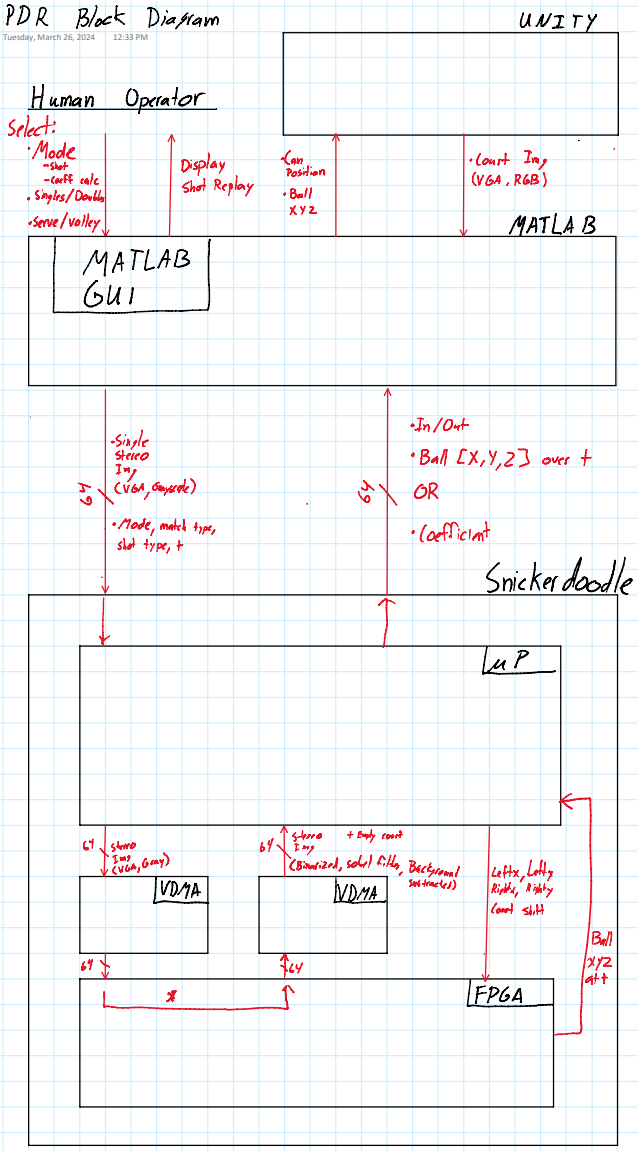
PDR Block Diagram

Tuesday, March 26, 2024 12:33 PM



= GUI

= MATLAB

= μP

= FPGA

= UNITY

Flow

1) User specifies the following parameters
   - Mode (In/Out or coeff calc)
   - Shot number (choose which shot is being illustrated over t in unity)
   - Shot type (Serve or Volley)
   - Match type (Singles or Doubles)

2) User selects "Run"

3) MATLAB initializes the board by sending [Mode, Shot type, Match, t=0] to snickerdoodle

4) Recieves initialization parameters, sets corresponding variables, and sends request for Frame n from MATLAB given current t (initially 0)

5) Receives request for frame n at t_n and uses relevant shot number function to generate ball X, Y, Z at t_n

6) Requests frame n from UNITY with Cam position set and Ball position = X_n, Y_n, Z_n. Will also request same frame with baseline added to cam position to make stereo img.

7) UNITY generates image given Ball and Cam position and sends back to MATLAB

8) MATLAB receives frame n (VGA / RGB) and grayscales the image. Sends Left and Right images together along with t to Snickerdoodle

64 => byte

0	Left img
1	Right img
2	·
3	·
4	·
5	·
6	·
7	t

Display img frame for replay?

9) μP receives stereo gray frame n and immediately just passes it through to FPGA thru the VDMA for processing. Start timer

10) FPGA receives stereo gray frame n, pixel by pixel and performs the following filters on it

11) Sobel filter frame n (Left and Right)
12) Binarize filtered frame n (Left and Rig): Ball left and Ball Right
13) Filter out Ball leaving only straight lines: Empty left and Empty Right
14) Pixel by Pixel perform background subtraction (Ball left − Empty left) = processed Left and processed Right

11.) FPGA sends the following frame n back to μP

64

0	processed Left
1	processed Right
2	Empty Left
3	Empty Right
4	
5	
6	
7	

12.) μP receives processed frame n back from FPGA and performs following

13.) Has Baseline Img for an empty Left and Right already stored (Binarized, VGA, sobel filter). And compares those baselines to received Empty Left and Empty Right to calculate the perceived shift that the cameras could have experienced from wind (x Left shift, y Left shift..)

14.) Performs centroid detection algo to find xLeft, y Left, x Right, y Right pixel coordinates of ball in left and right frames

15.) Applies pixel shift to all xLeft, yLeft, xRight, y Right
↑ xLeft shift...

May be unnecessary to send to FPGA ↘ 16.) uP passes xLeft, yLeft, xRight, y Right back to FPGA for quick calculation

17.) FPGA uses xLeft, yLeft, xRight, y Right to find ball X YZ in 3D space and sends XYZ back to uP

18.) uP stores ball XYZ at t.

19.) Stop timer for Δt. Request frame n+1 from MATLAB at $t = t + \Delta t$

Repeat 5-19 until entire shot processed

20.) Given [X, Y, Z] over t, calculate ~~t~~ at which Z = ground depth

21.) Depending on Mode
- Coeff = calculate velocity of ball before/after bounce t using [X Y Z] over t
- In/Out = at bounce t. is X Y within "In" X Y region

22.) Send Result to MATLAB

23.) Display Result