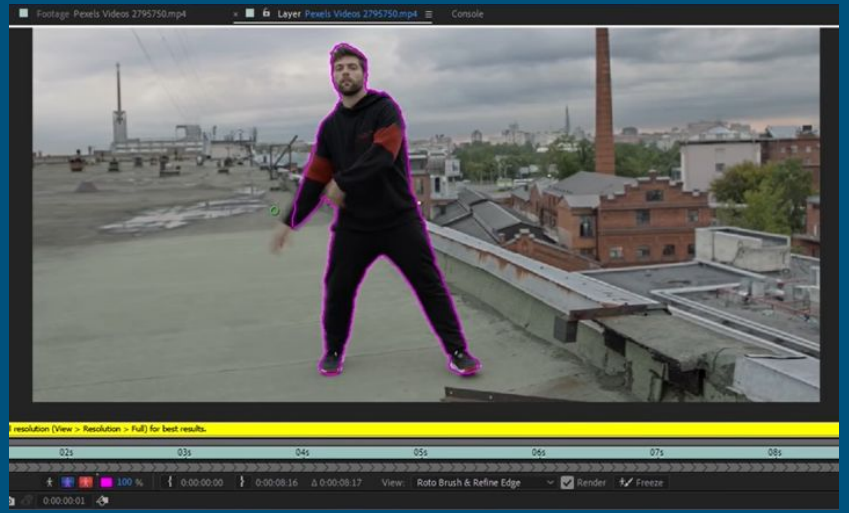
RotoCam

Team B2: Ian Brito, Nataniel Arocho-Nieves, Ian Falcon

Use Case

- Rotoscoping is a tedious process in which VFX artists painstakingly mask out layers of objects, often frame by frame.
- With the RotoCam attachment, filmmakers are able to record directly rotoscoped footage from their camera.
- They can import it to video editing platforms without having to manually rotoscope the footage.
- RotoCam aims to save filmmakers time, money, and allow for an alternative to green screens
- Areas: Software, Hardware, Signals & Systems



Use Case Requirements

Cost	Less than $200	
Battery Life	12h	Allows for a full day of shooting
Weight	Less than 3 lbs.	Reference camera (Panasonic G7) is 0.9 lbs
Dimensions	Less than 8 x 6 x 6 inches	Reference camera is 4.92 x 3.03 x 3.39 inches
Subject Distance	At least 5 feet away from camera	Allows for full body mapping

Use Case Requirements

Accuracy	100% of subject to be in rotoscoped footage, 97% background removed
Quality	1920x1080 Resolution
Framerate	24 FPS
Time to Rotoscope	10 seconds to allow for real-time feedback

Technical Challenges

- Maintaining portability for the hardware
  - Fitting both FPGA and battery into a device that will not restrict camera movement
- Having battery supply power for 12 hours
  - Want to ensure that we can power device for a full day of film shooting

Technical Challenges

- Maintaining desired footage resolution and framerate (1080p, 24FPS)
  - Do not want to compromise video quality while processing
  - 24FPS is a standard in filmmaking we want to maintain
- KEY: We want 100% of the subject to be rotoscoped
  - If the background appears in the rotoscoped footage, it can be manually rotoscoped out, but if the subject is accidentally rotoscoped out, it cannot be edited back in
- KEY: We want at least 97% of background to be removed
  - 97% gives room to make sure the subject is not accidentally edited out

Solution Approach

- Output camera footage via mini-HDMI to FPGA
- Process video on FPGA (i.e. rotoscope)
- FPGA will be reprogrammed to video processing hardware written in Verilog
- Output video to laptop from FPGA

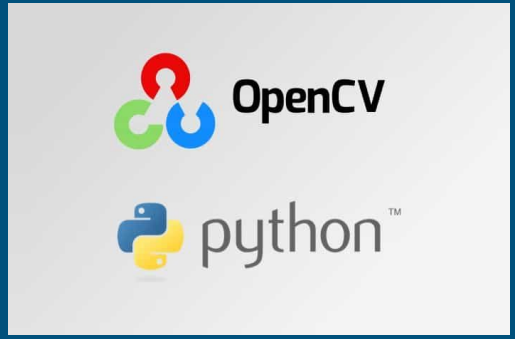
```
module FSM( input logic reset,
            input logic X,
           output logic Y );

 always_ff @(posedge clk)
     if(reset) currentState <= A;
     else      currentState <= nextState;

 always_comb
     case(currentState)
         A:  if(X) nextState = C;
             else  nextState = B;
```

Solution Approach

- Use Python OpenCV to help with rotoscoping footage
- Display footage on laptop, where rotoscoped footage is stored and ready to be used in video editing software such as After Effects



Testing, Verification, and Metrics

- Test battery life while using device to ensure it can last 12 hours of shooting
- Weigh and measure device to ensure portability for the user

Testing, Verification, and Metrics

- Ensure footage is at 1080p and 24 FPS
  - Will meet use-case requirements if footage is at desired quality for filmmakers
- Ensure rotoscoped footage is displayed within 10 seconds of filming
  - Allows users to get feedback on their shots and allow for adjustments
- Ensure the subject is completely in the rotoscoped footage, while most (>= 97%) of background is edited out
  - Test this by manually inspecting the raw footage and comparing it to the rotoscoped footage
  - Will meet use-case requirements if users have to do no (or minimal) manual rotoscoping while not having to reshoot scenes due to having the subject cut out

Tasks and Division of Labor

- Signal Processing – Ian F.
  - Pre processing the image
  - Determine power consumption
  - Implement algorithm to detect background and remove it
- Hardware – Nat
  - Convert software algorithm to digital logic design
  - Implement and verify design on FPGA
  - Implement communication protocols between FPGA and CPU
- Software – Ian B.
  - Python, OpenCV to help process image
  - Store and record footage on laptop

Schedule

RotoCam														
Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
	16-Jan	23-Jan	30-Jan	6-Feb	13-Feb	20-Feb	27-Feb	6-Mar	13-Mar	20-Mar	27-Mar	3-Apr	10-Apr	17-Apr
Setting up visual/recording interface														
Obtain necessary equipment														
Setup FPGA developing environment														
Setup OpenCV and Software environments														
Design software algortihm for background detection														
Optimize algorithm														
Programming FPGA and CPU FPGA connection														
Implementation of FPGA														
Determine power consumption														
Timed testing														
Integration of FPGA														
UI interface														
Storage and file type conversion														
Integration														
Slack														