S.P.U.D.

Single-Core Processor Unit and Display

Meet The Team

Mohamed Moussa
Computer Engineering
Hardware Design

Frank Laterza
Computer Engineering
Project Lead

Gabriel Saborido
Computer Engineering
Electrical Design

Valentina Terry
Computer Engineering
SoC RTL Design

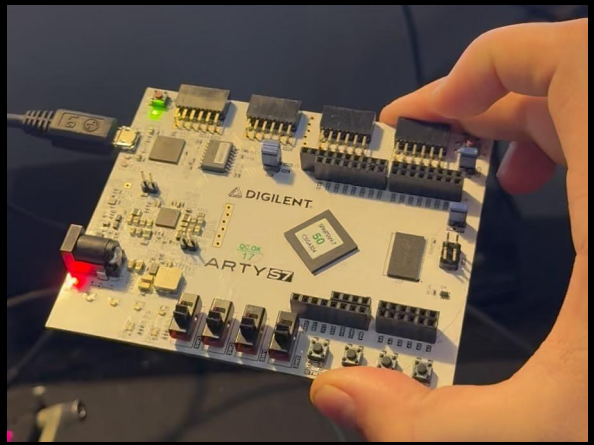
What Is Project S.P.U.D.?

A custom RISC-V CPU implemented onto an FPGA with an LED display for visual feedback

- Modern processors are fully optimized and difficult to inspect
- SPUD proposes a custom, modifiable, demonstrable, and accessible CPU for educational and entertainment purposes
- C programs can run directly on the FPGA-based CPU
- CPU and System on Chip (SoC) are implemented in Verilog
- A custom display core drives the 64x64 RGB Matrix
- The arcade controller will be the main interface to control our interactable demos

Motivation

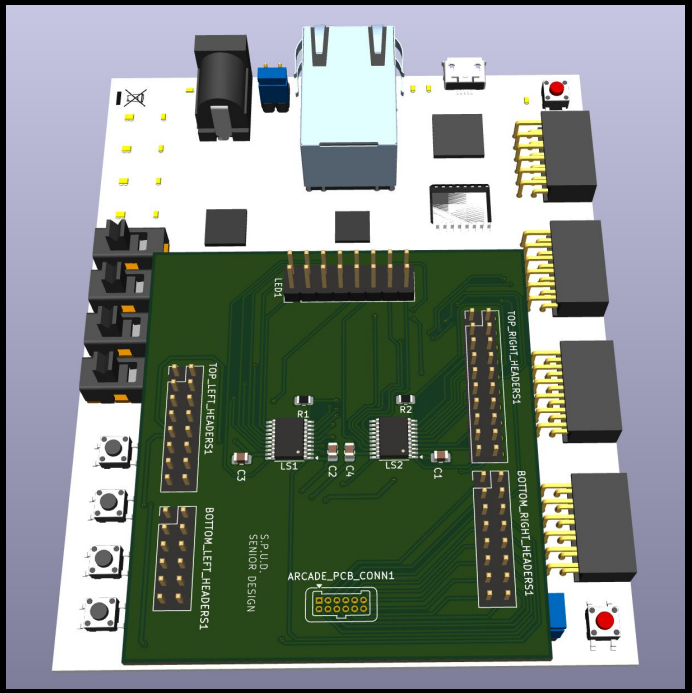
- Take advantage of the fact that our team is **100% CpE**
- Understand the fundamentals of computer architecture, register transfer level (RTL) design, and programming languages
- Build a project that challenges us in a new and creative space
- Develop specific skills that prepare us for our desired careers
- Earn the title of Computer Engineer and have fun!



RISC-V®

Goals and Objectives

- Design our own embedded processor catered to our project
- Build a C library for controlling peripherals like UART, DISPLAY, GPIO, etc
- Implement custom registers in SOC allowing hardware control via instructions
- Configure RV32IM toolchain to generate instructions from C code to load into RAM
- Have each member design a unique interactive demo program!



SPUD PCB connected to the Arty FPGA development board

Engineering Specifications

Specifications	
CPU Frequency	100 MHz
LED Matrix Refresh Rate	50 Hz
Processor Performance	10 MIPS
Functions	arithmetic, jump, and memory
FPGA Board Power Consumption MAX	36 W (12V, 3A supply)
LED Matrix Power Consumption MAX	20 W (5V, 4A supply)
Operational Conditions	Room Temperature: 68–77°F Humidity Level: 40-60%
Price Range	$300-$500

post cdr:

- dedicate entire slide to 3d model before hardware block diagram and after engineering specs, move block diagram to before hardware comparisons (HoQ)
- separate goals and objectives
- range of values for engineering specs
- add latency
- demo plan for specs
- hardware block diagram; add fpga, power supply
- hardware comparison: add single line to explain final reasoning for selection
- add power supply, goes last, before that add a table for all components needing power (power consumption of system)
- move software comparisons to after the entire hardware section
- label PCB parts
- talk about communication protocols, what they are and why we are using it
- replace ‘budget’ with ‘BOM’ or Bill of Materials
- how to improve showcase chances:
    - ask dr mike to be a judge
    - usually showcase picks most useful projects so idk

Hardware Comparisons

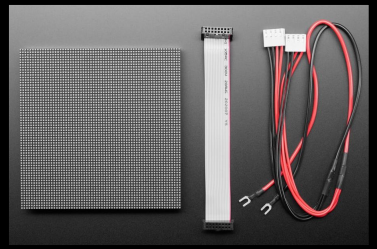
Hardware Comparisons - *System Display*

Feature	Custom LED Matrix (Merkury Innovations)	Adafruit 64x64 RGB LED Matrix	Waveshare RGB Matrix P2-64x64
Resolution	16x16	64x64	64x64
Brightness	Moderate (suitable for indoor use)	High (1,000 cd/m^2)	High (1,000 cd/m^2)
Physical Dimensions	~6 inches	7.13 inches	7.13 inches
Interface	Bluetooth app control	HUB75 interface	HUB75 interface
Power Requirement	USB-powered	4A 5V power supply	5V power supply
Best For	Beginners or simple decorative projects	Advanced projects requiring high resolution and brightness	Flexible and affordable projects needing high resolution

Part Selected - Adafruit Matrix

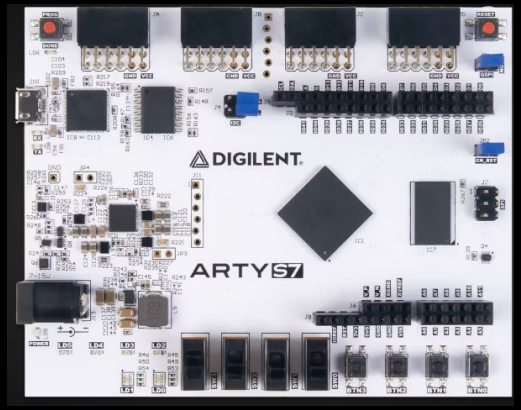
Feature	LCD (Liquid Crystal Display)	OLED (Organic Light-Emitting Diode)	Matrix Displays
Resolution Potential	High, dependent on pixel size and TFT technology	High, with fine control over individual pixels	Limited by matrix grid design
Brightness	Moderate to high (200-400 cd/m^2 with LED backlight)	Lower peak brightness compared to LCD; true blacks enhance perceived contrast	Limited brightness (~1/10 of segmented displays)
Contrast Ratio	Moderate (limited by backlight)	Infinite (self-emissive pixels can turn off completely for true blacks)	Moderate
Viewing Angles	Narrower than OLED (varies by LCD type: TN, IPS, VA)	Wide 180 degrees	Moderate
Response Time	Slower (~6-16 ms for LCDs; faster for TN panels)	Extremely fast (<0.01 ms), ideal for motion clarity	Moderate
Power Efficiency	Requires backlight, consumes more power	Highly efficient; only active pixels consume power	Dependent on the driving method

Technology Selected - Matrix Display



Adafruit 64x64 LED Matrix

Hardware Comparisons - *FPGA Selection*

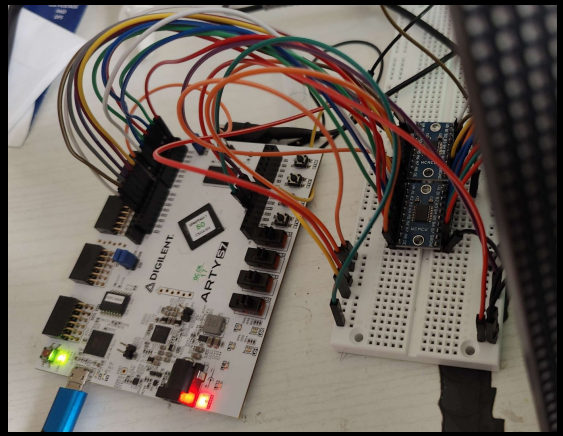


Xilinx Arty S7-50:
Spartan-7

Feature	Xilinx Arty S7-50: Spartan-7	Xilinx Arty A7-100T: Artix-7	Intel Cyclone V GX Starter Kit
Logic Cells	52,160	101,440	77,000
Flip-flops	65,200	126,800	56,832
Block RAM	2,700 Kbits	1,800 Kbits	1,532 Kbits
Clock Frequency	100 MHz	100 MHz	programmable (10 - 810 MHz)
Communication Protocols	- USB-UART - USB-JTAG - Quad-SPI Flash - Pmod ports - Arduino/chipKIT shield connector	- USB-UART - USB-JTAG - Quad-SPI Flash - Pmod ports - Arduino/chipKIT shield connector	- PCIe Gen2 - USB 2.0 - UART - I^2C - SPI - LVDS
Memory Interface Support	- 256 MB DDR3L with a 16-bit bus @325 MHz (650 MT/s) - 128 Mbits Quad-SPI Flash	- 256 MB DDR3L with a 16-bit bus @667 MHz - 128 Mbits Quad-SPI Flash	Supports DDR3, DDR2, LPDDR2
Development Tools	Vivado Design Suite, WebPACK Edition	Vivado Design Suite, WebPACK Edition	Quartus Prime, Lite Edition
Board Price	$150-$200	$299.00	$270.00
Applications	Embedded systems, education, prototyping, hobbies	High-performance systems, communications	Industrial, automotive, telecommunications

Part Selected - Arty S7-50 Spartan-7

Hardware Comparisons - *Level-Shifter*



TXS0108E Level Shifter

Feature	Resistor	Op Amp	Transistor
3.3V to 5V conversion	N/A Only 5V to 3.3V	Yes	Yes
Propagation delay	~50-200ns	~10-100ns	~5-50ns
Directionality	Unidirectional/ Bidirectional	Unidirectional/ Bidirectional	Unidirectional/ Bidirectional
Durability	Low	Medium	High

Technology Selected - Transistor

Feature	TXS0108E	TXB0108	PCA9306	74LVC245
Propagation delay	1-4 ns	<1 ns	10-20 ns	6-8 ns
Directionality	Bidirectional	Bidirectional	Bidirectional	Unidirectional
Voltage Range	1.2V to 5V	1.2V to 5V	1.2V to 5V	1.65V to 5V
Channels	8	8	2	8

Part Selected - TXS0108E

Hardware Comparisons - *Flash Memory*

Feature	M3004316	AT25DL081	24LC16B (Arty S7)
Size	4Mb	8Mb	16 MB (128 Mbits)
Mem Technology	MRAM	FLASH	FLASH
Write/Read Protocol	Parallel	SPI	Quad-SPI
Program Technique	Direct write (no erase needed)	Page programming with sector erase	Page programming with sector erase (similar to standard SPI)
Access Speed	35ns read/write	~100MHz clock, sequential access	Up to 50 MHz data rate for Quad-SPI programming
Endurance	Virtually unlimited	100,000 cycles (typical)	Typically 100,000 cycles (similar to standard SPI)
Data Retention	Long-term persistent	>10 years (typical)	>10 years (typical)
Voltage Range	2.7V-3.6V	1.65V-3.6V	1.65V-3.6V (for S25FL127S/S25FL128S)
Temperature Range	-40°C to +105°C	-40°C to +85°C	-40°C to +85°C

Part Selected - 24LC16B

Software Comparisons

Software Comparisons - *Instruction Set Architecture*

Feature	RISC-V	ARM	X86
Architecture Type	RISC	RISC	CISC
Instruction Length	Fixed 32-bit (base), supports 16-bit compressed	Fixed 32-bit (ARM), 16-bit (Thumb)	Variable length (1-15 bytes)
Extensibility	Highly extensible	Limited extensibility	Limited extensibility
Instruction Encoding	Simple, fixed format	Fixed format	Complex encoding
Address Modes	Simple	Multiple addressing modes	Complex addressing modes
Open Source	Yes	No	No
Market Adoption	Growing	Widespread	Dominant in desktop/server

Technology Selected - RISC-V

Feature	RV32IM	RV64F	RVDFD
Instruction Set Type	Base integer instructions	Single-precision floating-point	Double-precision floating-point
Registers	32 integer registers	32 integer + 32 floating-point	32 integer + 32 floating-point
Advantages	Simple and compact for embedded systems	Enhanced arithmetic precision for scientific applications	High precision and fused operations for advanced computing

ISA Selected - RV32IM

Software Comparisons - *Source Code*

Selection	RV32IM Core	RV32I (Single & Multi-Cycle)	UltraEmbedded RV32IM Core
Architecture	Five-stage pipeline with forwarding and hazard detection	Single-cycle or multi-cycle implementations	Two/three-stage pipeline (configurable) with optional caches, MMU, and AXI bus support
Complexity	More complex than SPUD's needs, but includes useful optimizations	Well-balanced; single-cycle design with an option for multi-cycle	More complex than the other cores, but configurable for FPGA SoCs
FPGA Compatibility	Not tested on hardware, simulation only (ModelSim)	Designed for FPGAs	Verified on similar (but not the same) FPGA
Testing & Verification	Functional simulation performed, but no FPGA testing	Well-organized and has all the necessary testbenches	Includes extensive testbenches and a working SoC reference design (booting Linux!)
Documentation	Limited, originally a school assignment	Clear structure with licensing information (MIT License)	Well-documented repository
Extensions	RV32IM (includes multiplication and division)	RV32I only	RV32IM (M-extension for multiply/divide), optional caches, MMU, and peripheral support

Source Code Selected - UltraEmbedded RV32IM

GitHub

Software Comparisons - *Toolchain Selection*

Selection	Vivado	RISC-V GNU Toolchain	OpenOCD
Primary Purpose	FPGA design and simulation	Software development for RISC-V	Debugging and flashing
Features	- FPGA synthesis and implementation - Hardware programming - Simulation support	- GCC compiler for RISC-V - Binutils for assembling and linking - GDB for debugging - Cross-platform support	- On-chip debugging - FPGA programming support - Software loading onto the processor
Role in SPUD	Implements and verifies the CPU on the FPGA, manages design synthesis and simulation	Compiles high-level software (including games) into executable machine code for SPUD's CPU	Facilitates debugging and flashing the CPU design onto the FPGA, ensuring software and hardware interaction works correctly
Reason for Selection	Needed for FPGA implementation and simulation	Open-source and widely supported; provides all the necessary tools for RISC-V software development	Enables real-time debugging but at the cost of significant extra time and effort

Toolchain Selected - Vivado & RISC-V

Software Comparisons - *PCB Design Software*

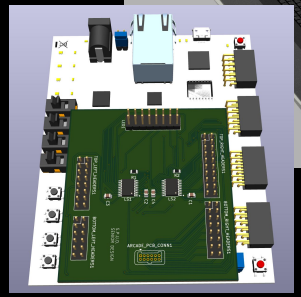
Feature	Fusion360	KiCad	Altium Designer
Cost	Free for students	Free	Subscription based
Ease of Use	Easy	Moderate	Intuitive
Routing Tools	Basic	Manual	Advanced & constraint-driven
Component Libraries	Good for common parts	Open-source	Large selection of verified parts
3D Visualization	Basic 3D support	Basic 3D support	Advanced 3D visualization
Version Control	Cloud based (Fusion360)	Git	Altium 365 (Built-in)

Technology Selected - KiCad

Hardware Design

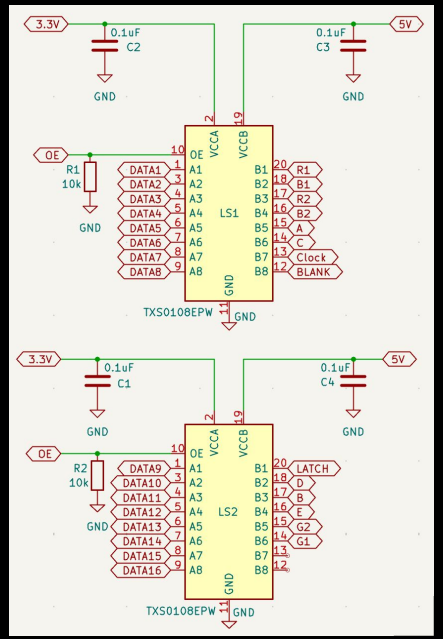
Hardware Design

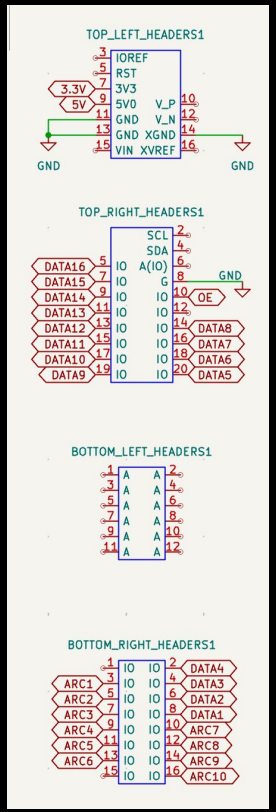
- SPUD uses a custom PCB hat that fits on top of the FPGA.
- The hat allows us to add connectors to make secure connections to the matrix and arcade controller
- Level shifter are also implemented onto the hat to allows signals be control at safe voltage levels
- Button inputs from the arcade controller PCB are sent through the GPIO pins, allowing for real-time interaction with SPUD

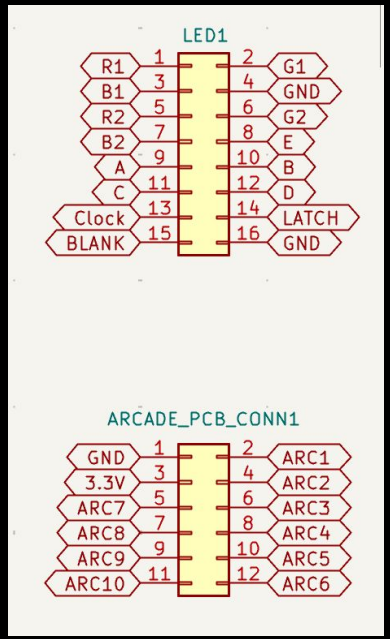


PCB Design

Peripheral Interface PCB - *Schematic*



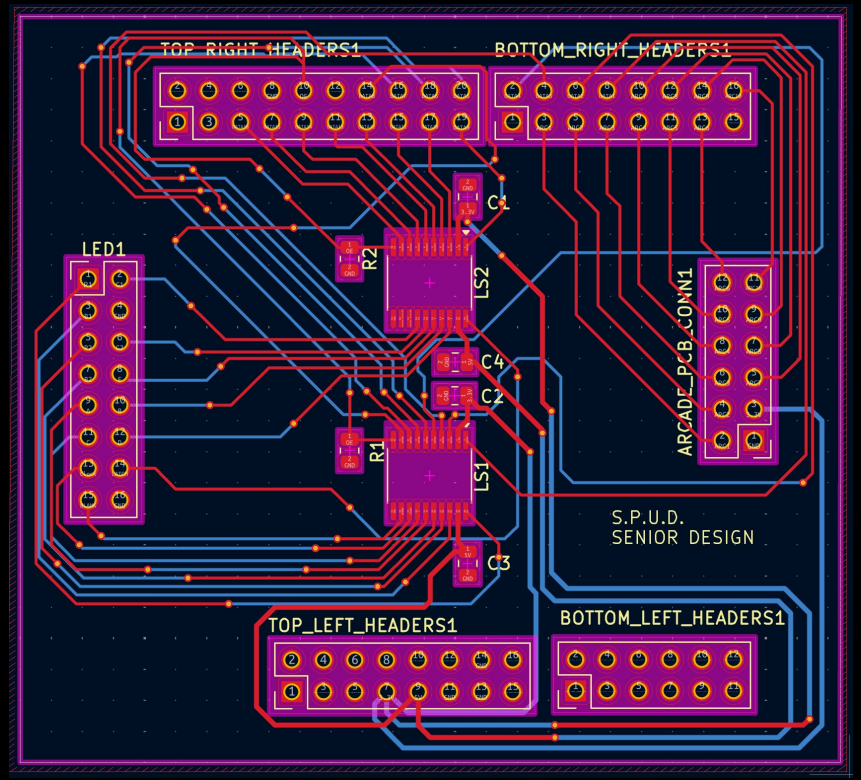


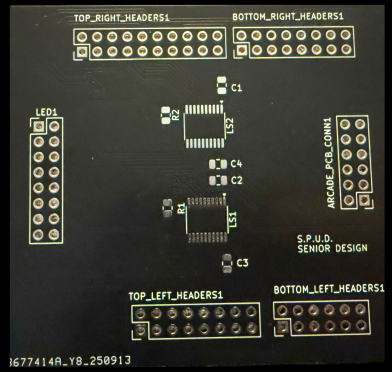


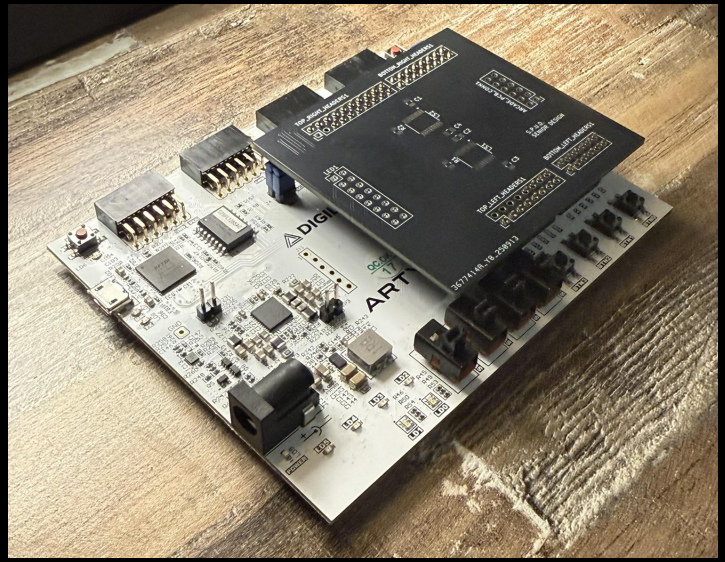
Houses FPGA, two 8-bit bidirectional level shifters, LED matrix & arcade controller connecters.

TXS0108E level shifters are for 3.3V-5V conversion for LED matrix.

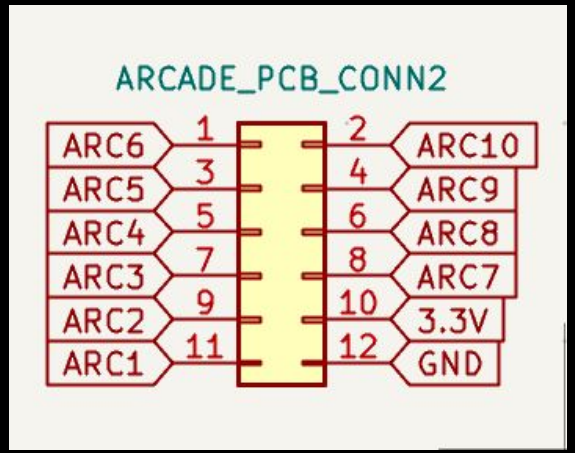
Peripheral Interface PCB - *Layout*

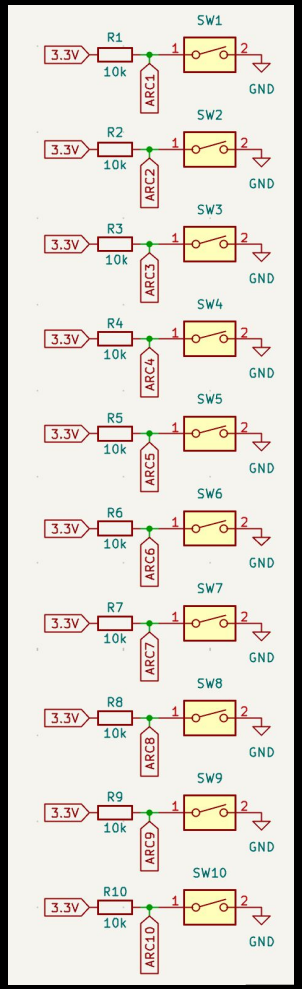






Arcade Controller PCB - *Schematic*



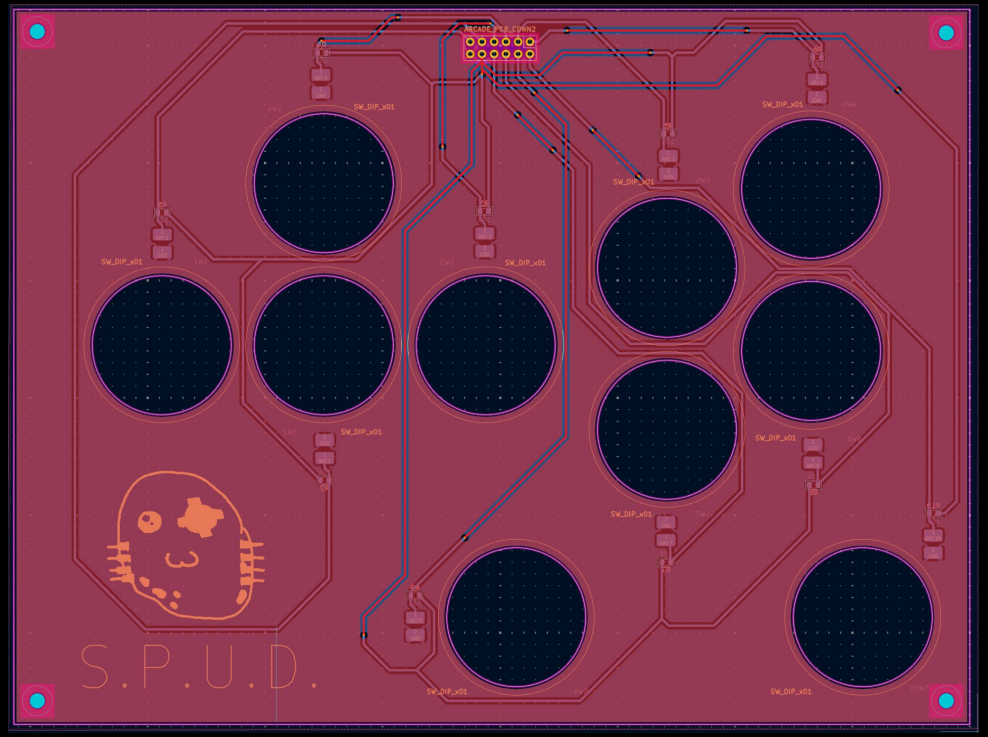


Powered by FPGA at 3.3V.

10 GPIO buttons for controlling S.P.U.D. in demo.

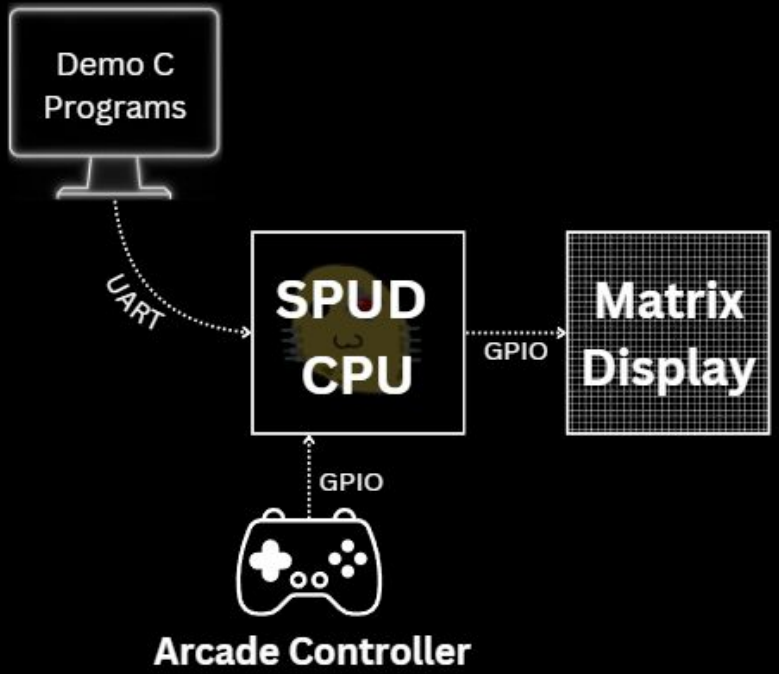
Connects directly to peripheral interface board.

Arcade Controller PCB - *Layout*

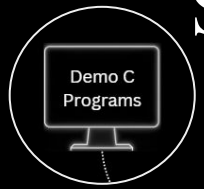


Software Design

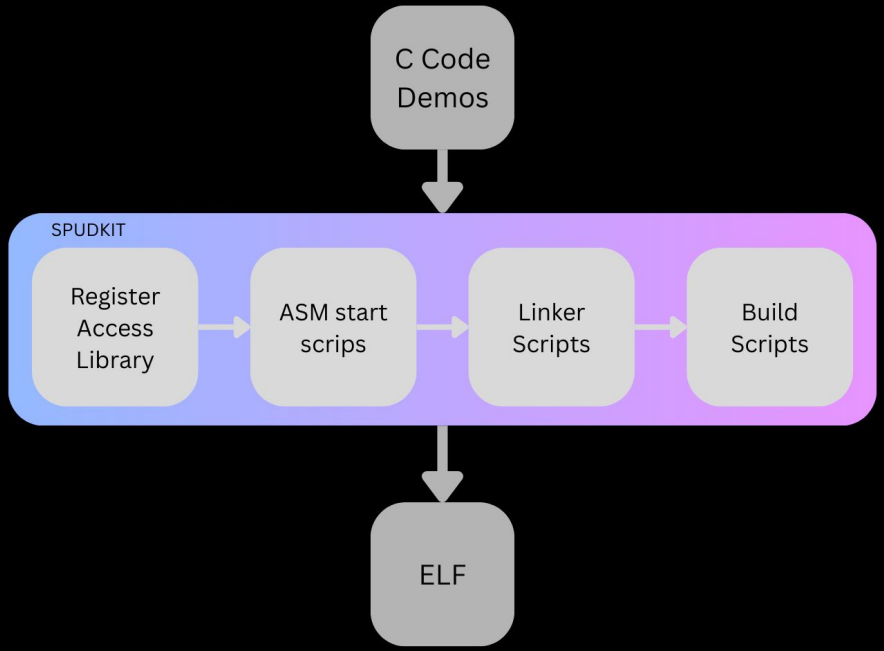
Software Design - *Overview*



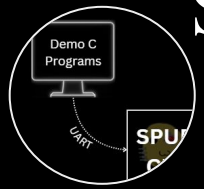
Software Design - *Development*



- Spudkit is a bundle of tools and software that build code designed to be ran on SPUD
- A C library is created to provide easy development to access SPUD's hardware
- Linkers scripts are written for our build scripts memory addressed like the main entry point address stack pointer locations
- RISC-V Cross compiler toolchains generate the ELF file to be loaded onto SPUD

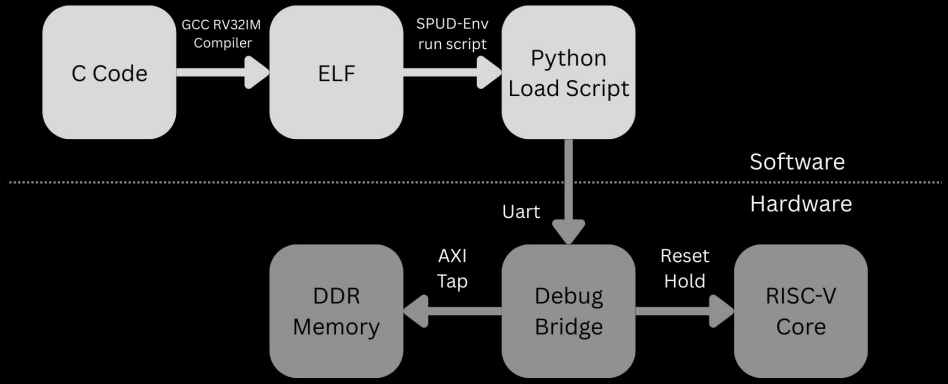


Spudkit flow diagram



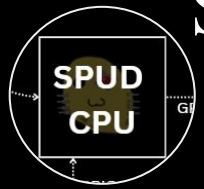
Software Design - *Program Loading*

- Each demo is written in C code and uses the GCC RV32IM compiler
- An .elf file is generated using linker scripts that properly configure code for our processor
- A custom python script is used to command the processor into debug mode
- The debug bridge will hold the CPU into reset and hijack the uart port
- Through Serial/UART we load the machine code into DDR3 RAM via the debug bridge

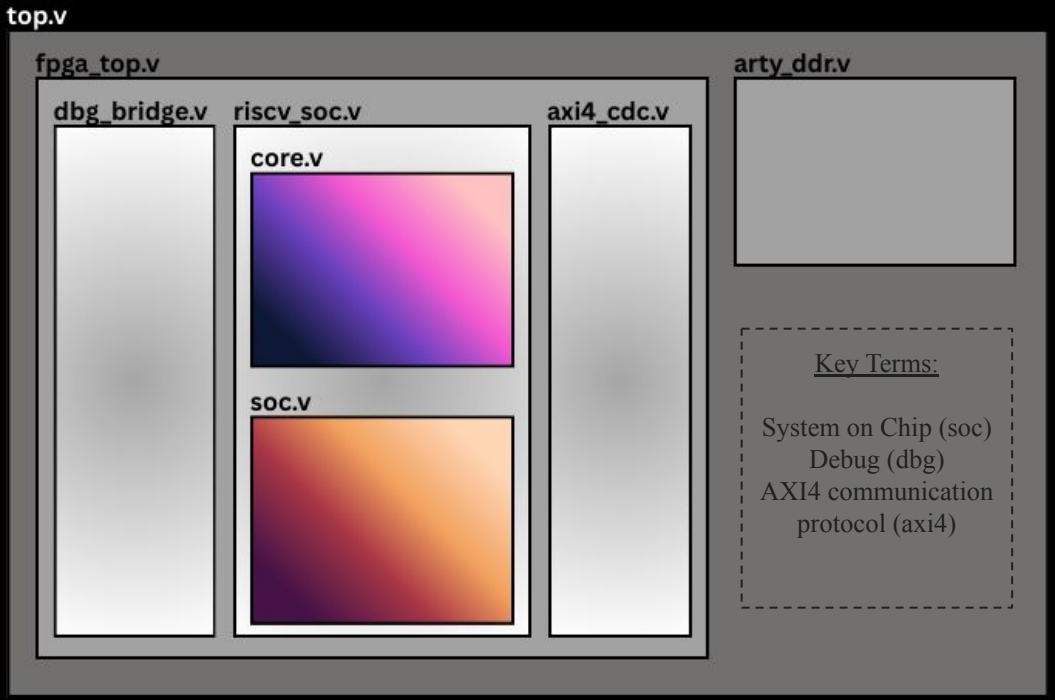


Software upload flow diagram

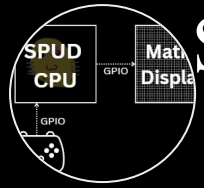
Software Design - *CPU Top Level*



- *arty_ddr.v:* RAM interface for the Arty FPGA
- *dbg_bridge.v:* used for debugging and accessing internal signals
- *riscv_soc.v:* houses the core from UltraEmbedded, and the modified SPUD soc
- *axi4_cdc.v:* handles *clock domain crossing* in the AXI



hierarchical, modular representation of SPUD's source code (Verilog)

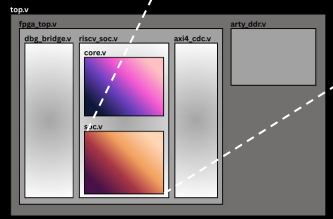


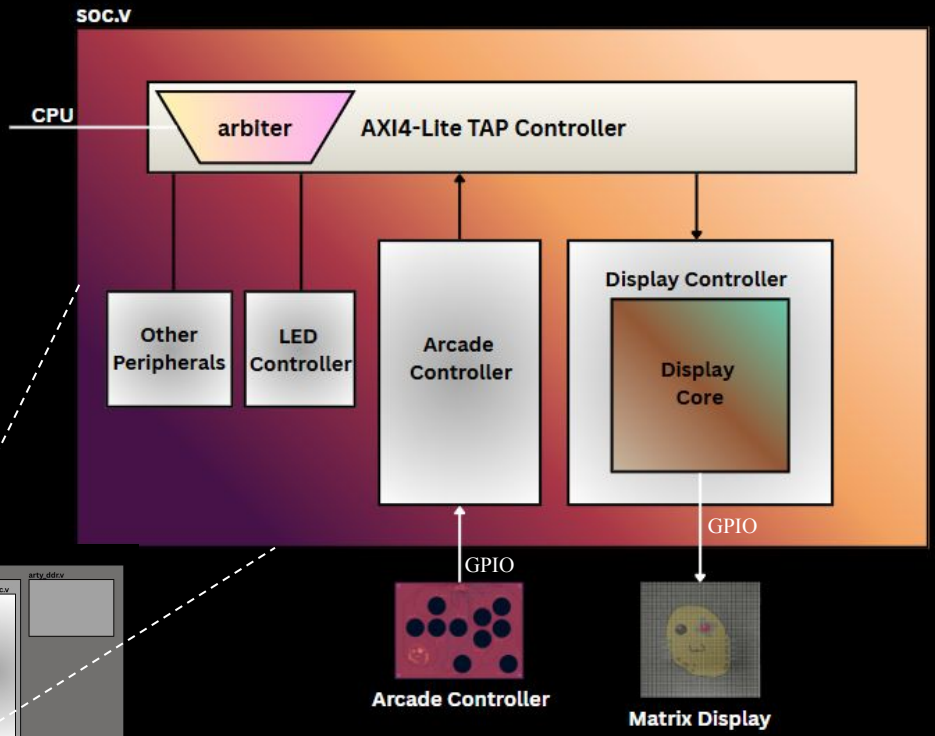
Software Design - *SoC RTL Design*

Table of Register-Mapped Peripherals

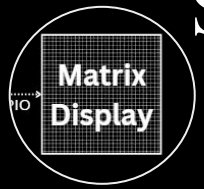
Peripheral	Base Address	Info
Other Peripherals	0x90000000	Interrupts, Timer…
UART	0x92000000	Serial Communication
GPIO	0x94000000	General I/O pins
LED Controller	0x95000000	Controls FPGA LEDs
Arcade Controller	0x96000000	Reads inputs from custom Arcade Controller PCB
Display Controller	0x97000000	Provides CPU register access to update pixel data on display

highlighted peripherals are designed specifically for SPUD



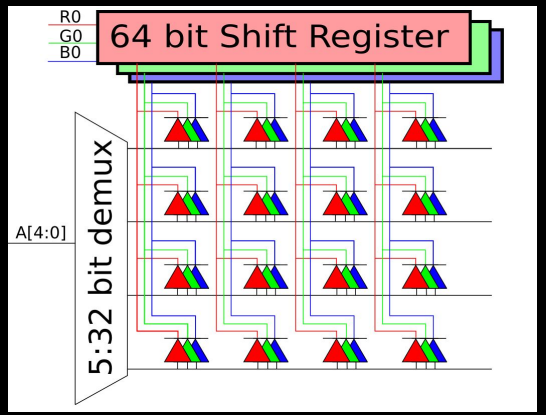


Software Design: *LED Matrix*

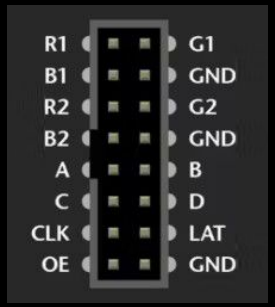


The 64x64 LED matrix utilizes an interface called HUB75

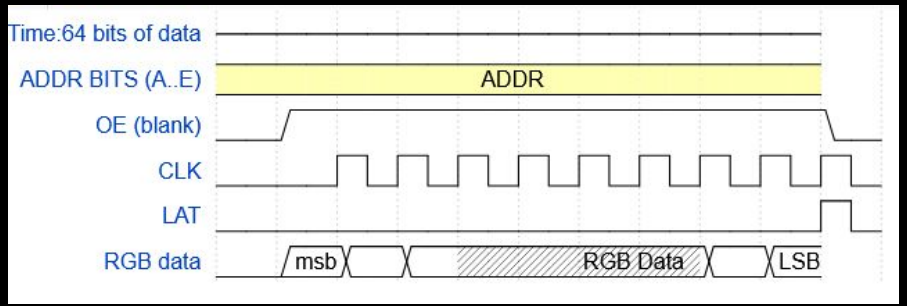
- 6 RGB pins, 2 for each color (R1,R2,G1,G2,B1,B2)
- 5 addressing pins, used to choose which row to select. Addresses one row per panel.
- Clock progresses shift register.
- Latch and OE for display and output control.
- Connects to the SOC video display interface



From Ross Schlaikjer- RHYE.org



(HUB75 Pinout)



HUB75 Waveform

Prototyping and Testing

Testing

Hardware:

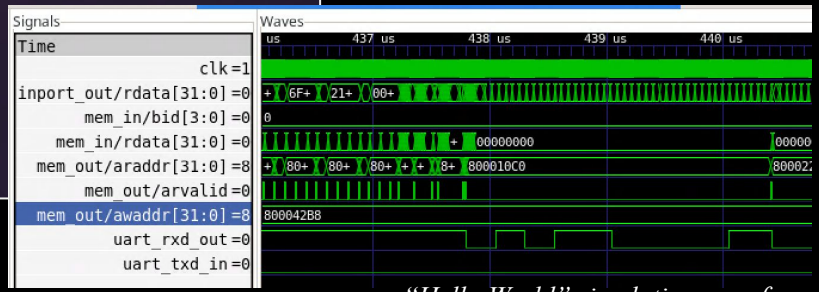
- FPGA Synthesis, Place & Route, Bitstream Generation
- Button input testing for the Arcade Controller
- Visual tests of the matrix display and LED Controller
- Resource utilization check on the FPGA

Software:

- CPU simulation using Verilator, an open-sourced simulator for Verilog
- Running compiled C test programs (booting Linux, printing “Hello World”, etc)
- Integration testing of software with custom peripherals (Arcade + Display Controller)

```
./build/test.x -f ../core/isa_sim/images/basic.elf

        SystemC 2.3.3-Accellera --- Sep 15 2025 18:45:01
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED

Info: (I702) default timescale unit used for tracing: 1 ns (sysc_wave.vcd)
Memory: 0x2000 - 0x3cd3 (Size=7KB) [.text]
Memory: 0x3cd4 - 0x3ce7 (Size=0KB) [.data]
Memory: 0x3ce8 - 0x4d07 (Size=4KB) [.bss]
Starting from 0x00002000

Test:
1. Initialised data
2. Multiply
3. Divide
4. Shift left
5. Shift right
6. Shift right arithmetic
7. Signed comparision
8. Word access
9. Byte access
10. Comparision
TB: Aborted at 110 us
```

CPU core simulation testing



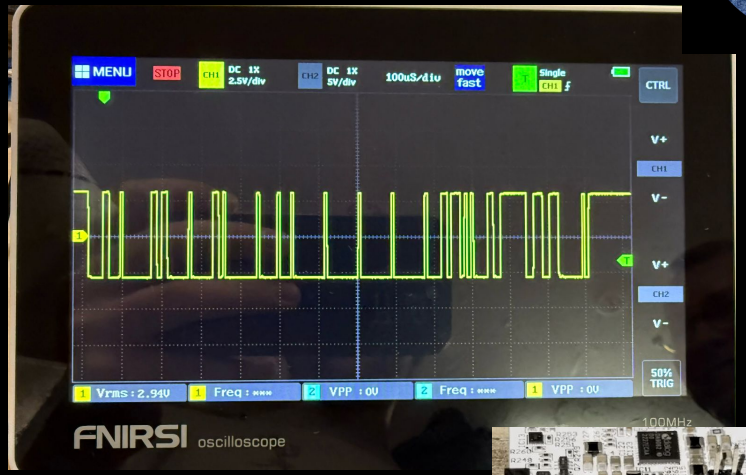
“Hello World” simulation waveform

```
./spud_rv32i-tools/demos/hello_world/build/hello_world.elf -b 0x80000000

        SystemC 2.3.3-Accellera --- Sep 15 2025 18:45:01
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED

Info: (I702) default timescale unit used for tracing: 1 ns (hello_test.vcd)
MEM: Create memory 0x80000000-81ffffff
Memory: 0x80000000 - 0x80001f1b (Size=7KB) [.text]
Memory: 0x80001f1c - 0x800022b7 (Size=0KB) [.rodata]
Memory: 0x800022b8 - 0x800022bb (Size=0KB) [.sdata]
Memory: 0x800022bc - 0x800042db (Size=8KB) [.bss]
Memory: 0x800042dc - 0x800042df (Size=0KB) [.sbss]
Starting from 0x80000000
hello world!!!
```

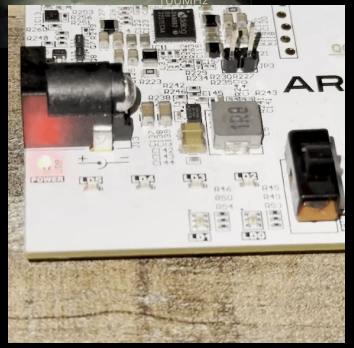
“Hello World” test C program using the spud processor

Testing Hardware

- Using Oscilloscope we were able verify the peripheral hardware logic
- Designed custom verilog to show when CPU is in reset and the debug bridge is enabled
- We were able to successfully verify the implementation and synthesis via vivado
- Successfully loaded machine code into RAM and verified expected behavior with UART outputs



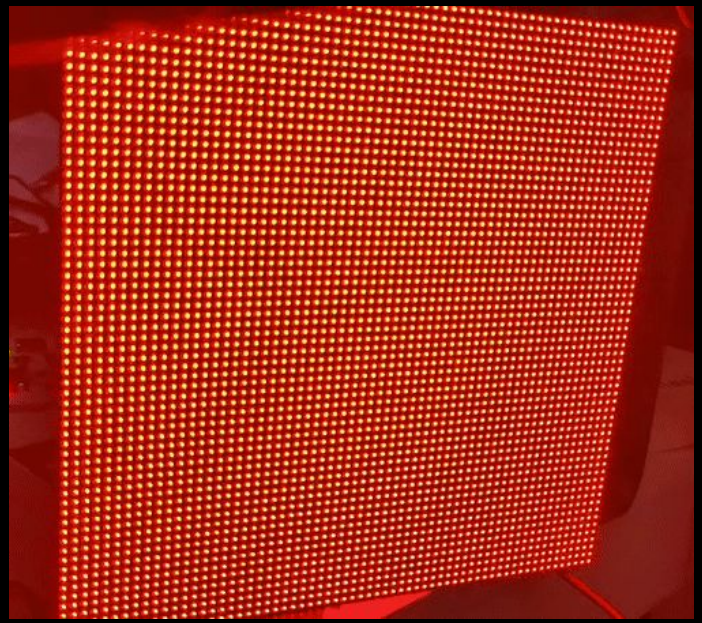
Testing Uart Signals on Oscilloscope



Shows Debug Mode activation

Challenges

- Unstable connections to display caused by cheap bread board and cables
- Verilator (simulator) setup was difficult to configure due to outdated dependencies and links
- The RISC-V design is large and complicated
- Configuring the Memory Interface Generator (M.I.G.) to work with our DDR3 chip
- Vivado setup difficulties
- Creating custom build scripts to generate machine code



Administrative Content

Budget

Type	Component	Quantity	Total
FPGA	Arty S7-50: Spartan-7 FPGA	1	$199.00
PCB	Peripheral Board	1	$2
PCB	Controller Board	1	$15.80
LED Matrix	64x64 LED Display	1	$37.90
PSU	5V 4A power supply	1	$14.95
Power Adapter	Female DC Power adapter	1	$2.00
Level Shifter	HiLetGo 5pcs TXS0108E Conversion Module	1	$9.75
Button	Arcade Button - 30mm Translucent Red	10	$53.60
Connector	Harwin Gecko 12POS 1.25mm	2	$6.15
Total:			$341.15

Work Distribution

Team Member	Responsibilities
Gabriel Saborido *Computer Engineering*	ARTY S7 Expansion Board PCB and Power Management
	User Control Interface PCB
	Soldering and Assemble
Mohamed Moussa *Computer Engineering*	Matrix display interface
	Bit Shift Display Control Logic
	Display Control C library development
Frank Laterza *Computer Engineering*	Team Lead, overseeing all tasks
	System integration and infrastructure setup
	Demo C library development (Spudkit)
Valentina Terry *Computer Engineering*	SoC design customization
	GCC RISC-V toolchain setup
	CPU simulation and testing

Progress & Plan

Demos

Each member on the team will create their own demo to run on SPUD! Below are possible ideas.

- Interactive
  - Chess
  - Tetris
  - Pong
  - Snake
  - Doom
- Simulation
  - Encoded video playback
  - Sorting Algorithms
  - Conway's Game of Life
  - Langton's ant

A 3D spinning donut rendered on SPUD! Built with spudkit and using uart to simulate display output.

Current Progress and Next Steps

Completed	Next Steps
PCB Design and Shipment	Full Implementation of Matrix Core
CPU Implementation on FPGA	Peripheral Customization in SoC (done)
Peripheral Customization in SoC	Create High Level Programs
Basic Display	Testing & Verification
	PCB & System Integration
	System Testing

A rough estimate: 50% project completion

Thank you!

~ The SPUD team 🫠