = 1  origin = [120, 65];

     add  a0  origin  =  [0, 0];
     add  a1  origin  =  [0, 120];
 endmodule
```

Digital IC Design Flow



Acknowledgement

1. Lectures by: Prof. Indranil Sengupta, Department of Computer Science and Engineering,
IIT Kharagpur, India.