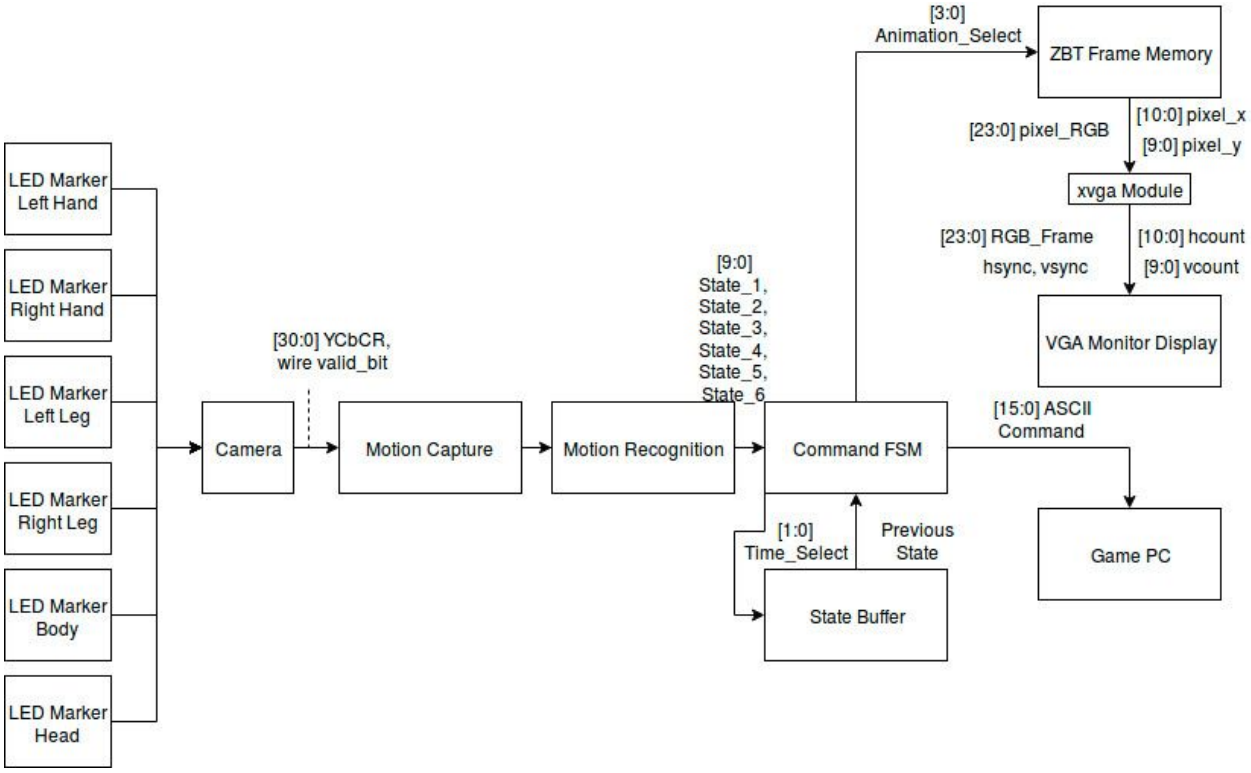
6.111: Final Project Proposal
Emanuel Perez
Ziwen Jiang

Introduction: Inspired by Ziwen's passion for virtual entertainment and procrastination, we decided to create our own interactive video game. We will position several differently colored LEDs on selective parts of the body (head, arms, body, and thighs) in order to track figure movement. This in turn will control a sprite on the screen (VGA monitor). The sprite should be able to walk, jump over obstacles, and collect coins.

Our project can be split into two main functionalities: animation and motion processing. Under the animation aspect, there will be game logic that appropriately updates the video display. With respect to the motion processing, there needs to be object tracking for the LEDs.

Block Diagram: (major modules and information flowing between them)
Our general block diagram contains 4 main modules. The first module is the motion capturing module, in which the camera captures the motion of markers on the gamer and transmits the video signal to FPGA. The second module is the motion recognition module that identifies the movement of the marker. Specific movements correspond to commands we will send to the character, which will be realized by the third module. The fourth module is the graphics module that receives commands and generates animation.



The inputs and outputs of each module are listed as following:

- YCbCR: 31 bit YCbCr value from NTSC camera
- Valid_bit: data valid bit to indicate when current pixel is valid
- State_1, State_2, State_3, State_4, State_5, State_6: the current position states of 6 markers, which include X and Y coordinates recognized by the system.
- Time_Select: select the reading time frame between the current position and previous positions of different marker.
- Previous State: previous X-Y position of each marker
- Animation_Select: select corresponding animation frames for each motion state
- pixel_RGB: RGB color for each pixel in stored frame
- Pixel_x, pixel_y: the x and y coordinates for pixels in each stored frame
- Hsync, Vsync: horizontal and vertical sync signal generated by the *xvga* module.
- Hcount, Vcount: counts pixels on the current scan line, generated by the *xvga* module.
- ASCII command: ASCII code of corresponding keyboard press that controls the game

Modules

Module 1. Motion Capture (Emanuel): In order to effectively capture movement, brightly colored LEDs will be used to provide high contrast relative to the environment. An NTSC camera will be used to track the brightly colored LEDs. The module will read an image frame YCbCR (as a series of pixels) and store the data into ZBT on the Nexys4.

Module 2. Motion Recognition (Emanuel): From the data that is stored, this module will be responsible for determining the location (x and y coordinates expressed as 8 bits) of each brightly colored LED with wire valid_bit to indicate that the current pixel is valid. The location will be determined as the center of mass for each brightly colored LED. We will use thresholding parameters in order to define boundaries.

Module 3. Command Converter (Ziwen):
For command converter module, we will create a command conversion FSM in FPGA to correlate the motion of different markers and specific command we want to send to control the game character. For the basic performance of our system, we will have six states: up, down, left, right, halt, and jump. The motion recognition module takes the inputs of the movement path of each marker. Command conversion FSM will track the inputs in a certain time frame by storing the positions of each marker in a buffer and using the relative motion of every marker in a certain time frame to decide the system state and command. The length of the time frame can also be changed so that we can decide the recognition resolution of the system.
For example, if the marker on the body core moves up and down in a large magnitude within a short time period, we will correlates that to jump. The relative motion between certain markers can also corresponds to more complex commands that can be implemented for a stretch goal.
There are two types of output from module that we plan to try. The first output is direct graphics change on the monitor that the FPGA connects to. For this output, we will create a simple game character and display it on the monitor, while the command module will control the movement of the character. The second output will be ASCII code converted from specific commands. The output will correspond to either keyboard presses and mouse click to control game on a separate computer. We will write python program that recognize ASCII code and converts it through HID protocol.

Module 4. Graphics (Ziwen):
As mentioned in the previous module, we plan to implement two kinds of displays: game character display on monitor through VGA, and game on another computer.
For the game character display, we plan to create 10 FPS anime for 2 seconds for each action. The frames will be saved on ZBT memory. Each frame will be displayed in RGB color and transported from FPGA kit to monitor, following what we did in lab 3 using xvga module. The control of the game character will be done directly in FPGA command FSM module. 3-bit animation select signal will be sent to the ZBT memory to choose the animation that is correlated to the current state. A reach goal will be creating a simple game environment that interacts with the game character, such as SuperMario.
For game on another computer, we will pick some FPS games, such as Counter Strike. The control command for the game will be from FPGA kit. The ASCII code will be picked up by external PC that runs the game, and converted to keyboard press or mouse movement through Python script. The reach goal will be have extra states if the command FSM that corresponds to more complicated motion, such as mouse movement, aim, etc.

Resources:

- NTSC Camera
- at least 6 pairs of differently brightly colored LEDs
- VGA monitor
- Nexys4

Challenges:

- Setting thresholds for color detection appropriately to avoid noise and other objects with the same intensity as the brightly colored LEDs
- Creating appropriate estimations for character movement on screen (such as jumping)