ANALYSING THE BITSTREAM OF ALTERA'S MAX-V CPLDS

Jean-Francois Nguyen

July 14, 2016

LSE Security System

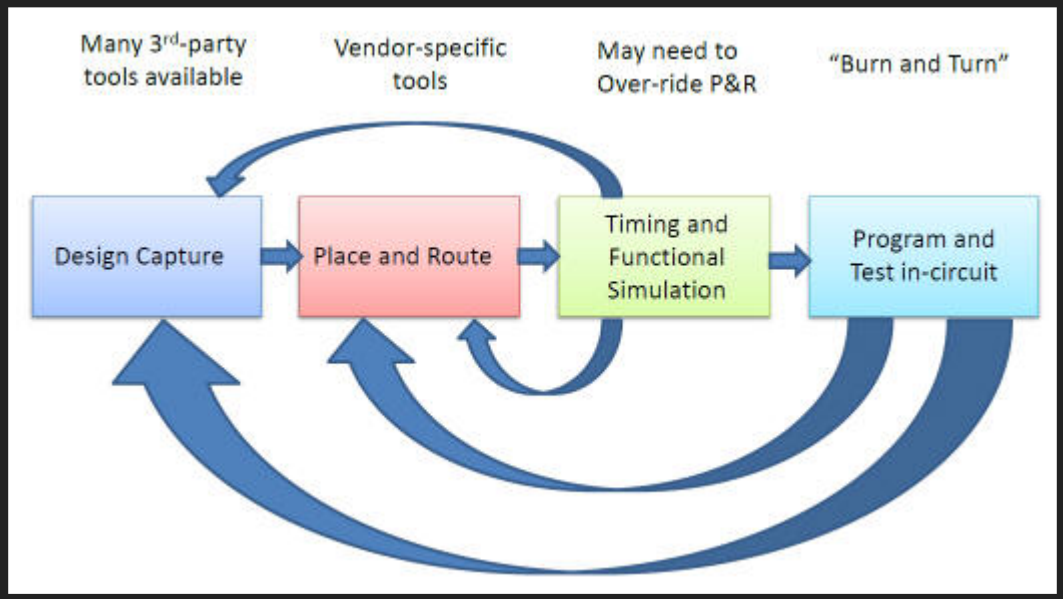
SUMMARY

1. CPLDs and their design flow
2. MAX V internals overview
3. Analysis of the bitstream

CPLD ?

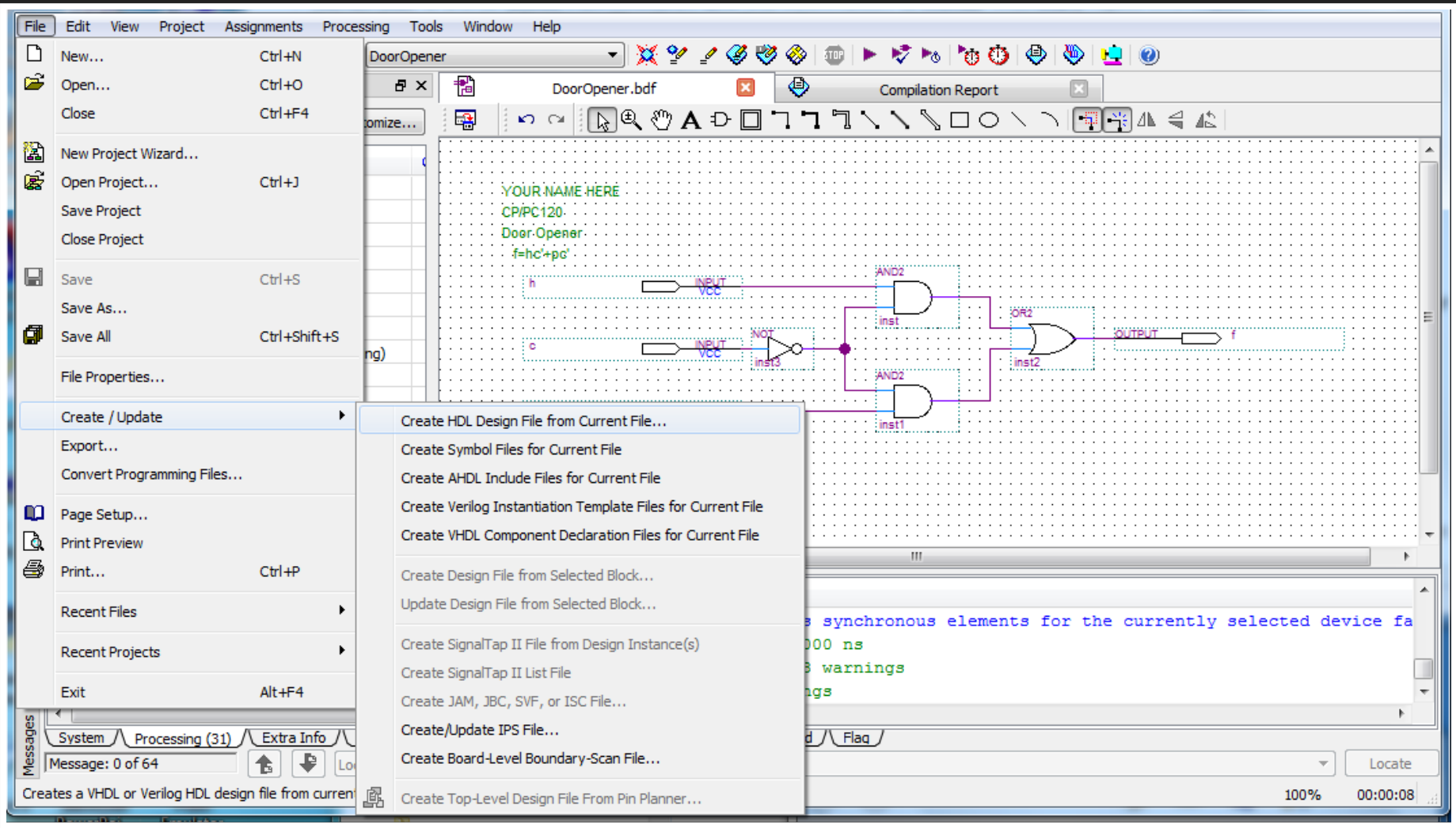
- Complex Programmable Logic Device
- Used to build reconfigurable digital circuits
- Less complex than a FPGA
- Configuration is stored on on-chip flash memory

DESIGN FLOW FOR CPLDS



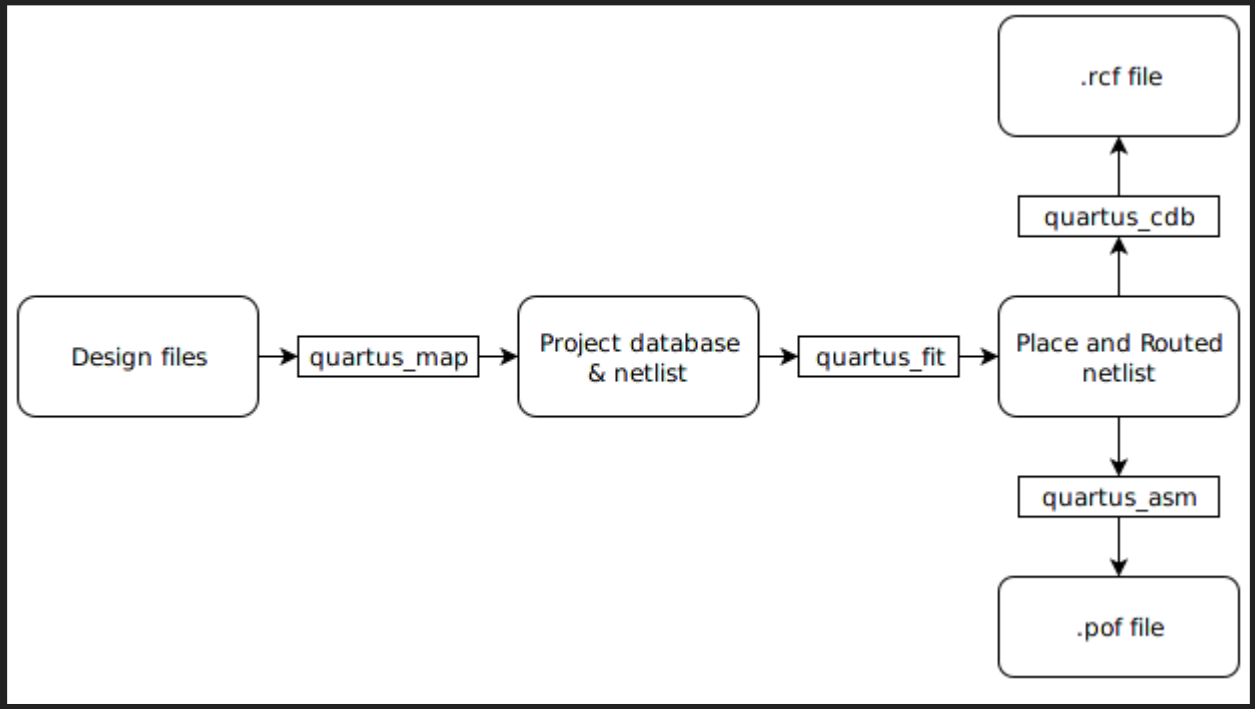
DESIGN FLOW FOR ALTERA CPLDS

DESIGN FLOW FOR ALTERA CPLDS



THE QUARTUS DESIGN FLOW

Actually, it does have command line tools...

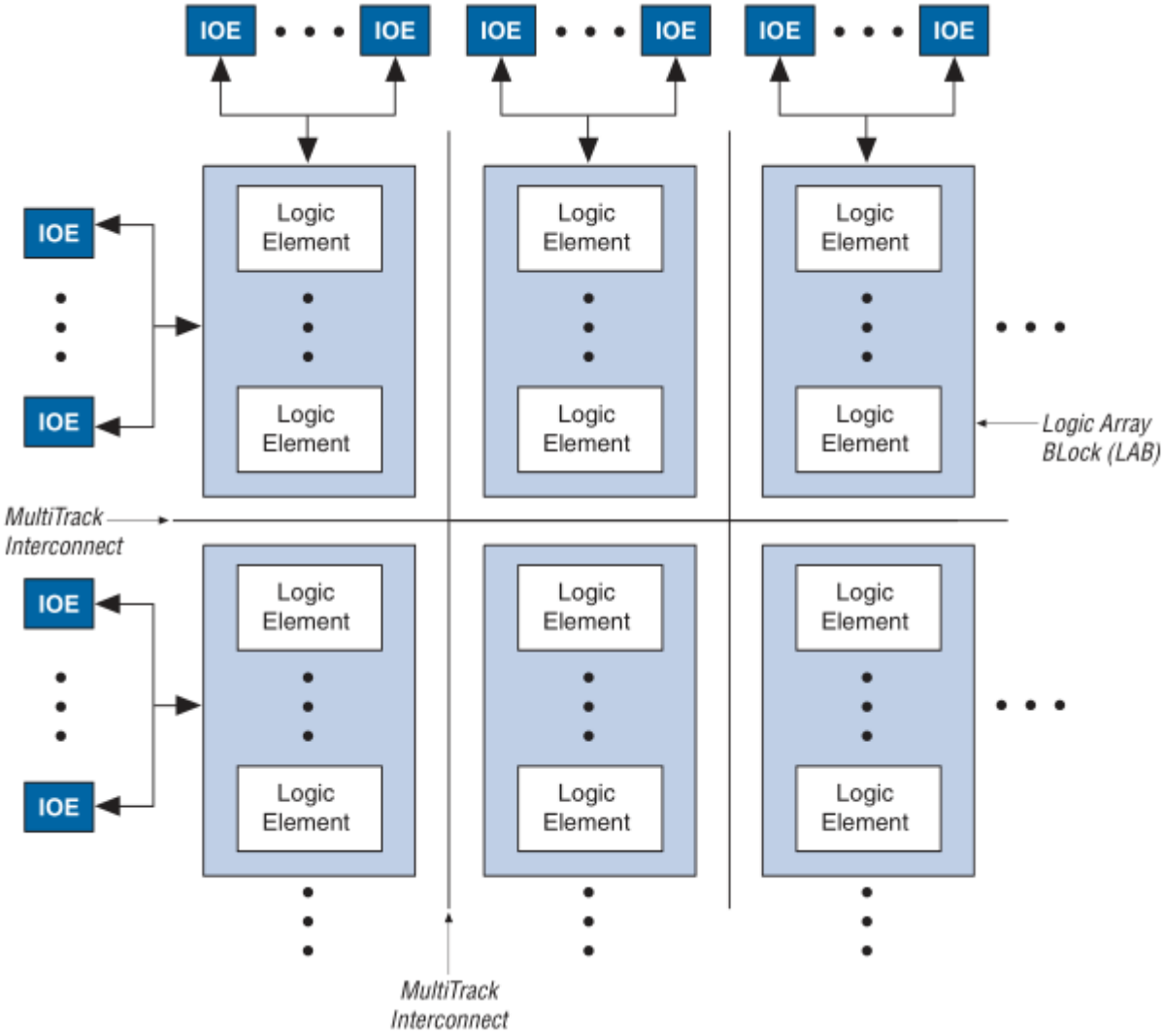


THE ALTERA MAX V

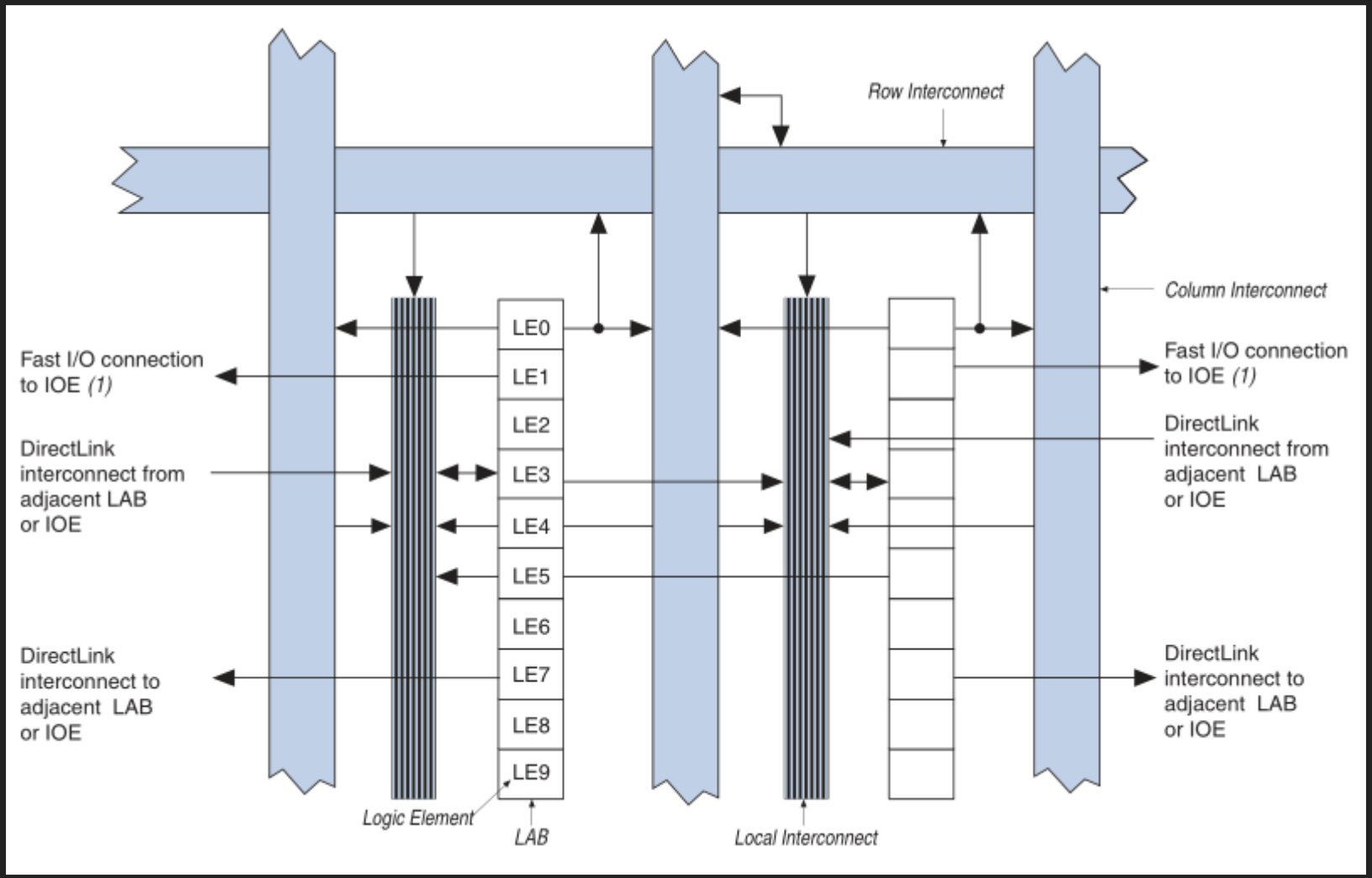
MAX V OVERVIEW

- MAXV 5M570Z
- ~ 570 LEs
- 159 user I/O pins

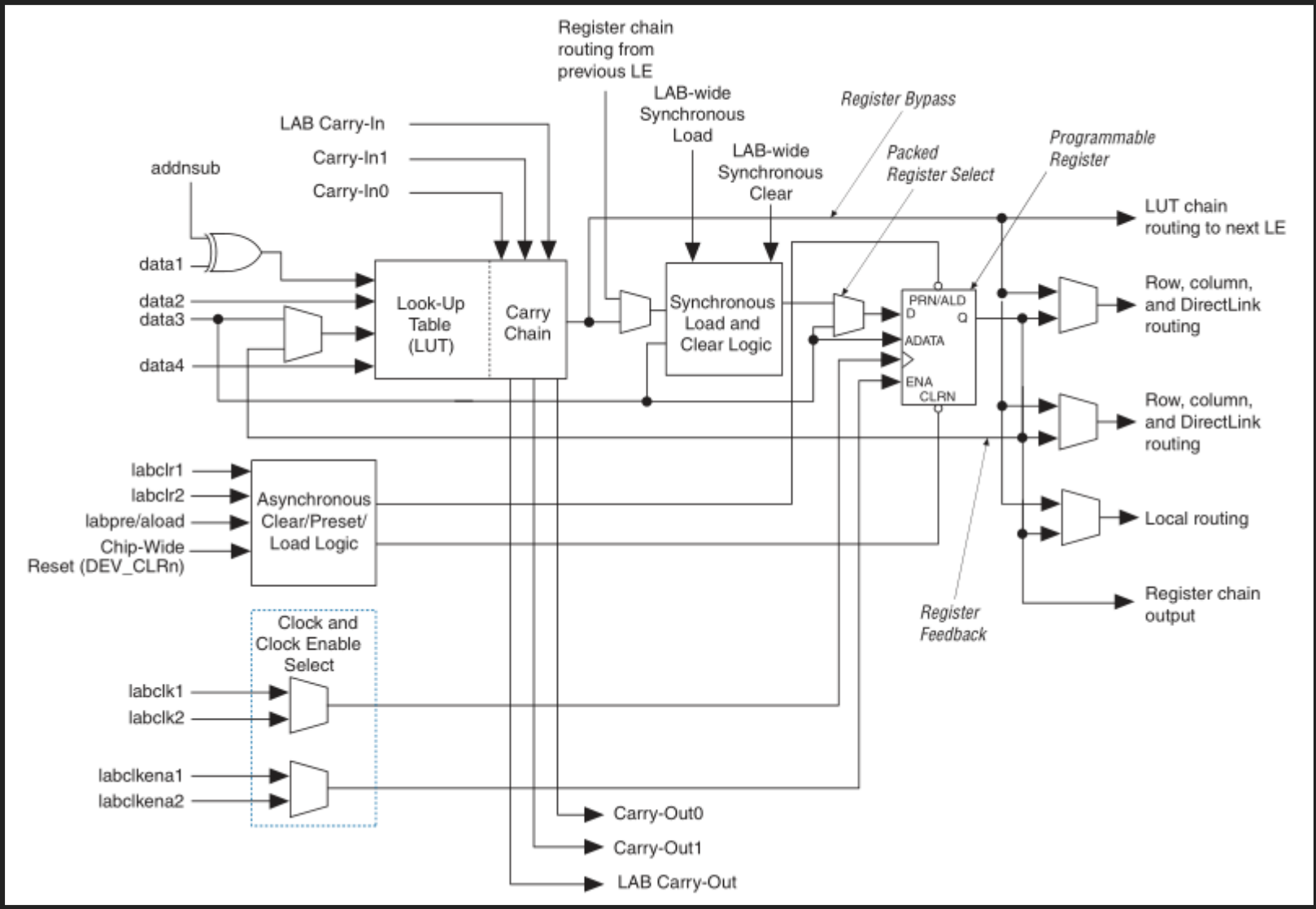
MAX V HIGH LEVEL OVERVIEW



MAX V INTERNALS: LAB STRUCTURE



MAX V INTERNALS: LE STRUCTURE



TO SUM UP

- The device is composed of a matrix of LABs
- LABs are interconnected by the MultiTrack Interconnect
- A LAB is composed of 10 LEs, a carry chain and a local interconnect
- Locality is taken into account during place & route

MAX V BITSTREAM ANALYSIS

GOALS

- What bits do we need to set to configure a given cell ?
- What bits do we need to set to route two given cells together ?

THE PROGRAMMER OBJECT FILE

- Contains the bitstream
- Result of `quartus_asm`
- Information is grouped into packets
- Each packet is composed of a header followed by data

```
struct packet_hdr {
  uint16_t tag;
  uint32_t length;
}__attribute__((packed));
```

THE PROGRAMMER OBJECT FILE

top.pof

```
00000000 50 4F 46 00 00 00 01 00 07 00 00 00 01 00 4D 00 00 00 51 75 61 72 74 75 POF.........M...Quartu
00000018 73 20 50 72 69 6D 65 20 50 72 6F 67 72 61 6D 6D 65 72 20 56 65 72 73 69 s Prime Programmer Versi
00000030 6F 6E 20 31 35 2E 31 2E 32 20 42 75 69 6C 64 20 31 39 33 20 30 32 2F 30 on 15.1.2 Build 193 02/0
00000048 31 2F 32 30 31 36 20 53 4A 20 4C 69 74 65 20 45 64 69 74 69 6F 6E 00 02 1/2016 SJ Lite Edition..
00000060 00 0D 00 00 00 35 4D 35 37 30 5A 46 32 35 36 43 35 00 03 00 09 00 00 00 .....5M570ZF256C5.......
00000078 55 6E 74 69 74 6C 65 64 00 05 00 02 00 00 00 00 00 11 00 0C 36 00 00 00 Untitled.............6...
00000090 00 00 00 00 00 00 B0 01 00 01 00 FE FF FF FF FD FF FF FF FE FF FF 7F F5 ........................
000000a8 7F FE FF FE FF FF FF FD FF F9 FF FE FF FF FF FD FF FF FF FE FF FF FF FD ........................
000000c0 EF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD ........................
000000d8 FF FF FF FE FF FF FF FD FF 7F 66 F6 CF CC FC FD 7F F6 9F CE CC FF CF 65 ..........f............e
000000f0 FE B9 99 FE FF 99 F9 3D 33 F3 FF FE FF FF FF FD FF FF FF FE FF FF FF DD .......=3..............
00000108 F7 FB FF FE FF EF FF FD FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD ........................
00000120 7F FF FF FE FF FF D7 FD FF F7 FF FE 7F FD FF FD FF FF FF FE FF FF FF FD ........................
00000138 FF 7F FF FE FF FF FF FD FF FF FF FE FF FF FF FD FB FF FF FE FF FF FF FD ........................
00000150 FF FF FF 7E FF FF FF FD FF FF FF FE FF FF FF FD FF FD FF FE FF FF DF FD ...~....................
00000168 FF FF FF FE FF FF FF FD FF FF FF 7E F7 FF FF FD FF FF FF FE FF FF FF FD ..........~.............
00000180 FF FF FF FE FF FF FF FD F7 DF FF FE FF FF FF FD FF FF FF 7E FF FF FF FD ..................~....
00000198 FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD ........................
000001b0 FF FF FF 7E FF FF FF FD FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD ...~....................
000001c8 FF DF FF FE FF FF FF FD FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD ........................
000001e0 FF FF FF FE FF FF FF FD F7 FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD ........................
000001f8 FF FF FF FE FF FF FF FD FF FF FF FE FF FF FF FD FF FF FF FE FB FF FF FD ........................
00000210 FF FF 7F FE FF FF FB FD FF FF 7E FE FF FF FF FD FF FF FF FE FF FF FE FD ..........~.............
00000228 FF FF FF FE FF FF FF FD FF FF FF 7E FF FF FF FD FF FB FF FE FF FF FF FD ...........~............
00000240 FF FF DF EE 7F FF FF FD FF FD FF F6 FF FF FF FD FF FF FF 7E FF FF FF FD ..................~....
00000258 FF FF EE DE FF FF FF FD FF FF FF FE FF FF FE FD FF FF FF FE FB FF FF FD ........................
00000270 FF FF BF FF FF FF F7 FD FF FD FF FF FF FF FF FD FF FF FF FF FF FF FF FD
```

WHAT NOW ?

1. Write a small piece of verilog
2. Generate a bitstream
3. Incrementally modify it
4. Observe changes in the bitstream
5. ???
6. What could go wrong ?

THE PLACE AND ROUTE STAGE

- Placement:
  - Decide where to place each electronic component
- Routing:
  - Wire them together

P&R IS NON-DETERMINISTIC

- Uses simulated annealing:
  - Move nodes *randomly*
  - High temperature: allow bad moves
  - Lower temperature: less bad moves are allowed
  - Slowly cool down temperature

HOW TO DEAL WITH IT

1. Generate a bunch of different bitstreams
2. Know which resources are used by each of them
3. Cross-correlate the bits used for a given resource

THE ROUTING CONSTRAINTS FILE

aka. our not-so-secret weapon

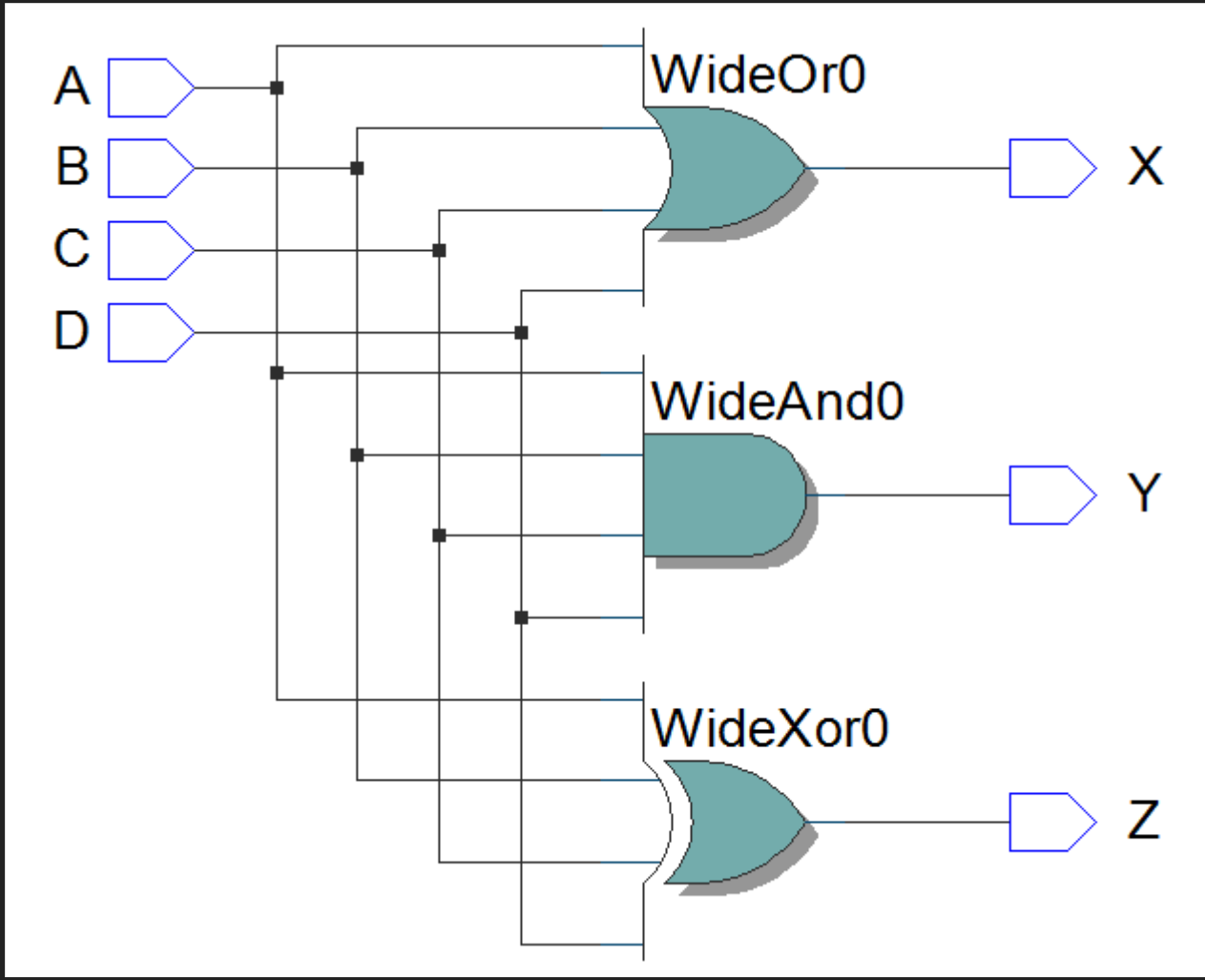
```
quartus_cdb --back_annotate=routing <my_project>
```

- Contains information about pin, cell and routing assignments
- Back-annotated after P&R

A TRIVIAL DESIGN

```
module demo(input A, B, C, D, output X, Y, Z);
        assign X = |{A, B, C, D};
        assign Y = &{A, B, C, D};
        assign Z = ^{A, B, C, D};
endmodule
```

GIVES US THIS NETLIST



AND THEN THIS RCF

```
section global_data {
    rcf_written_by = "Quartus Prime 15.1 Build 193";
    device = 5M570ZF256C5;
}
signal_name = A {    #IOC_X0_Y4_N0
    IO_DATAIN:X0Y4S0I0;
    label = Label_LOCAL_INTERCONNECT:X1Y4S0I23, LOCAL_INTERCONNECT:X1Y4S0I23;
    dest = ( WideOr0~0, DATAA ), route_port = DATAB; #LC_X1_Y4_N3

    branch_point = Label_LOCAL_INTERCONNECT:X1Y4S0I23;
    dest = ( WideAnd0~0, DATAA ), route_port = DATAB; #LC_X1_Y4_N7

    branch_point = Label_LOCAL_INTERCONNECT:X1Y4S0I23;
    dest = ( WideXor0~0, DATAA ), route_port = DATAB; #LC_X1_Y4_N4
}
...
```

PARSING THE RCF

Using pyrser to do the job

```
[{'device': '5M570ZF256C5',
  'dst': [],
  'name': 'global',
  'rcf_written_by': '"Quartus Prime 15.1 Build 193"'},
 {'IO_DATAIN': routing_coord(x=0, y=4, s=0, i=0),
  'branch_point': 'Label_LOCAL_INTERCONNECT',
  'dst': [{'block_name': 'WideOr0~0',
           'coord': back_annot(type='LC', x=1, y=4, n=3),
           'label': 'Label_LOCAL_INTERCONNECT:X1Y4S0I23, '
                    'LOCAL_INTERCONNECT:X1Y4S0I23',
           'port': 'DATAA',
           'route': 'DATAB'},
          {'block_name': 'WideAnd0~0',
           'branch_point': 'Label_LOCAL_INTERCONNECT:X1Y4S0I23',
           'coord': back_annot(type='LC', x=1, y=4, n=7),
           'port': 'DATAA',
           'route': 'DATAB'},
          {'block_name': 'WideXor0~0',
           'branch_point': 'Label_LOCAL_INTERCONNECT:X1Y4S0I23',
           'coord': back_annot(type='LC', x=1, y=4, n=4),
           'port': 'DATAA',
           'route': 'DATAB'}],
  'label': 'Label_LOCAL_INTERCONNECT',
  'name': 'A',
  'src': back_annot(type='IOC', x=0, y=4, n=0)},
...
]
```

FUZZING AT THE VERILOG LEVEL

- Generate random designs using the maximum number of user pins
- Reuse some icefuzz scripts

```
module top(input p0, input p1, input p2, input p3, input p4, input p5, input p6, input p7, input p8, input p9, output p
 localparam [15:0] p10_lut0 = 16'd 19820;
  wire p10_in0 = p10_lut0 >> {p16, p17, p14, p16};
  localparam [15:0] p10_lut1 = 16'd 9542;
  wire p10_in1 = p10_lut1 >> {p9, p2, p2, p8};
  localparam [15:0] p10_lut2 = 16'd 23726;
  wire p10_in2 = p10_lut2 >> {p17, p17, p12, p18};
  localparam [15:0] p10_lut3 = 16'd 43527;
  wire p10_in3 = p10_lut3 >> {p14, p17, p18, p15};
  localparam [15:0] p10_lut4 = 16'd 38962;
  wire p10_in4 = p10_lut4 >> {p6, p9, p12, p12};
  localparam [15:0] p10_lut5 = 16'd 60603;
  wire p10_in5 = p10_lut5 >> {p3, p2, p4, p2};
  localparam [15:0] p10_lut6 = 16'd 30058;
  wire p10_in6 = p10_lut6 >> {p2, p4, p3, p9};
  localparam [15:0] p10_lut7 = 16'd 11382;
  wire p10_in7 = p10_lut7 >> {p17, p17, p18, p14};
  assign p10 = ^{p10_in1, p10_in5, p10_in7, p5, p7, p8, ~p10_in0, ~p10_in2, ~p10_in3, ~p10_in4, ~p10_in6, ~p11, ~p9};
  localparam [15:0] p13_lut0 = 16'd 28835;
  wire p13_in0 = p13_lut0 >> {p14, p11, p12, p11};
  localparam [15:0] p13_lut1 = 16'd 43306;
  wire p13_in1 = p13_lut1 >> {p12, p12, p15, p9};
...
```

ASSUMPTIONS

- The default value for the cell configuration bits is 1
  - work with the bitwise inverse of the bitstream
- The enabled routes of a cell define its configuration bits

POPULATING THE DATABASE

- Associate the sample bitstream to its RCF
- For each signal defined inside the RCF:
  - Add source cell to the database if not found
  - For each of its destinations:
    - Add dest cell to the database if not found
    - Add route to database associated to this sample

ANALYZING INTER-CELL CONFIGURATION

- For a given route R:
  - A := all bitstreams who use R
  - B := ~A
  - I := intersection of each element in A
    - "the bits who are set in all bitstreams that use R"

ANALYZING INTER-CELL CONFIGURATION: A BETTER WAY

- For a given route R:
  - A := all bitstreams who use R; B := ~A
  - I := intersection of each element in A
    - "the bits who are set in all bitstreams that use R"
  - J := union of the complements of each element of B
    - "the bits who are not set in at least one of the bitstreams that doesn't use R"
  - Configuration bits for R are the intersection of I and J
    - "the bits who are set in all bitstreams that use R and are not used in at least one bitstream that doesn't use R"

CONCLUSION

- We overcomed the pain caused by the P&R by using the RCF file
- We are able to isolate the bits used by a route in the bitstream
- Not quite reliable. We probably need more bitstreams

NEXT STEPS

- Correlate the specific interconnects used
- Retrieve the content of a LUT
- RAM cells
- Probably more

RESSOURCES

- Altera documentation: https://www.altera.com/en_US/pdfs/literature/hb/max-v/max5_handbook.pdf
- From the bitstream to the netlist: http://www.fabienm.eu/flf/wp-content/uploads/2014/11/Note2008.pdf
- Icestorm project: http://www.clifford.at/icestorm/

QUESTIONS?