The FABulous Open Source eFPGA Framework

Dirk Koch
Heidelberg University, The University of Manchester
dirk.koch@ziti.uni-heidelberg.de
The FABulous Framework

- Fully integrated framework for eFPGAs
- Uses many projects:
  - Yosys & ABC
  - nextpnr
  - OpenLANE
  - VPR
  - OpenRAM
  - Verilator

This talk

Fabric description (layout & wires)

FABulous (synthesis & mapping)

 ASIC RTL & constraints

ASIC backend (Cadence, OpenLANE)

ASIC (GDS)

Fab (TSMC180, Sky130)

Primitive library

Model (architect. graph)

User design optimization

Yosys and ABC (synthesis & mapping)

Json (mapped netlist)

nextpnr (place & route)

FASM (routed netlist)

BitMan (bitstream assembly)

User bitfile

Cost, performance

Physical optimization

Stats (utilization, routability, etc.)

Fabric architecture optimization

User Verilog (benchmark)
The FABulous Framework

- FABulous eFPGA generator
- ASIC RTL and constraints generation
- Generating models for nextpnpr/VPR flows
- FPGA emulation
- Virtex-II, Lattice clones (patent-free!)

See our FPGA 2021 paper „FABulous: An Embedded FPGA Framework“
The first open-everything FPGA

- Built using open tools
  (Yosys, OpenLane, Verilator…)
- Open PDK
  (Skywater 130 process)
- Google Shuttle (MPW5):
  https://github.com/nguyendao-uom/open_eFPGA
Basic concepts

- Basic tiles have same height, but type-specific width (for logic tiles, DSPs, etc.)
- Adjacent tiles can be fused for more complex blocks (see the DSP example)
Let's build a small eFPGA: Fabric Definition

- 4 x register file, 2 x DSPs, 8 x LUT-tiles (CLB), I/Os left and right,
- A fabric is modelled as a spreadsheet (tiles are references to tile descriptors)
Let's build a small eFPGA: Tile Definition

- Wires
- Primitives (basic elements)
- Switch matrix

<table>
<thead>
<tr>
<th>#</th>
<th>TILE</th>
<th>LUT4AB</th>
</tr>
</thead>
<tbody>
<tr>
<td>121</td>
<td>TILE</td>
<td>LUT4AB</td>
</tr>
<tr>
<td>122</td>
<td>#direction</td>
<td>source   X-offset Y-offset destination wires</td>
</tr>
<tr>
<td>123</td>
<td>NORTH</td>
<td>N1BEG</td>
</tr>
<tr>
<td>124</td>
<td>NORTH</td>
<td>N2BEG</td>
</tr>
<tr>
<td>125</td>
<td>NORTH</td>
<td>C0</td>
</tr>
<tr>
<td>126</td>
<td>EAST</td>
<td>E1BEG</td>
</tr>
<tr>
<td>127</td>
<td>EAST</td>
<td>E4BEG</td>
</tr>
<tr>
<td>128</td>
<td>SOUTH</td>
<td>S1BEG</td>
</tr>
<tr>
<td>129</td>
<td>SOUTH</td>
<td>S2BEG</td>
</tr>
<tr>
<td>130</td>
<td>WEST</td>
<td>W1BEG</td>
</tr>
<tr>
<td>131</td>
<td>WEST</td>
<td>W4BEG</td>
</tr>
<tr>
<td>132</td>
<td>JUMP</td>
<td>J_BEG</td>
</tr>
<tr>
<td>133</td>
<td>BEL</td>
<td>LUT4.vhdl</td>
</tr>
<tr>
<td>134</td>
<td>BEL</td>
<td>LUT4.vhdl</td>
</tr>
<tr>
<td>135</td>
<td>BEL</td>
<td>LUT4.vhdl</td>
</tr>
<tr>
<td>136</td>
<td>BEL</td>
<td>LUT4.vhdl</td>
</tr>
<tr>
<td>137</td>
<td>BEL</td>
<td>MUX8LUT.vhdl</td>
</tr>
<tr>
<td>138</td>
<td>MATRIX</td>
<td>LUT4AB_switch_matrix.vhdl</td>
</tr>
<tr>
<td>139</td>
<td>End</td>
<td>TILE</td>
</tr>
</tbody>
</table>
Describes the adjacency in a symbolic way <mux_output>,<mux_input>

Alternatively adjacency matrix
FABulous: an Embedded FPGA Framework

FABulous is designed to fulfill the objectives of ease of use, maximum portability to different process nodes, good control for customization, and delivering good area, power, and performance characteristics of the generated FPGA fabrics. The framework provides templates for logic, arithmetic, memory, and I/O blocks that can be easily stitched together, whilst enabling users to add their own fully customized blocks and primitives.

The FABulous ecosystem generates the embedded FPGA fabric for chip fabrication, integrates SymbiFlow toolchain release packages, deals with the bitstream generation and after fabrication tests. Additionally, we will provide an emulation path for system development.

This guide describes everything you need to set up your system to develop for FABulous ecosystem.
Tile-based Design in FABulous

- Tile-based design
  - Consistent timing
  - Reliable implementation
  - Reuse optimization effort
    → Share tiles as GDS macros
      (Thanks Google for the Shuttle Program)
- Routing fabric is dominating area
  → Simple optimizations gain most optimization potential
    - Use cheap configuration memory
      (custom cells or latches)
    - Use custom multiplexers
Define CLB (pragmatically) as smallest square tile

Defines the grid height

Widths depend on tile-type (CLB, REG, DSP, etc.)

Replace standard cell multiplexers with custom mux-4

\[
A_{\text{std-cell}} - A_{\text{c-mux4}} \times N = (33.8 \, \mu m^2 - 17.5 \, \mu m^2) \times 376 = 6,116 \, \mu m^2
\]

### Standard cell

<table>
<thead>
<tr>
<th></th>
<th>height</th>
<th>width</th>
<th>area</th>
<th>util.</th>
<th></th>
<th>area</th>
<th>util.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLB</td>
<td>219 , \mu m</td>
<td>219 , \mu m</td>
<td>47,961</td>
<td>81.8%</td>
<td></td>
<td>46,225</td>
<td>60.7%</td>
</tr>
<tr>
<td>REG</td>
<td>219 , \mu m</td>
<td>214 , \mu m</td>
<td>46,866</td>
<td>84.1%</td>
<td></td>
<td>46,655</td>
<td>64.3%</td>
</tr>
<tr>
<td>DSP</td>
<td>443 , \mu m</td>
<td>185 , \mu m</td>
<td>81,955</td>
<td>80.9%</td>
<td></td>
<td>81,780</td>
<td>56.7%</td>
</tr>
</tbody>
</table>

### Custom mux-4

Observation:

- No area improvement
- Instead: core utilization went down

→ Congested tile routing
In short

improvement

expected

actual

work
The configuration bit cells may induce inferior placement of multiplexers

We can remap configuration bits $\rightarrow$ requires remapping of the bitstream (trivial)
Optimization: Bitstream Remapping

- Change order of flip-flops in the chain
- Change latch-to-gridpoint mapping (with frame-data and frame-strobe gridpoints)
We use Google's Operations Research tools to compute the grid points (https://github.com/google/or-tools)
Optimization: Bitstream Remapping

frame data before

frame data after

frame strobe before

frame strobe after
Bitstream Remapping – Results

- FBC std cell (47,961)
- FBC cust Mux4 (46,225)
- FBC full metal (40,804)
- FBC opt (37,249)
- FBC full metal opt (36,100)
- SRC_DFF (64,009)
- SRC_DFF_opt (62,001)
- SRC_CFF (52,441)
- SRC_CFF_opt (38,416)
- pin swap (37,249)
- comb. opt. (36,481)

Baseline: 70,000

+3.8%

-11.7%

-19.4%

-21.9%

+38.5%

+34.1%

+13.4%

-16.9%

-19.4%

-21.7%
FABulous versus OpenFPGA (on Sky130)

- FABulous and OpenFPGA have a Google Shuttle2 submission
- ~ same physical impl. problem
- OpenFPGA CLBs are 17% bigger
- New optimizations gave us further 21.7% in density on the same netlist!
Sky130 with CLBs, DSPs, RegFiles, BRAMs
Google Shuttle - MPW-2
(can implement RISC-V)

https://github.com/nguyendao-uom/eFPGA_v3_caravel

Dual-Ibex-Crypto-eFPGA
Google Shuttle - MPW-4
(custom instructions, T-shaped fabric)

https://github.com/nguyendao-uom/ICESOC
FABulous Chip Gallery

Open ReRAM FPGA test chip

- Sky130, Google Shuttle
- Just enough logic to send „Hello World“ to a UART
- Different configuration modes

Possible advantages of ReRAM FPGAs

- Security (user circuit is encoded in resistive states)
- Reliability (ReRAM is radiation hard)
- Probably density
- Instantaneous on
- CMOS friendly
The FABulous eFPGA Framework – Wrap-up

- Heterogeneous (FPGA) fabric (DSBs, BRAMs, CPUs, custom blocks)
  - Multiple tiles can be combined for integrating more complex blocks
  - Custom blocks can be instantiated directly in Verilog and are integrated in Yosys, VPR/nextpnr CAD tools (Synthesis, Place&Route) (as primitive blocks)
- Support for dynamic partial reconfiguration (some elements of XC6200, like wildcard configuration)
- Configuration through shift registers or latches (or custom cells)
- Support for custom cell primitives (passgate multiplexers)
- Good performance / area / power figures (about 1.3x worse than Xilinx) (could be narrowed down through customization)
- Usable by FPGA users (you don’t have to be an FPGA architect) → there are FPGA classics that we have/will clone
- ToDo: multiple clock domains, mixed-grained granularity, more ReRAM, processes, ...
FABulous Team

People:
Nguyen Dao  nguyen.dao@manchester.ac.uk
Jing Yu      jing.yu@manchester.ac.uk
Bardia Babaei bardia.babaei@postgrad.manchester.ac.uk
King Chung  king.chung@student.manchester.ac.uk
Tuan La      tuan.la@manchester.ac.uk
Andrew Attwood a.j.attwood@ljmu.ac.uk
Bea Healy   tabitha.healy@student.manchester.ac.uk
Dirk Koch   dirk.koch@ziti.uni-heidelberg.de

See our projects under:
https://github.com/FPGA-Research-Manchester

This work is kindly supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/R024642/1.
We also thank Google for sponsoring the Efabless Open MPW Shuttle Program.