One-player Blackjack

A project plan for ECE 383

by

C2C Dananga K. Agalakotuwa

April 22, 2024

1.1 Objective Statement

This will be a single-player game of Blackjack against the computer. The computer will act as the dealer.

1.2 Requirements

Basic functionality

- A player must be able to play a complete game of Blackjack against the computer.
- Use keyboard inputs to implement game inputs.
- The HDMI Monitor must be used to draw the "cards" and simulate gameplay. Card dimension would be 32x32 pixels.
- For this level, the deck of cards can be fixed instead of being random.
- *The UART feedback may be used for debugging and echoing gameplay.*

B-functionality

- Complete Basic functionality.
- A mouse must be used for game control.
- For this level, the deck of cards needs to be random, and cards cannot be repeated (2 Jack of Clubs for example).

A-functionality

- Complete both basic functionality and B-functionality.
- Include audio sound outputs for start of game and end of game.

1.3 Level-0 Description & Top-Level Design

Overall Inputs:

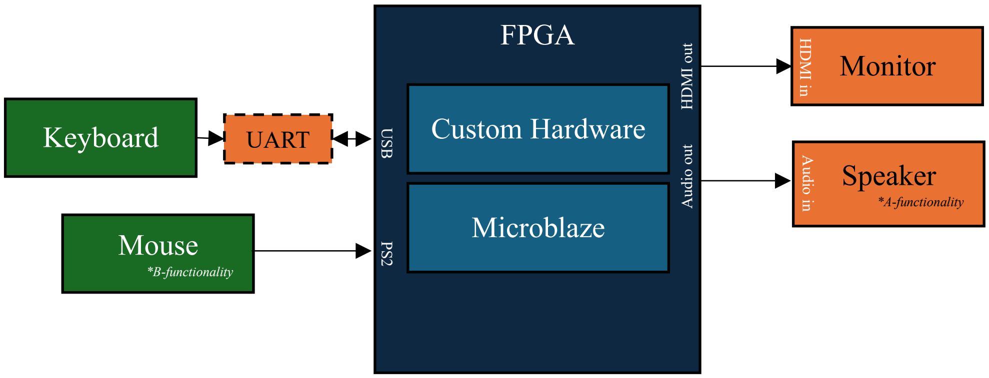
Keyboard via USB
Mouse vis USB
Switch

Overall Outputs:

Display on Monitor
Speaker

Overall Behavior:

A switch on the FPGA would be used to start the game. This switch will act as an interrupt, if turned off game will end regardless of progress. The game will initially be played using the keyboard (“H” key to hit and “S” key to stand). A new input device will be introduced later(mouse) to control game inputs (left mouse button to hit and right mouse button to stand). The display will show cards as they are dealt (dealer’s initial card will not be revealed till end of game). Once the end of game has been reached (dealer is done hitting) the result of the game will be shown on the display (a green screen for player win, red screen for player loss).



2 PLAN

The project plan defines how you are going to go about implementing the design set forth in your proposal. The plan should then go on to include the following sections.

2.1 PROPOSAL

I have corrected the mistakes pointed out to me by Dr York for Section 1.

2.2 DETAILED ARCHITECTURE AND SUB-SYSTEM DESIGN

You need to provide the detailed design of your system. A detailed design should be split into level-1 subsystems, such as Datapath and control.

2.2.1 LEVEL-1 DESIGN

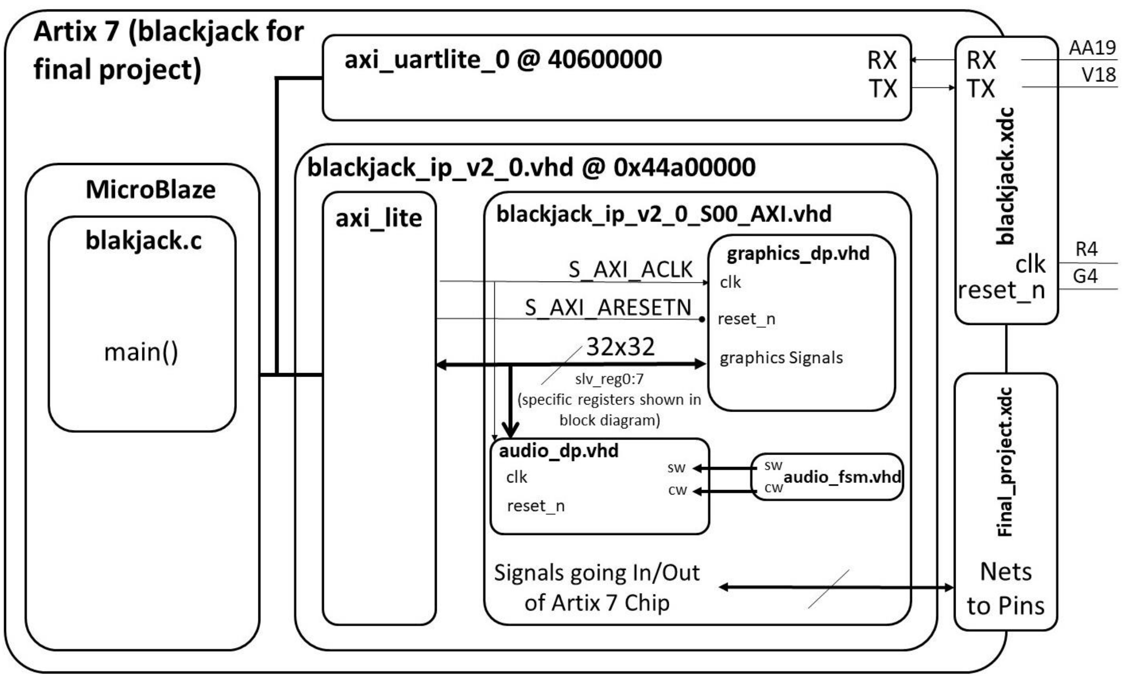


Figure 1: Microblaze Level-1 Design

Figure 1 shows the microblaze design which will be used for my final project. The specific registers written and read using microblaze will be shown below.

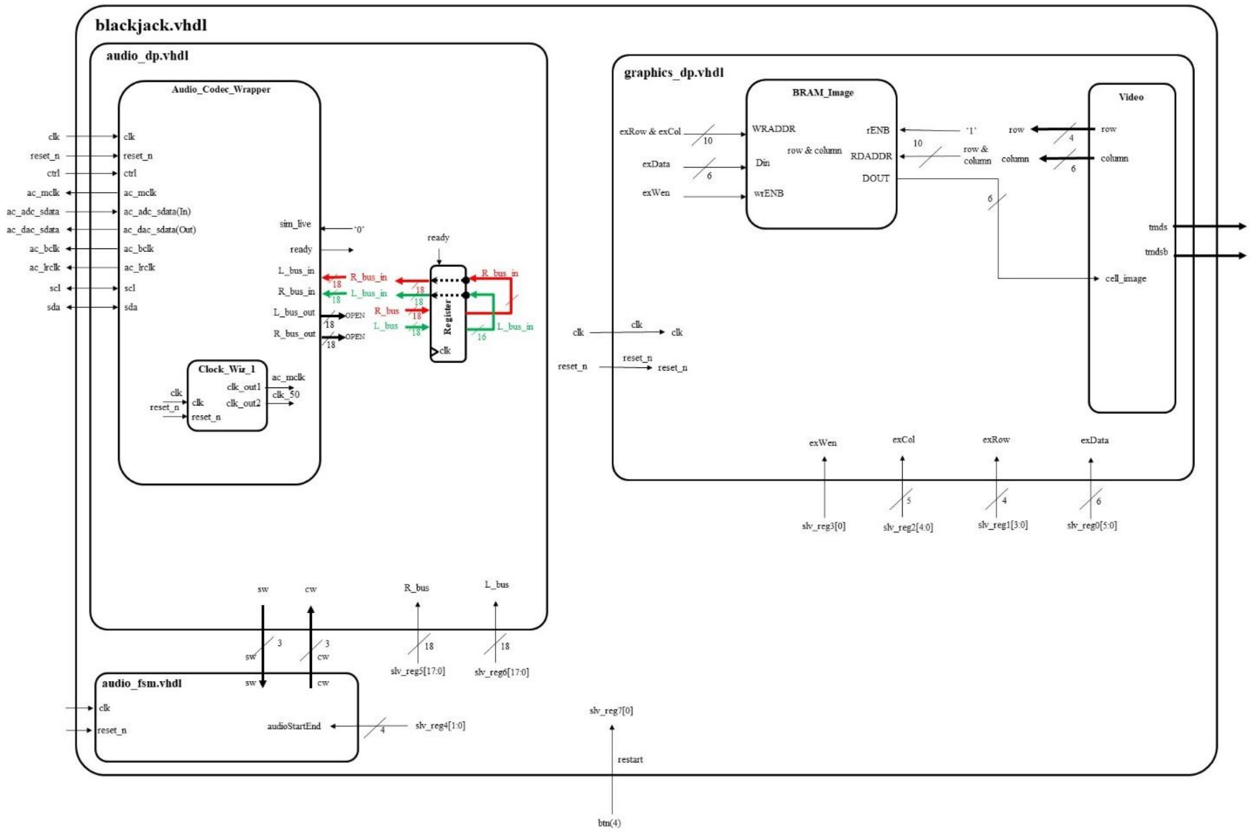


Figure 2: Level-1 Design

Figure 2 shows the main three components that I will be using to achieve the final project. I rely on the microblaze registers to run my graphics datapath, therefore, most of my "game" code will be in C-code.

MicroBlaze Registers [in = read; out = write]		
Signal	**Direction**	**Register/Bits**
exData	read	slv_reg0[5:0]
exRow	read	slv_reg1[3:0]
exCol	read	slv_reg2[4:0]
exWen	read	slv_reg3[0]
audioStartEnd	read	slv_reg4[1:0]
R_bus	read	slv_reg5[17:0]
L_bus	read	slv_reg6[17:0]
restart	write	slv_reg7[0]
h	(internal: keyboard inputs)	
s	(internal: keyboard inputs)	

Outside Artix 7 (lab2.xdc)		
Signal	**Type**	**Package Pin**
clk	clock	R4
reset	button	G4
ac_mclk	Audio Codec	U6
ac_adc_sdata	Audio Codec	T4
ac_dac_sdata	Audio Codec	W6
ac_bclk	Audio Codec	T5
ac_lrclk	Audio Codec	U5
sda	QSPI	V5
scl	QSPI	W5
tmds[3:0]	HDMI out	T1 AB3 AA1 W1
tmdsb[3:0]	HDMI out	U1 AB2 AB1 Y1
restart	button	B22

Figure 3: Datapath signals to Microblaze registers

Grid Description	Index	Binary Value	BRAM Hex Value
Black Box	0	000000	0000
White Box	1	000001	0001
Dealer Box	2	000010	0002
Player Box	3	000011	0003
Diamonds	4	000100	0004
Hearts	5	000101	0005
Spades	6	000110	0006
Clubs	7	000111	0007
Ace (Black)	8	001000	0008
2 (Black)	9	001001	0009
3 (Black)	10	001010	000A
4 (Black)	11	001011	000B
5 (Black)	12	001100	000C
6 (Black)	13	001101	000D
7 (Black)	14	001110	000E
8 (Black)	15	001111	000F
9 (Black)	16	010000	0010
10 (Black)	17	010001	0011
Jack (Black)	18	010010	0012
Queen (Black)	19	010011	0013
King (Black)	20	010100	0014
Ace (Red)	21	010101	0015
1 (Red)	22	010110	0016
2 (Red)	23	010111	0017
3 (Red)	24	011000	0018
4 (Red)	25	011001	0019
5 (Red)	26	011010	001A
6 (Red)	27	011011	001B
7 (Red)	28	011100	001C
8 (Red)	29	011101	001D
9 (Red)	30	011110	001E
10 (Red)	31	011111	001F
Jack (Red)	32	100000	0020
Queen (Red)	33	100001	0021
King (Red)	34	100010	0022
P	35	100011	0023
L	36	100100	0024
A	37	100101	0025
Y	38	100110	0026
E	39	100111	0027
R	40	101000	0028
D	41	101001	0029
E	42	101010	002A
A	43	101011	002B
L	44	101100	002C
E	45	101101	002D
R	46	101110	002E
W	47	101111	002F
I	48	110000	0030
N	49	110001	0031
S	50	110010	0032
!	51	110011	0033
T	52	110100	0034
I	53	110101	0035
E	54	110110	0036

Figure 4: Grid Memory

2.3 Calculations/Analysis/Drawings

<u>Grid Memory Calculations</u>

- I am implementing a 20x15 grid with each cell containing 32x32 pixels.

$$\frac{640pixels}{32pixels} = 20\ cells\ across$$

$$\frac{480pixels}{32pixels} = 15\ cells\ down$$

- My cell rows would be represented with 4-bits and cell columns would be represented with 5-bits. From this,

$$2^4 \times 2^5 = 300\ cells$$

- This means that my BRAM would have 300 entries.

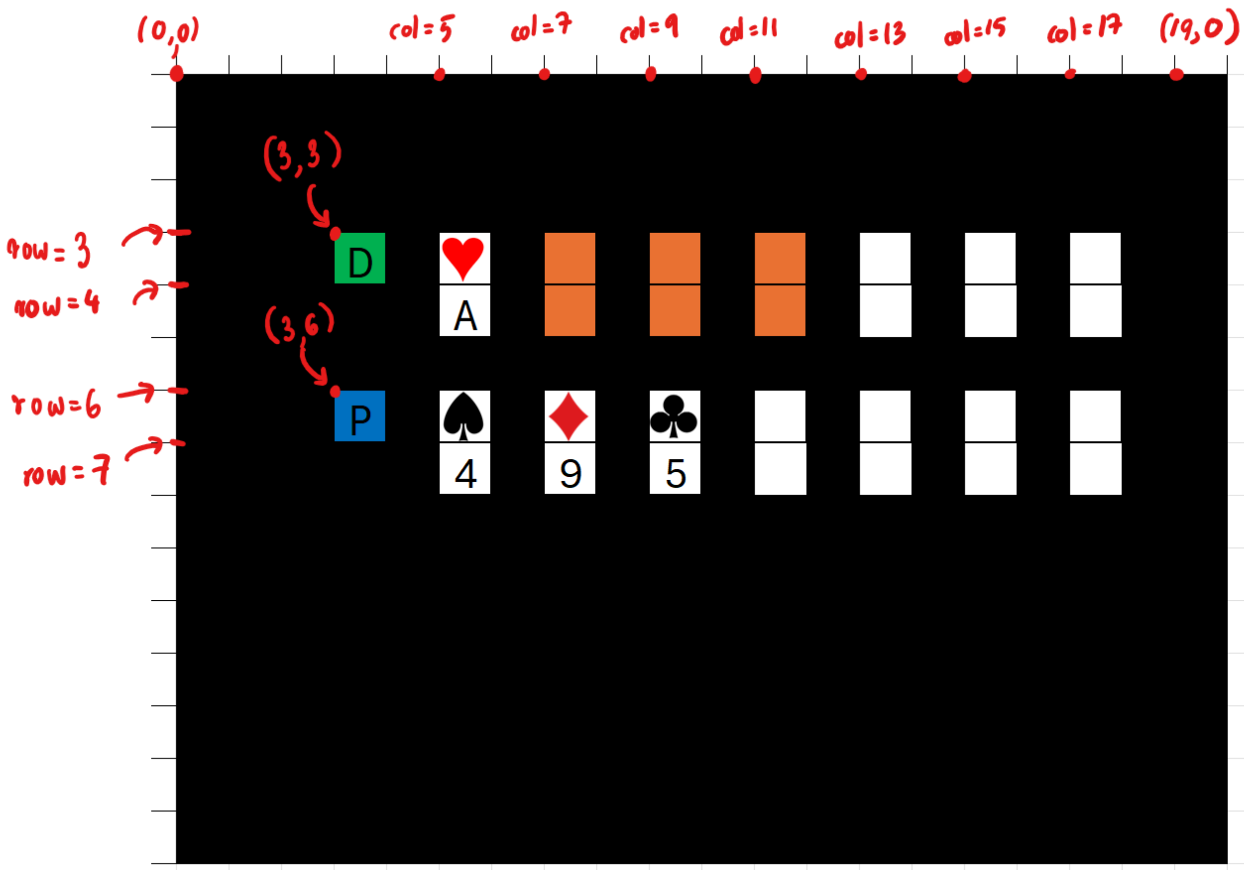


Figure 5: Gameplay Screenshot

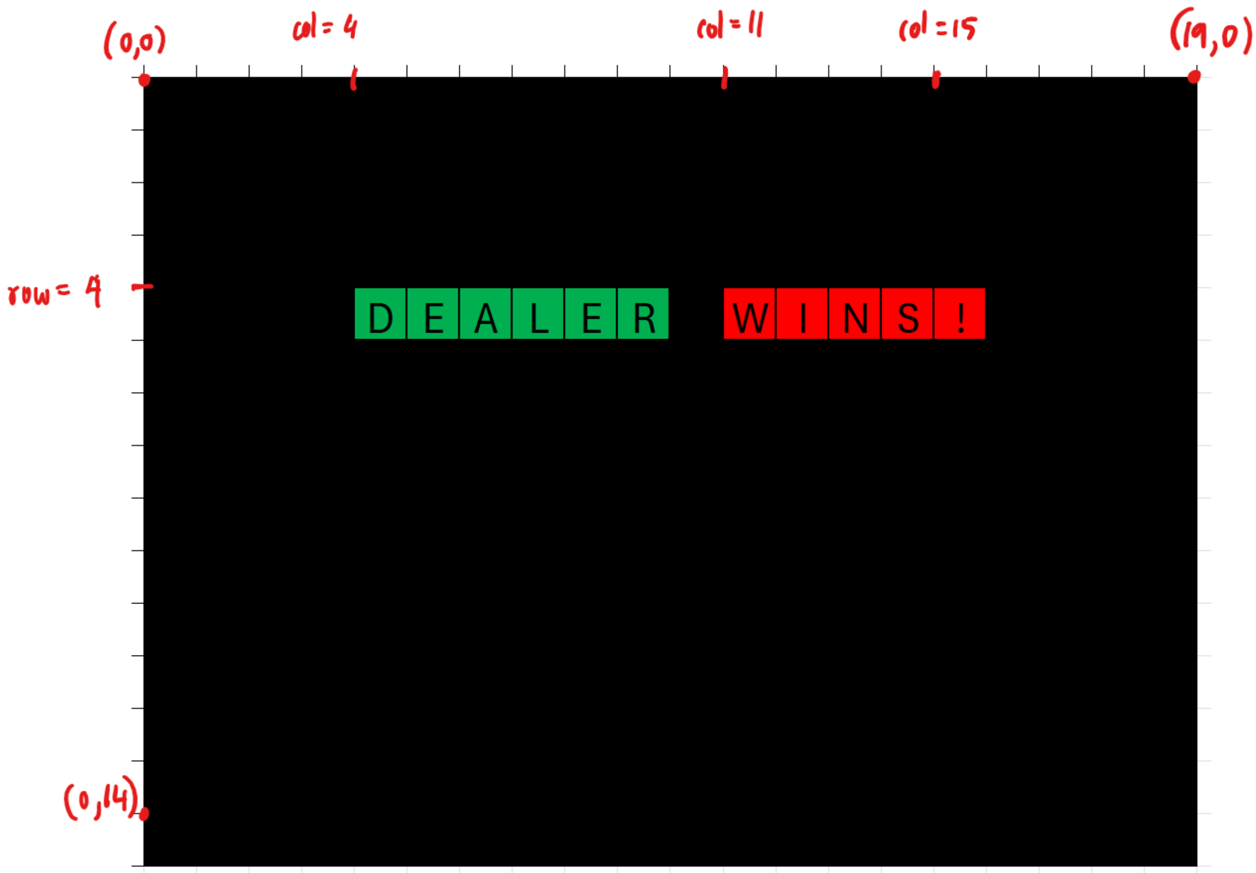


Figure 6: End result screenshot

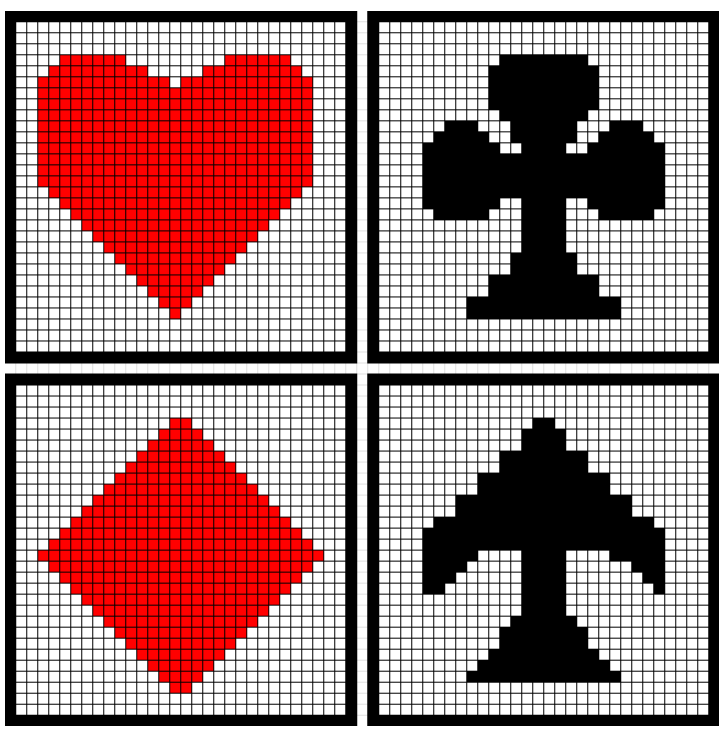


Figure 7: 32x32 pixel cells containing suits

Figure 5 shows the expected output of the display when the game is being played. The orange cards shown on the dealers hand are cards that are in the dealers hand (but the player cannot see them).

Figure 6 shows one of the result possibilities, this case where the Dealer wins.

Figure 7 shows the 32x32 pixel cells that would contain the suits of the deck of cards.

On figures 5, 6, and 7, I have only noted the most important coordinates that would be needed when interfacing this.

2.4 Milestone I

This milestone will mostly consist of putting all the necessary files together and setting up microblaze. After setting up microblaze and other files, as seen in the level-1 design, set up an array of a deck of cards and use SDK to test different combinations of cards being printed to the screen. You do not have to implement the game play yet, although encouraged.

2.5 Milestone II

Finish the gameplay code and ensure the game runs properly by testing with an unrandomized deck of cards and see if behavior is as expected. After this, set up gameplay to work with a random deck of cards, and ensure no cards will be repeated. You can use print statements on your .c file to see which card is used every time a "hit" is made by the player.

Ensure you have set up the mouse circuit needed and know how to implement it. If time permits, set up PS2 mouse to implement "hit" and "stand" by the player.

2.6 Updated Functionality and Requirements

Major corrections included in Dr York's feedback were updated in the Section 1.

Documentation

None.