14/5/25

__/__/____

FIELD PROGRAMMABLE GATE ARRAY

Gates: Tiny pieces of h/w on the chip that make up the design
eg: AND, NOR, NAND.

Array: Group of gates in millions and billions

Field Programmable: Connections b/w the internal components are programmable after deployment. / "RE-PROGRAMMABLE, RE-CONFIGURABLE.

ASIC - Application Specific

Reconfigurability can be done remotely, it's called remote system upgrade. → FIELD, out in the world

Software can control processor
control hardware

SoC → System on chip.
processor, memory,

Why FPGA
→ Reprogrammable & Flexible
→ Future proof & ensure product longetivity
→ Reduced time to market
→ FPGA market size optimised (low to mid range unlike ASICs)

$\overline{(\bar{A}+B)} \cdot C = \overline{(A \cdot B)} + C$ __/__/__

Application

→ Consumer Automotive
→ Instrumentation
→ Military & Industrial
→ Computers

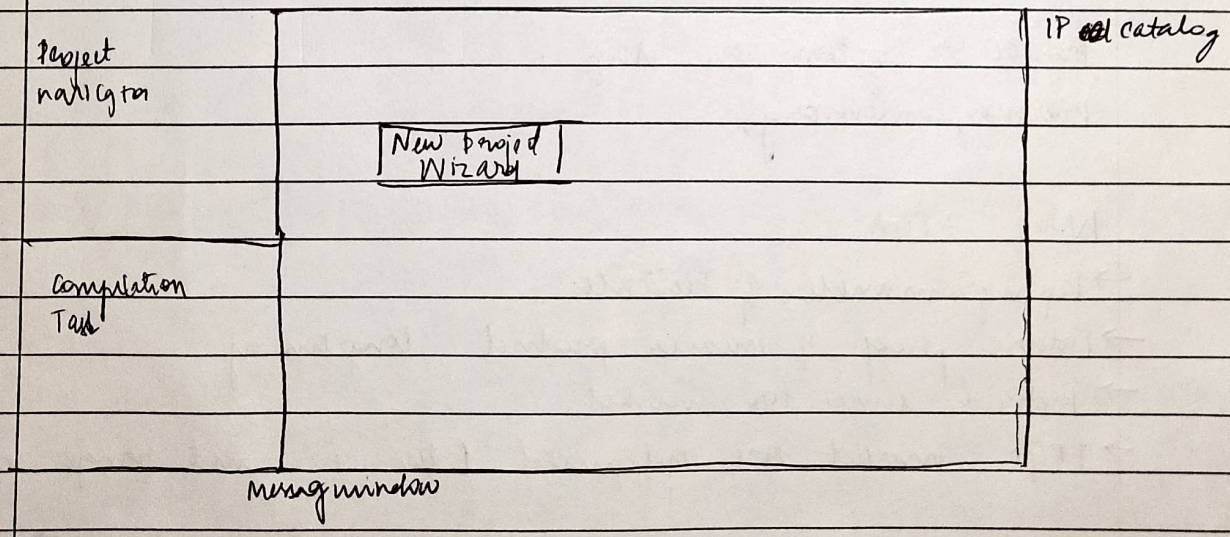
4 or 8 inputs per LUT

ALM - adaptive logic module: every might one can

reg - sequential logic

combinational logic + sequential = counters, etc

Quartus Prime - Lite
Quartus Prime - Standard
Quartus Prime - pro



__/__/____

- Proo name & directory (project name = module name)
- Name of top level design entity
- Project files & libraries
- Target device family and device
- EDA tool settings → NEXT

Change directory name → C:/DLD_VIT
name → fulladder
top level module → fulladder → NEXT

Create a new directory → YES

Empty proj ✓

No dependencies

Family, Device & Board settings

Family : Max 10

50K ~~RAM~~ LUT
40nm
10M50DAF484C7G → Type in name ~~filt~~ filter
commercial

→ check NEXT

EDA TOOL SETTING → leave it alone → next

SUMMARY page → Top level = fulladder → ensure this
device family name 10M50DAF484C7G → ensure this

File → new → verilog HDL file → OK

__/__/____

```
module fulladder
(input a, b, cin),
output s, c);
assign s = a^b^cin,
       c = a&b | b&cin | cin&a;
endmodule
```

save it as "fulladder.v"

Processing → start → start analysis & Elaboration
it should show "Quartus prime was successful"

Tools → Netlist Viewers → RTL Viewer (Register Transfer Logic)

Pin planning → in data sheet

Assignments → pin planner → package name = chip name.
MAX10 - 10M50DAF484C7G

Node name	Direction	Location	I/O Bank	I/O Std
a	i/p	PIN_C10	7	3.3V VTTL
b	i/p	PIN_C11	7	3.3V VTTL
c	o/p	PIN_A9 ~~PIN_C~~	7	3.3V LVTTL
cin	i/p	PIN_D12	7	3.3V LVTTL
s	o/p	PIN_A8	7	3.3V LVTTL

→ minimise

Analysis & synthesis in compilation window → right click
Total No. of logic elements: 2.

Tools → Netlist viewer → Tech map viewer (post mapping)
Box → Right click → properties → Truth table
minimised

__/__/____

Synthesis report → Analysis & Synthesis

Resource usage summary → 4 i/p func=0, 3 i/p func = 2

Fitting prog into device

Compilation window

Fitter → right click → start ; gives flow summary, less than 1% of 49706 LUT are used, device doesn't consider FA to even be a design

TOOLS → chip planner (shows where program exists on board)

Assembler → (generate programming file) · aka 'SRAM object file' & extension (.sof)

Use a JTAG to USB
- JTAG → chip
- USB → laptop

TOOL → programmer → Hardware setup USB Blaster → click start

waveform if at all is called functional verification

directory : /DLD_VTT_EX2

mux 2 to 1

15/05/25

We take IPs

SOC : contains hardcore - can not be changed after manufacturing
soft core - available as a code, can be edited
user logic

SVB → SOC
↳ FPGA, Memory, DSP } should be available as an IP

1
a Insert template option → New Proj Wiz
b IP code → Open Proj

2
a Template
b IP

1a) New → C/Memory   Name: mem
Next, empty; MAX10 ; filter name ; OK
Next , finish

file New Verilog HDL

Attachment like symbol ; Next one is insert

popup: choose ver lang; here Verilog
verilog HDL → full designs → RAMs & ROMS
→ single Port RAM
click Insert → close

→ Change Name of module to folder name so that errors don't occur & so that conditions are satisfied.

→ file → save as → Same as folder / modulename

→ Double click → Compiler → Does all the Things.

→ filter → Resource Section → Resource Usage Summary

Here: MBKS → 1/102

; 64x8 = 512 bits used ; 9x1024x182 = 167 = 312 pages

Blue: Empty White = DSP Yellow: Memory

1b) New → C:/Memory_IP Name: mem

→ same as usual

Right hand Side → IP catalog → search "on chip Memory" → OCM

Select "RAM 1-PORT"

pop up = Give proper C: Folder name / filename

pop up = 8 bits ; 64 width → Next

Next, ---: "Yes"

change from Hierarchy to file in the Proj Nav

Tools → chip Planner to see in the chip

_/_/__

2a)

directory : C:/multiplier

name : mult → NEXT

Family : MAX10

: [illegible]

File → new → verilog HDL → ~~insert template~~ Full design → ~~multiplier~~ arithmetic → multiplier → unsigned multiplier → insert

change module name to "mult"

#(parameter WIDTH = 16) → save

compile in compilation window

shows (2/288) multipliers occupied

Tools → chip planner.

2b New proj wiz

C:/ multiplier_ip

name: mult

Family : same as before

file : "

IP catalog : Type MULT : LPM_MULT → parameterized module

Library : size = 16 bits, size = 16 bits } size of product = 16 + 16 = 32 bits

- Another pop up → No change when both are variable 32
- No need for pipelining; leave them default
  Next → finish
- Compile the code.;

3) Platform Designer: → soft processor & 4) Cyclone V → hard processor

→ New project → C: PD_VII name: Top

Tools → Platform designer
left side → search
processor - Nios 2

Errors → due to No connection to Clk & Reset
Connect Clk & Reset by clicking on dots present.

- memory → on chip memory
  → connect master signal of processor to slave signal of memory

System: assign → base address

- UART → Jtag →

Nios → Vectors → Reset Vector: On chip
Exception: On chip

Generate HDL when no errors.
Give name. Close ...

_/_/__

Three dots filename.

Project → Add/Remove → "..." → text. open

separate eg. integ. file

Note

↗ System Integration : Just save as Version each New edit.