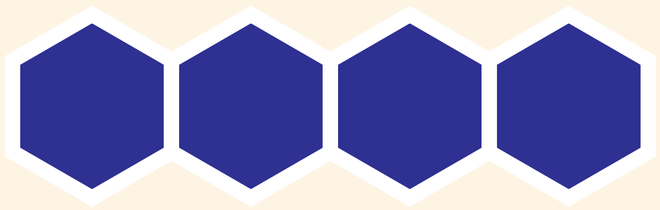
How to play:

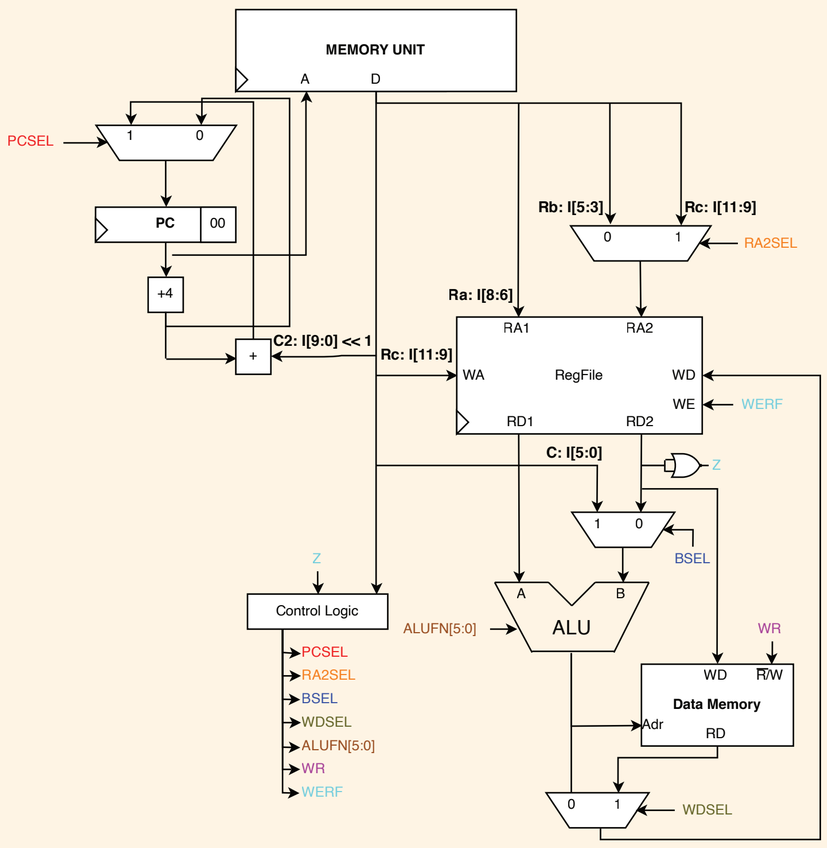
1. Take turns to place pieces of your colour in an empty hex.
2. If you form four in a row, you win.
3. If you form three in a row (without a four in a row), you lose.



you win

you lose

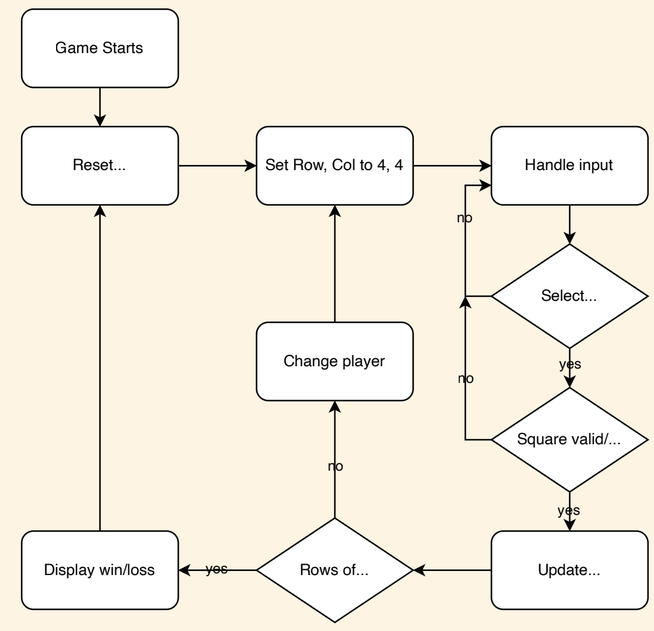
The Pre-Alpha CPU



The schematic for our “pre-alpha” architecture.

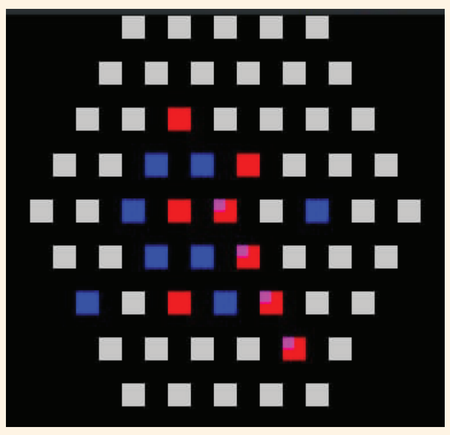
	OPC			OP	
	LD	ST	ADDC/SHRC	ADD/SHL/AND	BF
ALUFN	'+'	'+'	OP	OP	0
WERF	1	0	1	1	0
BSEL	1	1	1	0	0
WDSEL	1	0	0	0	0
MWR	0	1	0	0	0
RA2SEL	0	1	0	0	1
WERF	1	0	0	0	0
PCSEL	0	0	0	0	Z

Control signals for our instruction set.



Simplified flowchart of our game logic.

yavalath



Our Pygame Emulator.

Challenges faced

Architecture Due to the size of our board, there are many possible ways to form a n-in-a-row connection, and checking for all of them is a difficult task. Thus, we developed our own 16-bit architecture for our project.

Testing We would construct an emulator to prototype our restrictive architecture, using only 8 instructions for our program logic.

Output and Input A significant issue faced was outputting at least 122 bits of information for 61 LEDS. Since the FPGA can only output 96 bits, we have to use a matrix of wires to send the correct outputs. Through sending alternating, successive outputs, we create the illusion of having 122 bits of information with only 27 output wires.

Data Representation Since our architecture only allows for 6 bit constants, we decided to use only 64 words of memory for all our data. To store information about our board state, we use 9 two-byte numbers to store each successive row as a bit mask.

Why this game?

Yavalath is a board game invented by the computer program LUDI (by Cameron Browne) in 2007. With its simple rules (described in just three lines) and surprising depth, we felt like it was an apt game to implement for this project.

group 11-1: Cawin chan, Daniel low, Mark huang, Sean gunawan, & sim Jia Ren