FPGA Ignite 2024 Scratch

This file is intended to provide live support during the Summer School and will probably grow throughout the event.

VM Login:
Username: user
Password: FPGAIgnite
root password vm: FPGAIgnite

OpenLane Instructions

1. In order to install a viewing tool (klayout) , change directory to openlane and run
   sudo apt-get install klayout
2. Without sudo access, copying and creating files within the openlane directory does not work. Make sure to use sudo for this.
   sudo cp .. .. (for copying)
   ***sudo touch .. (for creating files*)**
3. Viewing def files in kalyout is not possible.
4. Non-interactive flow will not work. It's recommended to use the interactive one.

OPENRAM IP/MACRO

Skywater has several RAM IP's that you can use. They fall into the following categories;

- Single port with byte write SRAM suitable for RISC-V and other processor main memory.
- Pseudo-dual port SRAM (one write, one read) suitable for FIFOs.
- True dual port SRAM (two read/write ports) suitable for high speed data sharing between two devices.
- Dual access SRAM (one read/write, one read port) suitable for applications which need to read two data values every cycle (such as register files).

Currently supported sizes are 1kbytes and 2kbytes. webiste

How to use SRAM IP/MACRO

Detailed Guide

However there are few things that need to be considered.

1. Our **PDK path is /root/.volare/sky130A.**
2. Here is a list of configuration variables you can play with.
3. `VERILOG_FILES_BLACKBOX` `represents` Black-boxed, Verilog files where the implementation is ignored. Useful for pre-hardened macros you incorporate into your design, used during synthesis and OpenSTA. `/// sta-blackbox` can be added to a file in order to skip that file while doing STA. This will blackbox all the modules defined inside that file. It is recommended to provide a gatelevel netlist whenever possible to do full STA.
4. There is a false positive by `check_power_grid`. There are vias at the corners of the power ring of the macro, where the tool is complaining, and it seems to be connected. Continue the flow with `FP_PDN_CHECK_NODES` disabled and see if you get any other errors.

Testbench

1. Add $dumpfile("filename.vcd") and $dumpvars(0,tb_name) in the initial block.
2. Then run the following commands for simulation

   sudo iverilog -o tb_pre.out something.v testbench.v
   sudo vvp tb_pre.out
   Gtkwave filename.vcd
3. For after layout simulation again add $dumpfile and $dumpvars.
4. Take the powered netlist file which is written during run_lvs step.
5. Change directory to ***/root/.volare/sky130A/libs.ref/sky130_fd_sc_hd/verilog folder***
6. Copy sky_130_fd_sc_hd.v and primitives.v file in your simulation folder.
7. Add following lines to sky130_fd_sc_hd.v file

   `define UNIT_DELAY #1
   `define USE_POWER_PINS
   `define FUNCTIONAL
8. Then run the following commands for simulation

sudo iverilog -o tb_pre.out something.v sky130_fd_sc_hd.v primitives.v testbench.v
sudo vvp tb_pre.out
Gtkwave filename.vcd

MACRO_INSTANTIATION in TOP_LEVEL

Project: Posit Coprocessor

Team participants
Tim Fernandez-Hart - Brunel University London, UK
Ekrem Altuntop - Heidelberg University, Germany
Adrian Pitterling - TU Ilmenau and IMMS GmbH, Germany
Sayed Mohammad Tariful Azam - Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau
Jonas Lienke - IMMS GmbH, Germany

Short abstract / idea
The project brief is to implement a posit-enabled ALU co-processor that can perform addition, division and multiplication using posit arithmetic.
Posits are a new number format based on Floating-Point (FP). As such, they contain a fraction that is scaled by some number. But unlike FP, this scaling value is the product of two numbers encoded within a posit bit string, termed the regime and the exponent. While most studies show that posit arithmetic is more accurate than floating-point at the same bit width, this comes at a cost. The regime bits are dynamically sized during run time which greatly complicates their implementation and significantly impacts the size of a Posit Arithmetic Unit (PAU). However, hardware comparison studies rarely take the additional accuracy of posits into account, preferring to compare equal bit-widths FP and posits. This ignores one of the main advantages of posits and biases any conclusions.
In this project, we hope to address his question. Can an n-bit PAU outperform an m-bit FPU, where n<m in resource usage, while remaining competitive in terms of accuracy?

Used area
ALL OF IT
(really)

Used pins
32 CVXIF Instruction Pins + RS1 + RS2 + DST

Used memory

Some

Project: VGA Ignite

Team

Hugh Squires-Parkin, Newcastle University
Chao Qian, University of Duisburg-Essen
Florian Feltz, Mannheim University of Applied Sciences
Mengbi Yu, Heidelberg University
Justin Cott, Karlsruhe Institute of Technology
Al-Harith Farhad, DFKI/ RPTU
Jan Zielasko, DFKI CPS Bremen
Qaisar Farooq, University of Turin
Jelle Biesmans, KU Leuven (Online)
Asma Mohsin, Universität Heidelberg

Short abstract / Idea

An on-chip VGA driver with a simplified color encoding scheme and a separate pixel processing unit (PPU) that processes pixel data and creates multiple shader graphic effects based on a mode register. The data transfer protocols are done using our own light-weight version of Wishbone bus, named “Boneless”.

Resources used

Used area

tba (probably quite small) - First OpenLane flow run for VGA_DRIVER : 60x60 um2

Used pins

We would like to use 8 shared I/O pins of the chip’s pins for output to the hardened VGA driver

Used memory

Project: CXBex

Team <full name, affiliation>
Bea Healy - University of Cambridge
Kelvin Chung - University of Manchester
Meinhard Kissich - Graz University of Technology
Emil Cozac - NXP Toulouse
Jonas Kuenstler - Heidelberg University
Ron Sass - UNC Charlotte
Guy Lemieux - University of British Columbia

Guy's students - not at FPGA Ignite:
Brandon Freiburger - University of British Columbia
Joseph Maheshe - University of British Columbia

Short abstract / idea
CXBex iterates on FlexBex, an open-everything eFPGA fabric that packages an Ibex RISC-V core with a reconfigurable instruction extension. CXBex integrates the CX interface, a protocol to support any combination of RISC-V instruction extensions without conflict or modification to the extension, between the core and the eFPGA.

Used area

Used pins

Used memory

Project: A Custom Neural Network RISC-V Processor for Machine Learning

Team <full name, affiliation>
Duc Dung Vu - Macquarie University, Australia
Vu Hoang Thang Chau - Macquarie University, Australia

Short abstract / idea

This project presents the design and implementation of a custom RISC-V processor with an extended instruction set tailored for efficient fixed-point mathematical computations for neural network. Leveraging the CORDIC (COordinate Rotation DIgital Computer) algorithm, we introduce three specialized instructions to accelerate the computation of the sigmoid, square root, and hyperbolic tangent (tanh) functions. The sigmoid function is crucial for machine learning applications, particularly in neural network activation, while the square root function is essential in geometric computations and signal processing. The tanh function, another neural network activation function, is advantageous for providing zero-centered output, aiding faster convergence during training. By integrating these functions directly into the processor's instruction set, our design significantly reduces computational overhead, enhancing performance in machine learning, digital signal processing, and control systems. (ChatGPT assisted writing, please don't judge us) 🙂

Used area

Used pins
Maximum I/O 68 pins:

- 32 x2 Data pins
- 1x 2 valid pins
- 1x 1 ready pin
- 1x 1 mode pin.

Used memory
Maximum 10k LUTs and 25k register (32 bits)

Project: THE RING

Project Name
THE RING: True Hardware Embedded Random eight-bit INteger Generator

Team <full name, affiliation>
Matthias Musch, ZF Friedrichshafen AG
Jan Steinmann, Heidelberg University
Jakob Ternes, Heidelberg University

Short abstract / idea
Our project implements a random number generator, which exploits electrical noise (jitter) as a source of entropy, in hardware.
The entropy source for the random bit generation is a cluster of 8 pairs of ring oscillators. Each pair consists of ring oscillators with 3 and 5 inverters, respectively.
The two output bits are pairwise XORed and subsequently sampled with the 50MHz IO Clock of the chip to generate truly random sequences which are accessible via the *d_out* pins (Fig. 1).
This post-processing procedure is based on a design described by Erdinc Acaroglu et. al in their publication “A novel chaos-based post-processing for TRNG”

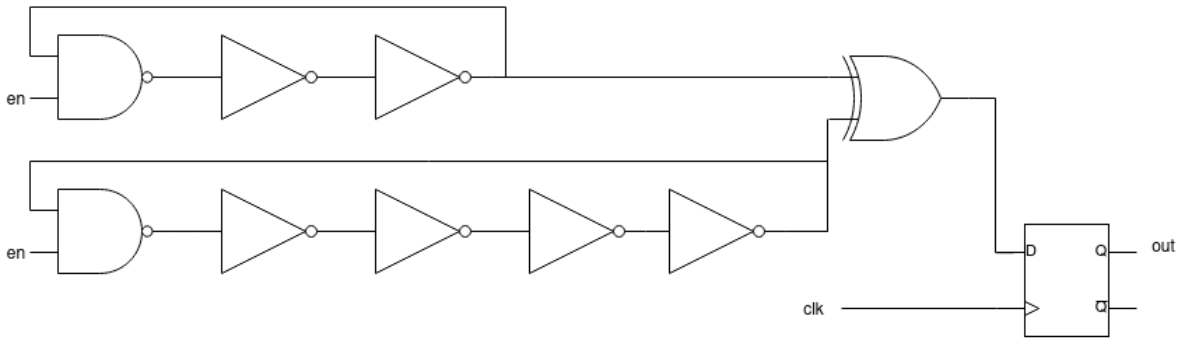

Fig. 1: Generation and sampling of a single random bit

Our process of sampling ring oscillators as an entropy source for random number generation is derived from a design described by [Xinzhe Wand et. al] in “10-Gbps true random number generator accomplished in ASIC”.

We also included instructions for testing the entropy level of the random generation which are based on the NIST SP 800-90B standard, which was published by [Meltem Sönmez Turan et. al., 2018].

Used area
60x60µm²

Used pins

- [Input] 1-bit Enable
- [Input] 1-bit Clock (assumed 50MHz - does not really matter)
- [Output] 8-bit Random number “d_out”

Used memory
none

Project: Image resolution upscaling

Project Name
Image resolution upscaling: byte-wise operation parallelism using custom instructions

Team <full name, affiliation>
Kagan Dikmen, TU Munich
Tarik Ibrahimović, Chili.CHIPS*ba, University of Sarajevo
Farhad Ebrahimiazandaryani, FAU Erlangen-Nurenberg
Arefeh Mahdavi, University of Tehran, Iran

Short abstract / idea
The image's initial resolution can be upscaled to a higher one by using existing information contained in the initial image to predict and interpolate the upscaled picture. A simple upscaling algorithm which we are using is linearly interpolating the generated pixels. Operating on a grayscale picture, every pixel takes one byte. A 32-bit RISC-V CPU is extended with byte-wise operations which operate on 4x4 pixel matrix at a time, kind of a “kernel”, generating an upscaled 7x7 matrix. This way, instead of operating on each combination of adjacent pixels one by one, it is done in a single instruction

Used area

Used pins

W
Used memory

Project: Micro 8 bit CPU for I/O management + SPI module

Project Name:

Team <full name, affiliation>

Bastian Blochberger, University of Heidelberg (master student)
Biswajit Kumar Sahoo, IIT Bhubaneswar, India (bachelor student)
Bhadra Mayuri, Technische Universität München (PhD)
Nessma Hafez, Technische Universität Chemnitz (PhD candidate)
Yuchao Wang,Technische Universität Chemnitz(master student)

Short abstract / idea
We are implementing a simple SPI module that is managed by a very small custom 8 bit RISC CPU to communicate with SD-Cards, Sensors, … . The main goal is to provide a “management” system that handles memory transfers between the chip and the outer world and can also provide on-chip management, basically replacing the Caravels RISC V with a much smaller system.

Used area
(not run through OpenLane yet)

Used pins
4 (for SPI: CS, MOSI, MISO, SCLK)

Used memory:
512 Byte ROM (Bootloader to load the initial SD card sector with further programs). Can be probably reduced to 256 Byte

1024 Byte RAM (Actual runtime code and loaded/to be written data). This memory can be easily extended by just adding more RAM-Blocks with an address mask. If there is still unused/unoccupied space then it should be considered to extend it.

Project: AES

Team <full name, affiliation>
Zheyu Liu, University of Manchester
Czea Sie Chuah, Technical University of Munich

Short abstract / idea
Using a separate AES accelerator and BRISKI RISC-V core to perform multi-threading AES encryption-decryption. Implement AES & DMA control engine interface in the memory map I/O, connect stream data port of the DMA engine into both AES and BRAM.

Used area

Used pins
~40 (clk, rst, 5: JTAG, 32: data I/O)

Used memory
5k (2k instruction mem, 2k data ram, 1k dma buffer)

Project: Tiny Tsetlin Machine

Team <full name, affiliation>
CONGYI XU, Newcastle University
Hugh Squires-Parkin, Newcastle University
Yefan Zhen, Newcastle University

Short abstract / idea
The tsetlin machine (TM) is a form of learning automaton collective for learning patterns using propositional logic. This project implements a tiny TM for Iris dataset classification with an accuracy ~95% (UCI ML Repository). This tiny ML module is composed of several Tsetlin Automata (TA) blocks, a voting block and an argmax block. The purpose of this project is to demonstrate the potential of TM-based machine learning accelerators at the edge.

Used area

Used pins
~19 (data_in, rst, result)

Used memory
No memory needed

Project Name

Team <full name, affiliation>

Short abstract / idea

Used area

Used pins

Used memory

Project Name

Team <full name, affiliation>

Short abstract / idea

Used area

Used pins

Used memory

Project Name

Team <full name, affiliation>

Short abstract / idea

Used area

Used pins

Used memory