

Customized eFPGAs with FABulous

dirk.koch@ziti.uni-Heidelberg.de



• 16K RISC-V Threads on a single FPGA

We also work on FPGA Hardware Security (not today)

- A hotspot design can burn
 100 mW / CLB (8 LUTs)
- Power-hammering
 potential of an AWS F1
 instance FPGA is >15KW
- Chip survived 40 W
 hotspot (...but up to 40%)
 aging in 1 week)
- We also design
 Hardware Trojans



RISC-V Processor running at 737 MHz (also not today)



Why embedded FPGAs

- Configurability is expensive!
- FPGAs are slower than ASICs
- FPGAs draw more power than ASICs
- Systems still need flexibility

Solution:

- Use soft logic only where it is needed
- Also: making chips get cheaper

(mask set cost 130nm: 150K\$; 28nm: ~1M\$)



Source: Wikipedia

You think you buy logic,

... but you pay for the routing

What about customizing your FPGA to your application?



Source: https://en.webfail.com/98841d83a5b

The FABulous Framework

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



What is FABulous offering?

- Fully integrated open-source FPGA framework with good quality of results (area & performance)
- Entirely open and free, including commercial use
- Supports custom cells (if provided) \rightarrow some tooling is on the way
- Supports partial reconfiguration
- Designed for ease of use while providing full control as needed
- Versatile
 - Different flows (OpenLane $\leftarrow \rightarrow$ Cadance) (Yosys/nextpnr $\leftarrow \rightarrow \forall PR$)
 - Easy to customize, including the integration of own IP

Let's build a small eFPGA: Fabric Definition

term		term	term	term		
IO Pin	REG (mem)	DSP DSP_top	LUT	LUT	CPU IO	
IO Pin	REG (mem)	DSP_bot	LUT	LUT	CPU IO	
IO Pin	REG (mem)	DSP DSP_top	LUT	LUT	CPU IO	
IO Pin	REG (mem)	DSP_bot	LUT	LUT	CPU IO	
	term	term	term	term		

• 4 x register file, 2 x DSPs, 8 x LUT-tiles (CLB), I/Os left and right,

Let's build a small eFPGA: Fabric Definition

	term	term		term	term				
IO Pin	REG (mem)	DSP	L	UT L		1			
		DSP_		A	B	С	D	E	F
IO Pin	REG (mem)	1		1 FabricB					
	(mom)	DSP_	2	NULL	N_term	N_term	N_term	N_term	NULL
10	REG	DSP	3	W_IO	RegFile	DSP_top	LUT4AB	LUT4AB	CPU_IO
Pin	(mem)		4	W_IO	RegFile	DSP_bot	LUT4AB	LUT4AB	CPU_IO
		DSP_	5	W_IO	RegFile	DSP_top	LUT4AB	LUT4AB	CPU_IO
10 D:	REG		6	W_IO	RegFile	DSP_bot	LUT4AB	LUT4AB	CPU_IO
Pin	(mem)	DSP	7	NULL	S_term	S_term	S_term	S_term	NULL
	term	term	8	Fabric	End				

• 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)

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) \rightarrow Supertile

Let's build a small eFPGA: Tile Definition



- Wires
- Primitives (basic elements)
- Switch matrix
- Configuration storage

Let's build a small eFPGA: Tile Definition



Configuration storage



Search

FABulous Chip Gallery



Sky130 with CLBs, DSPs, RegFiles, BRAMs Google Shuttle - MPW-2 (can implement RISC-V)

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

FABulous Chip Gallery



Sky130 with CLBs, DSPs, RegFiles, BRAMs Google Shuttle - MPW-2 (can implement RISC-V)

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



Our first (full) open-everything FPGA

- Flow runs through without manual intervention
- 672x LUTs, 7x DSPs, 7x BRAMs
 (88% of what we achieved with

Cadence Innovus)

- Process: Skywater 130 (~12mm²)
- get it here:

https://github.com/FPGA-Research/ open_eFPGA_v2



Chip Gallery – Open ReRAM FPGA test chip

Sky130, Google Shuttle MPW4https://github.com/nguyendaouom/rram_testchip

Posible advantages of ReRAM FPGAs

- Security (IPs defined by resistive states)
- Reliability (ReRAM is radiation hard)
- Probably density
- Instantaneous on



FABulous Chip Gallery



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

- connects two operands to the fabric and
- multiplexes results back from different slots
- eventually include some dela slots if

custom instruction is slower than CPU

Implementation



- Easy method: tab into the operands and multiplex in a result
 - \rightarrow may require different number clock cycles to evaluate



Reconfigurable Instruction Set Extensions

CPU ISAs intend to get bloated

 \rightarrow trend to feature-rich instruction sets and acceleration

→ creates a legacy problem

Crypto





FABulous Users

- Stanford University (TSMC 130 & TSMC 28)
- TU Graz (IHP 130 & Sky 130)

https://github.com/mole99/greyhound-ihp

- New York University (2x TSMC 28)
- University of Bristol (Sky 130)
- Berkely (TSMC 130)
- Fudan University (TSMC 180)







Outputs from eFPGA (16b counter) probed with oscilloscope



Next Steps

- Timing model extraction (for timing-driven Place&Route)
- Physical constraints

(bounding boxes and macro placement)

- More optimizations
- More tapeouts GF22, TSMC 16, IHP 130
- Open-everything RISC-V-eFPGA board
 - RISC-V (Ibex?) + MMU
 - 1K LUT-4, 8 DSPs, 8 BRAMs,
 - Debug infrastructure



Ch



\cap		i		6	~	÷	~		÷	
ч	u	I	C	ĸ	С	ι	a	I.	ι	

Building fabric

Fal	bric	definitio	n

Fabric ASIC implementation



 \sim

FABulous: an Embedded FPGA Framework

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.





 $Q \langle \rangle$

lii\ 🗊

... 🖂 🤭

150%



HeiChips Summer School (Aug. 4-8)

- I day posters, talks, keynote, hike
- 2 days classes (open-source tools)
- 2 days Hackathon
 - → with ASIC tapeout!







Customized eFPGAs with FABulous

People:

Nguyen Dao Ying Yu Myrtle Shah Bea Healy King Chung (Kelvin) Gavaskar K Gennadiy Knies Andrew Attwood Jonas Künstler Asma Mohsin Marcel Jung Gennadiy Knis Timo Braungardt Biswajit Kumar Sahoo Jakob Ternes MANCH<mark>Est</mark>ER 1824 EPSRC Engineering and Physical Sciences

The University of Manchester





GEFÖRDERT VOM Bundesministerium für Bildung und Forschung





Dirk Koch dirk.koch@manchester.ac.uk

See our projects under:

https://github.com/FPGA-Research

Reconfigurable Instruction Set Extensions

The FlexBex (lbex with eFPGA) approach:

We use the following instruction encoding:

eFPGA[result_select]d[delay] dest, RS1, RS2 // delay: 0...15 cycles

- Register manipulation instruction: dest \leftarrow RS1 OP RS2

FlexBex: A RISC-V with a Reconfigurable Instruction Extension **FPT 2020**

```
Inline assembly: int ina, inb, result;
                 ina = 10;
                 inb = 20;
                 asm volatile
                    "\_eFPGA3d2\__\%[z], \_\%[x], \_\%[y] \setminus n \setminus t"
                    : [z] "=r" (result)
: [x] "r" (ina), [y] "r" (inb)
```

FPGA Configuration (as used in FABulous)



The FABulous eFPGA Ecosystem

- 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"



FABulous versus OpenFPGA (on Sky130)



New optimizations gave us further 21.7% in density on the same netlist!

Basic concepts



- I/Os belong logically to the fabric but are physically routed to the surrounding
- Internal wires, buses, etc. are "just" wires at the border of the fabric

Tile-based Design in FABulous

Replace standard cell multiplexers with custom mux-4

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



No area improvement

Observation:

■ Instead: core utilization went down → Congested tile routing

In short







 We use Google's Operations Research tools to compute the grid points (https://github.com/google/or-tools)

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 (passtransistor multiplexers)
- Good performance / area / power figures (about 1.5x 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, ...

FPGA Basics – Logic

- Look-up tables (LUTs) are basically multiplexers selecting configuration latches storing a function as a simple truth table
- Configuration latches are usually written through the configuration port only
- In distributed memory options (LUT is used as a shift register or memory file, table is also writeable through the user logic)



FPGA Configuration

- Do not use shift register configuration
 - High power during configuration (thousands of bits)
 - Configuration only valid if completely shifted in (transient short-circuits or ring-oscillators)
 - Cannot do "real" partial reconfiguration (static routes through reconfigurable regions)
 - Too expensive (shift registers need flip flops, frame-based configuration can do with latches)





FPGA Basics – Routing (Virtex-II style)



- AMD/Intel use multiple levels of one-hot encoded routing with pass-transistors
- Multiple activated inputs can cause short-circuit situations
 → this is why you should blank a region before overwriting it with a new module
 → less of a problem for encoded bitstreams (not one-hot encoded)

Tile-based Design in FABulous

Replace standard cell multiplexers with custom mux-4

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

		Standa	Custom mux-4			
	height	width	area	util.	area	util.
CLB	219µm	219µm	47,961	81.8%	46,225	60.7%
REG	219µm	214µm	46,866	84.1%	46,655	64.3%
DSP	443µm	185 µm	81,955	80.9%	81,780	56.7%

Observation:

- No area improvement
- Instead: core utilization went down
- \rightarrow Congested tile routing

In short



- The configuration bit cells may induce inferior placement of multiplexers
- We can remap configuration bits \rightarrow requires remapping of the bitstream (trivial)

 We use Google's Operations Research tools to compute the grid points (https://github.com/google/or-tools)

