What is an Integrated Circuit (I.C.)

Explanation:

What is a circuit?! – C , R , L transistors wired together to perform specific function

A circuit Board is made up of many , different , circuits , wired together

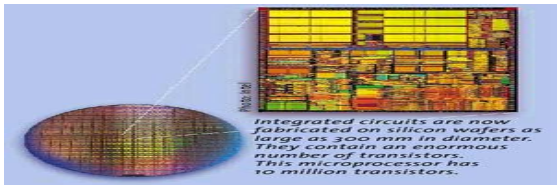
Imagine shrinking the circuit boards together into one small chip

You've just integrated circuits together , thus you have an Integrated Circuit!

History and Development

First transistor had 3 R , 1 C and 1 Transistor

Now millions!

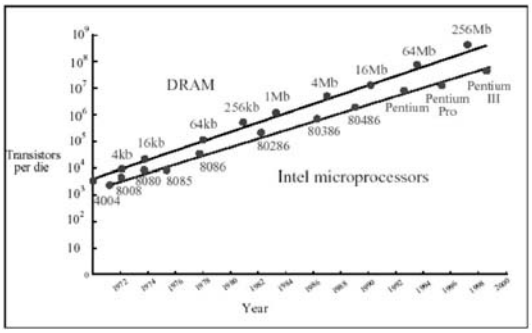


I.C is now the basic standard of industry

Manufactured mainly out of Silicon

Complex fabrication in billion dollar plants makes todays I.Cs

The Number of Devices on s single chip double every 18 months – Moore's Law



Basic Chip Criterion:

- Increased processing power over rivals
- Reduced Area
- Reduced Power Consumption
- Minimal/No increase of production costs
- More for less – life's a bitch!

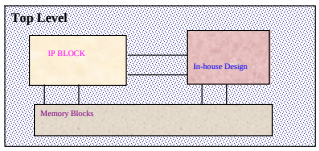
Industrial Practices

2 choices:

Design Everything

or

Buy part ,design the rest



Explanation:

A Core Macro(s) – call an I.P Block

- re-use
- royalty
- guaranteed
- black-box

Memory Blocks

- size
- speed
- model

In House design - Sexy!

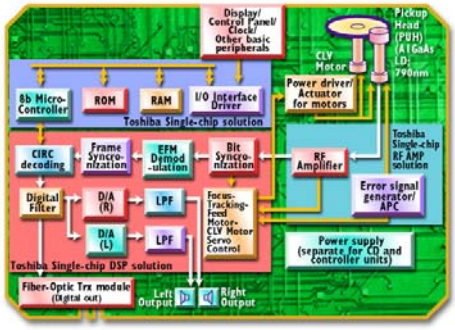
A Chip is divided in blocks

Designers work on 1/2 blocks each

Designs integrated at top level

Top level and pads tested

Example - CD Player



Blocks - divided into sub-blocks

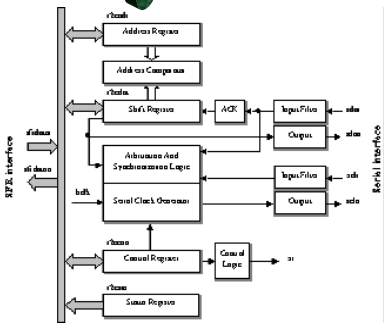
Each sub-block has a particular function within the block

We'll use the I2C block as an example

I2C - 2 wire block used for inter-comms between blocks

Present in many chips found on Circuit Boards

and of a Block



The Guts

- Each sub-block performs a task
- Same clocked blocks should be grouped together
- Need to test every sub-block
- The buck stops with Designer

Now What?

Sitting at my desk

Mr. Toshiba rings

Offers me $1,000,000 for complete in house design

What do I do?! :

I follow the
DESIGN FLOW DIAGRAM!

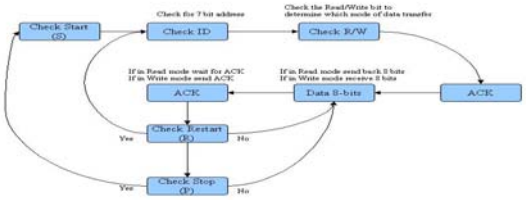
DESIGN FLOW DIAGRAM

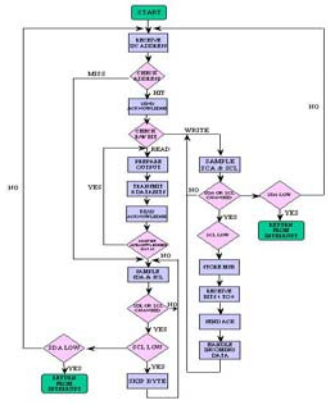
- Document + Review
- Behavioural Model Rtl
- Behavioural Testbench
- Observe and re-design
- Register RTL Coding
- Block RTL Testbench
- Synthesis
- Gate Level Testing
- ATPG Prelim.
- Layout – DRC etc
- Post Layout Test + ATPG

Documentation and Review

Need to Fully describe the "internals" of the design

Use State Machine Diagrams – I2C Example:





State Machine Diagrams

Here all possible actions and counter-actions are examined

Square boxes = Action

Rotated boxes = choice

If no state machine diagram, have timing diagrams explaining actions

Boring, time consuming

Behavioural Model RTL

Do not care about physical reality

Testing algorithms

Testing pins/general functions

Can use C++,MATLAB , or a HDL

Testing

At final stage , if you don't test and block doesn't work….P45

$100,000 to fix a mistake

Exhaustive testing will save your ass

Register RTL Coding

• Need executable specification for our design

•Need to be able to complete describe what you want

•Verilog / VHDL are the 2 HDL languages

•Use GUIs like Undertow or Signalscan to view simulation and debug

General Coding Practices:

- Code must be compact and area efficient
- Chip divided between synchronous and asynchronous
- Avoid latches , tri-states,un-defined states
- Remove any glitches – can cause power surges

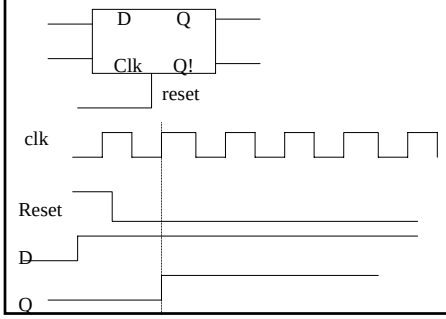
Verilog Example

```
module <module_name> (input1 , input2 , output1 , output 2 , clock , reset);

input <input1>, <input2>;
output <output1>;

begin

        Code here
end

endmodule
```

What's this?

```
always @ ( posedge clk or negedge reset)

    input d , clk , reset;
      output q;

      reg q;
       begin

         if (!reset)
              q <= 0;
           else
              q <= d;
        end

    endmodule
```

A Simple Synchronous D-Type Flip Flop

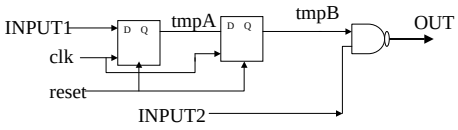


Logic Gates and another code:

Logic gates , like NAND , NOR , XOR are written

<logic_gate_name> <reference_name> (output,input1,input2);

i.e. : NAND ND1 (out1,in1,in2);




```
module new_design (       INPUT1 , Random , OUTPUT1 , clock , reset);

input INPUT1 , INPUT2 ,clock,reset;
output OUT;

reg tmpA , tmpB;

flip_flop ff1 ( INPUT1 , tmpA , clock , reset); "flop_flop"  instantiated

flip_flop ff2( tmpA , tmpB , clock , reset); "flop_flop" instantiated

nand nd1( OUTPUT1 , tmpB , INPUT2) ; "nand" instantiated

endmodule
```

More RTL Rules:

– Register all outputs

– Separate blocks for positive and negative clocked flops : to avoid lock up latches and timing issues

–Avoid too many hierarchial blocks : for ease of synthesis

– Critical path in one level of hierarchy only

And even more rules!

- Use 1 Master Reset
- Use minimal amount of Clocks
- Use D type flops- most efficient
- Use multiplexors instead of tri-states

•DO NOT put logic on clock line , ie do not “AND” the clock line with anything else – very dangerous

BLOCK RTL Testbench

TESTING IS VITAL

Every node,every possible combination must be tested

Tools available that will generate testbenches and check amount of nodes toggled in tests

Typically 98% of a block should be tested

SYNTHESIS

2 Steps :

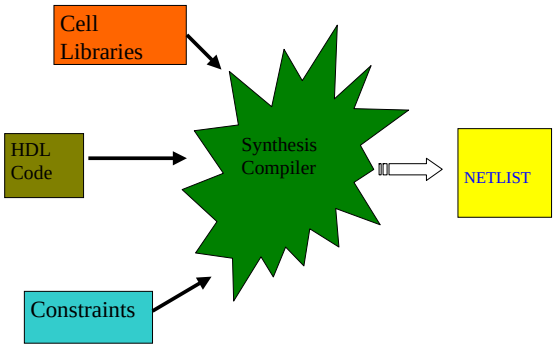
Translation

And

Optimisation

Translation : Transforms HDL to Gates

Optimisation:Select cells and libraries to constrain design



CONSTRAINTS – 2 types

- Optimisation Constraints
- Design Rule Constraints (DRC)

- DRC – imposed by cells/libraries used
- Optimise for :
  - » Speed and Area

DRC has priority

Three DRC Constraints:

1. Fanout
2. Transition
3. Capacitance

Fanout = No. loads a pin can drive

Transition = Longest time from 1 to 0 or vice-versa (RC)

Capacitance = D'uh!

A CELL

Devices with certain , specified characteristics

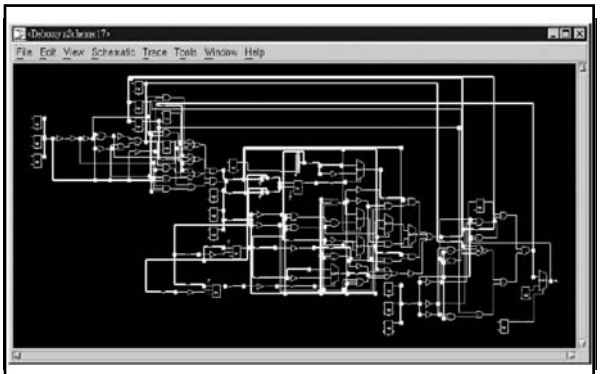
Flip-Flops , NANDS , NORS etc

– characterised for rise/fall time , heigth, width etc

Can design or get from foundry – like buying/making a cake

Libraries

- Given by Foundry Maker (Intel,ST)
- A Collection of CELLS
- Library characteristics defined by foundry capabilities – this controls optimisation
- Have Best Case and Worst case capabilities



Post Layout Testing

•Takes netlist from layout, with real values for the C , R , L of actual wires and of the load

•Timing will thus be different as real time delays

•Must check that behaviour is still correct , with these real , physical delays

•Must test WC and BC – timing provided by layout

What is Chip Test ?

Chip Test is the mechanism by which silicon samples are deemed to be free of manufacturing faults

•Proves the manufacturing process hasn't introduced defects

•"Guarantees" that the device is the taped out one

•Used as a Go/No-Go in device production

Test : To avoid costs associated with device failures

• manufacturing process is not perfect

•For a given process and foundry , there's an expected yield

•Yield is expressed as the number of defects per device , or the no. of defects per square inch

How is Chip Test Performed

By running a set of vectors on each silicon sample and verifying that the device generates the expected output

A "vector" is a of stimuli to be supplied to the device under test (dut)

A vector set needs to be as small but as efficient as possible – to maximise time on tester machine

Test Methodologies

Controllable

• easily set to known value

Observable

• easy to see the effect of that node having that value

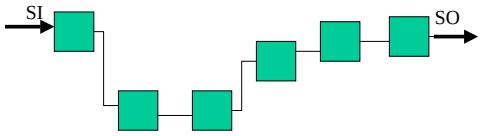
Note : Not always possible to toggle cct. node , eg nodes actually tied to Vdd/Gnd

SCAN IN , SCAN OUT , TEST-ENABLE – all I/O's

When TEST ENABLE asserted , the scannable sequence forms a "Scan Chain"

- a mechanism by which complete state of cct. Can be initialised (SCAN IN phase)
- through which the state of the cct. Can be observed (SCAN OUT phase)

SCAN CHAIN



Scan Test

- The SCAN IN input of a flop connects to chip level SCAN IN pin
- The SCAN IN of the remainder of scannable flops is connected to SCAN OUT of previous flops
- The final SCAN OUT is connected to chip/block level SCAN OUT pin

SCAN PHASES

- Scan In Phase
  data loaded into chain from primary inputs
- Capture Phase
- Scan Out Phase

"**Single Stuck at**" Fault Model

- Most Commonly accepted Fault Model
- Assumes the fault makes a logic gate behave as if it's tied to Vdd or Gnd
- Relatively simple and quick to calculate

Fault :
A physical Defect in the circuit that affects its correct operation

- Mask Generation Errors
- Dopant Level / Temp/ Time innacuracies
- Die Manufacturing errors/stresses/cracks
- Packaging Errors

Synchronous or Asynchronous

Synchronous

Every event is controlled by a clock edge

Can use State Machines – as in I2C

Used in ~90 of modern day ICs

Asynchronous

No global clock – use handshaking instead

Used in early computers

Advs. Of Synchronous

- Helps avoid “metastability problems”
- Race free – glitch free design (?!)
- Guarantees testability of large designs
- Signals traceable using clock edges

Disadvantages of Synch.

- Increasing clock frequency leads to possiblity of “clock skew”
- Excessive power consumption due to toggling clock signal
- High Current transients
- Compatability with EMC regulations

Benefits of Asynchronous Design

•No Clock Skew problem (as no clock!)

•Potential of reduced power – as there is no continuous clock

•Less EMC emitted

HANDSHAKE

- 1. The Sender places a valid data value onto the bus
- 2. The Sender then issues a “Request” event
- 3. The Receiver accepts the data when it’s ready

4. The Receiver issues an Acknowledge event to the Sender

5 .The Sender may then remove the data from the bus and begin the next communication

Why Synchronous

•Asynch Difficult to test – hard to generate ATPG

•Not established – therefore not trusted

•Tried and tested and has proven itself

•Speed of device uses clock as reference

•Synthesis tools are clock driven