UART TO HDMI INTERFACE

TEAM GreenTeck

- Thor Smith
- Chun Ta Huang
- Hang Xie
- Xiao Zuguang

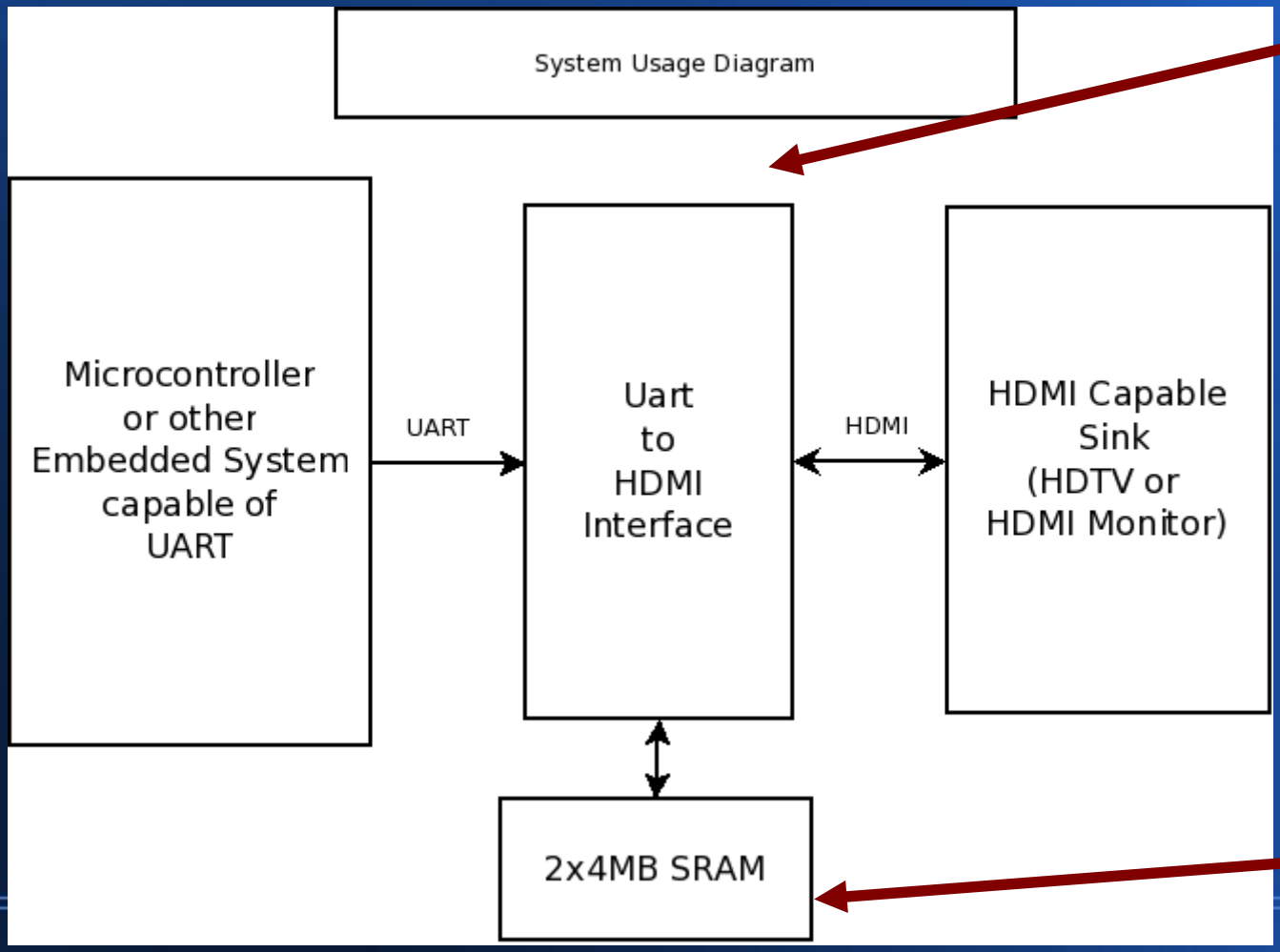
Overview

- Product:
  - UART To HDMI Interface
  - Low Level GPU
- Purpose
  - Allow a simple microcontroller to drive an high definition TV by separating video processing from the microcontroller.
  - Provide a cheap, simple, and fast product for use by students and hobbyists.

Overview

- Main Features
  - UART
  - Frame Decoding
  - GPU
  - 2x4MB SRAM Interface
  - HDMI Encoder

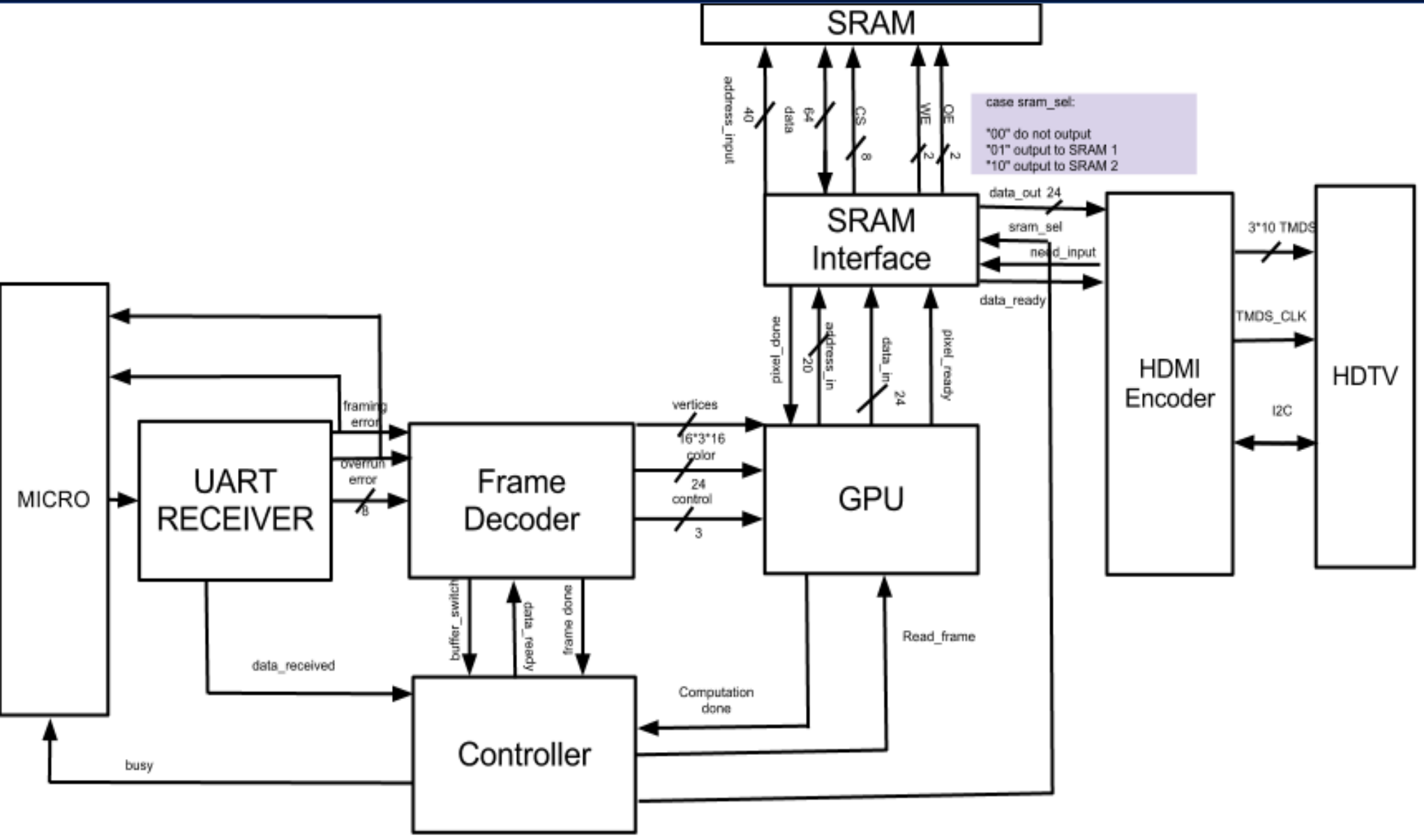
System Level Diagram



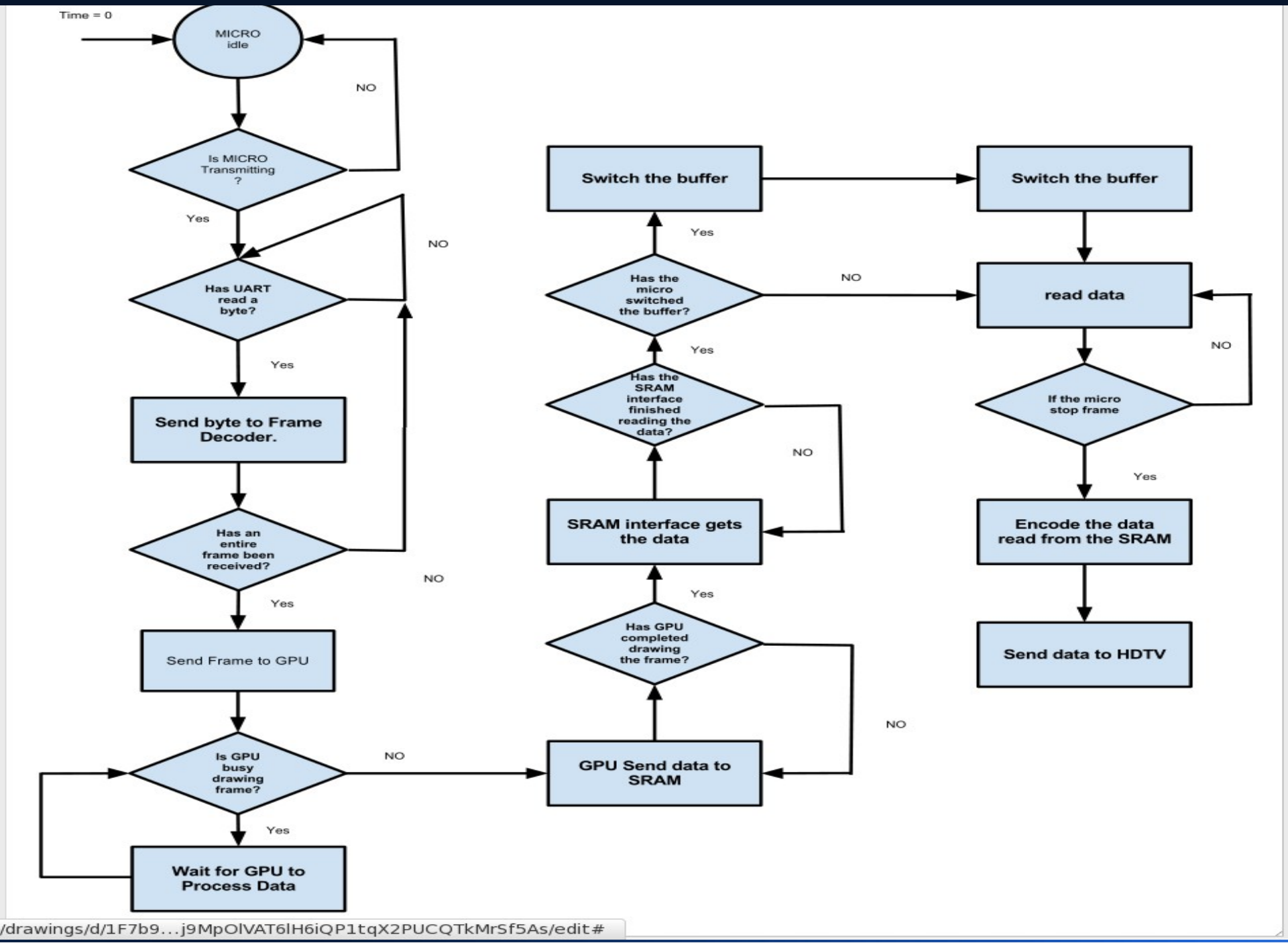
HDMI 1.3 Type A

CYM1851

Architecture Diagram



Flowchart



Detailed Diagrams

- UART
- Frame Decoder
- GPU
- Controller
- SRAM Interface
- HDMI Encoder

UART

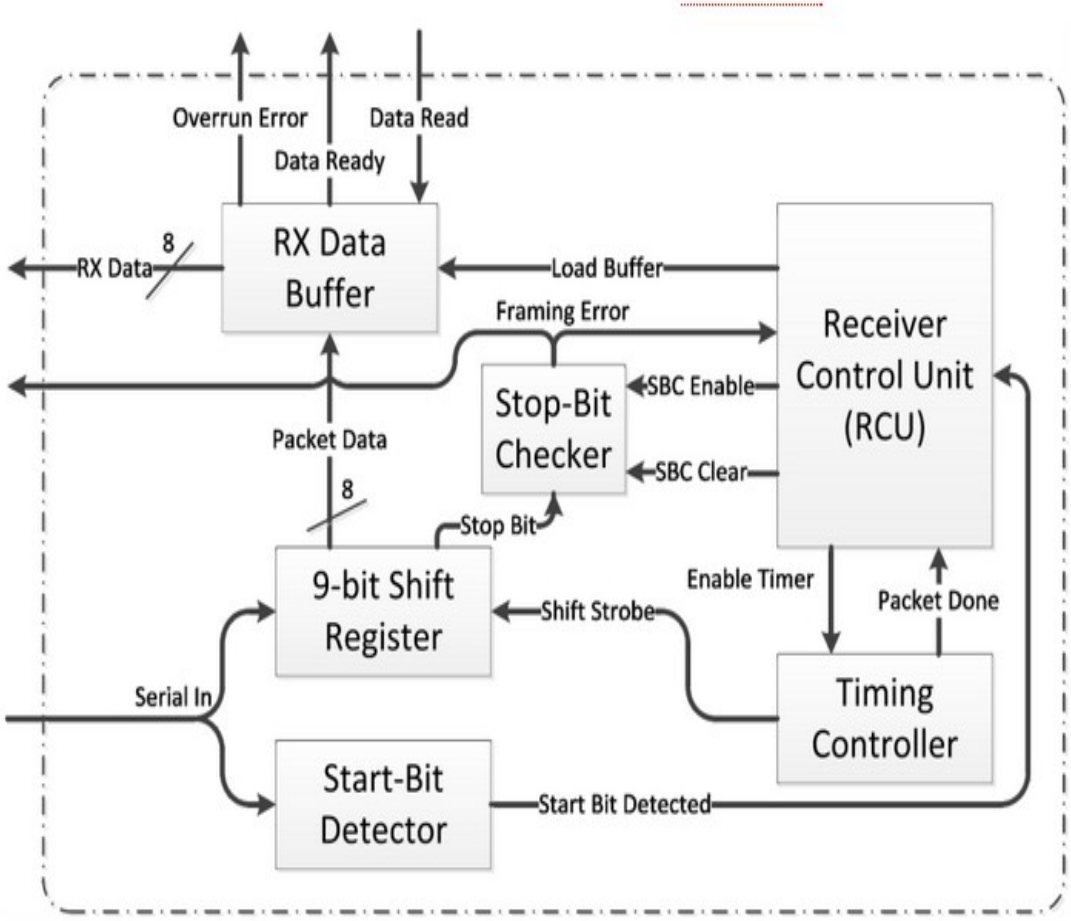
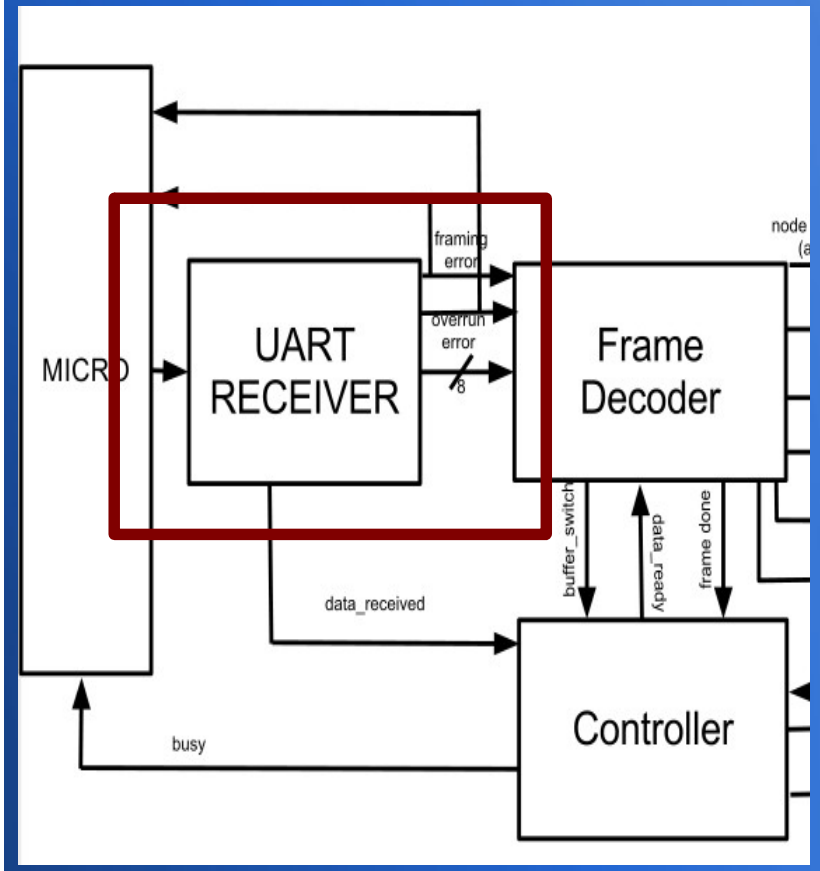
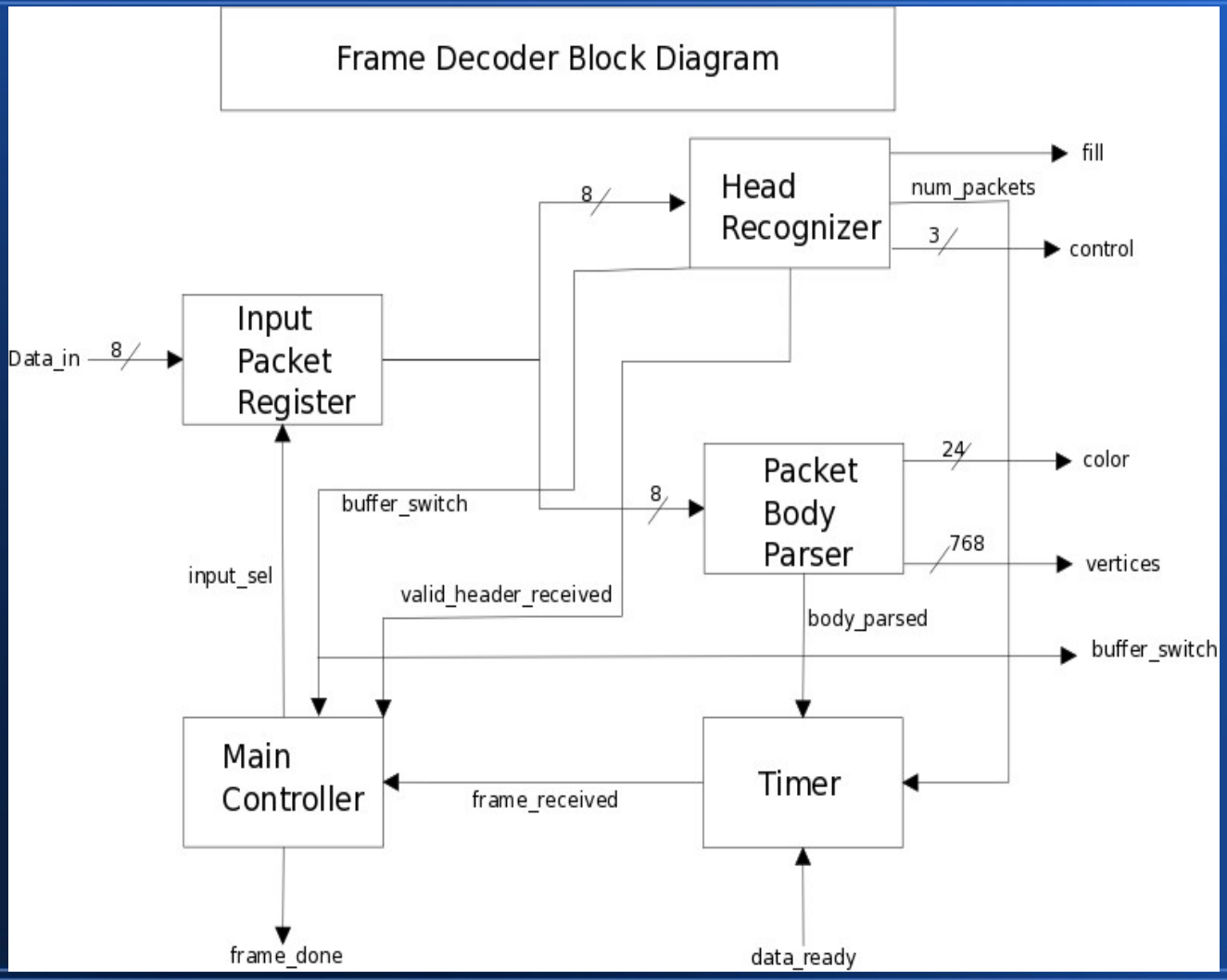


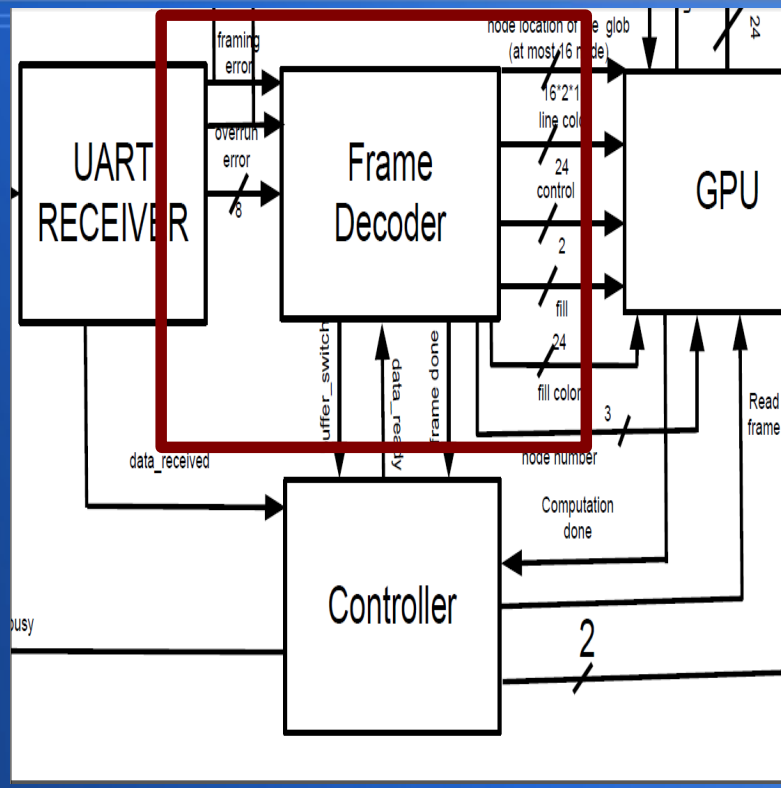
Figure 4. UART block diagram



Taken from the lab manual for ECE337 ASIC Design class Lab 5 P.9

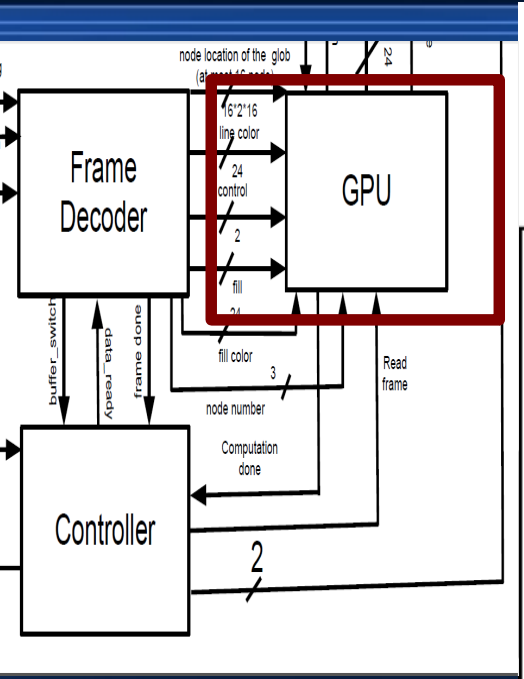
Frame Decoder

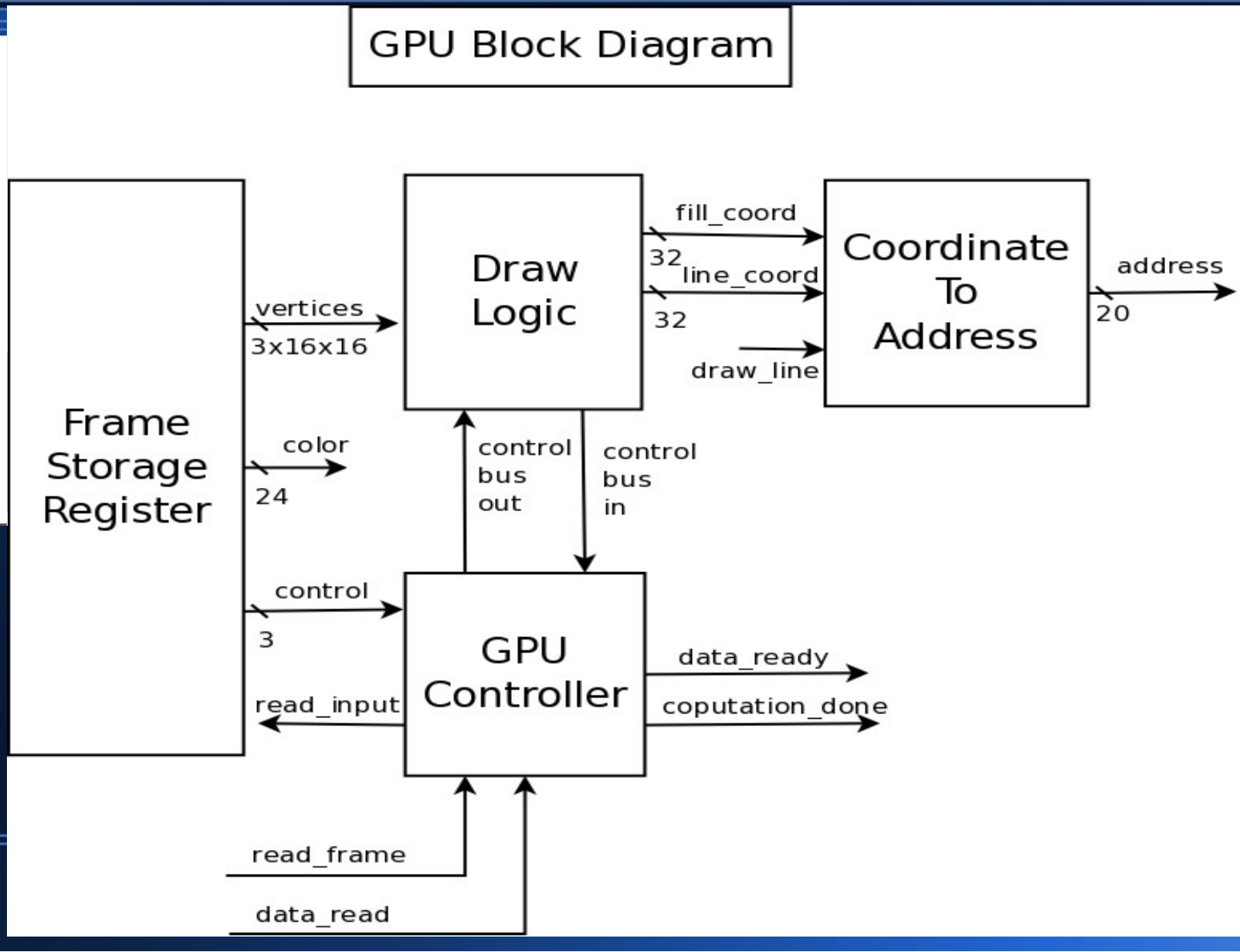




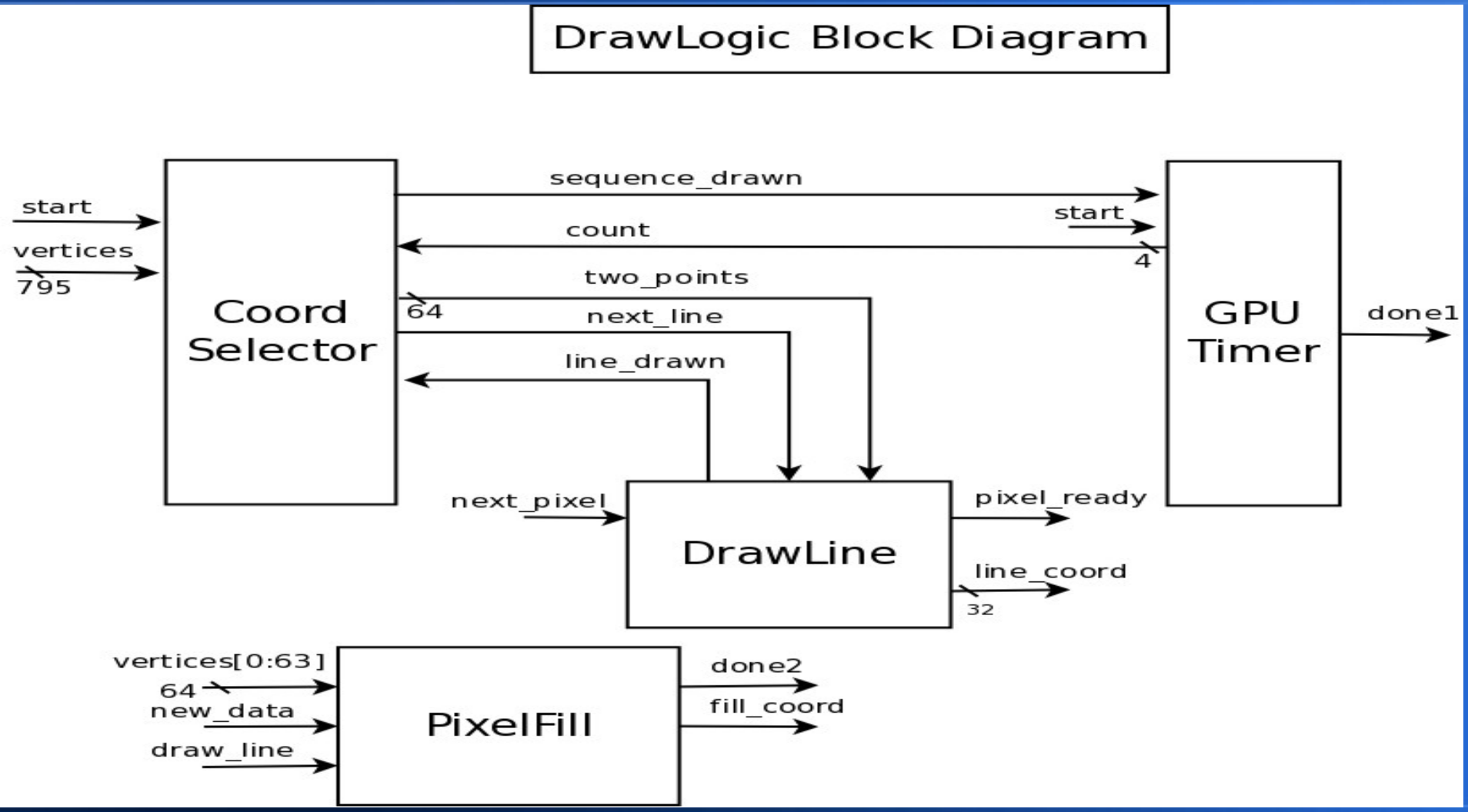
The Frame Decoder is not completely specified.

GPU

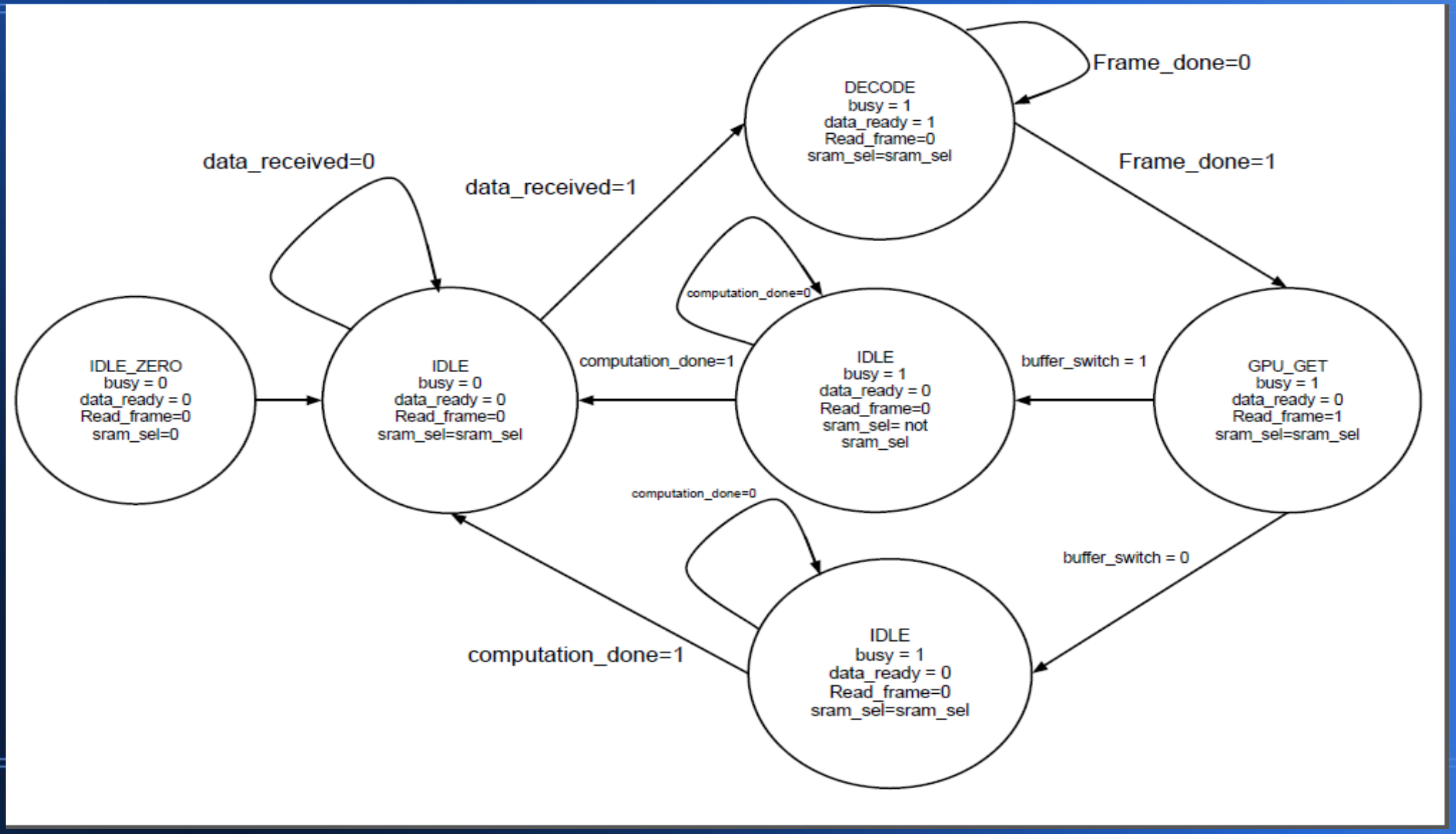

node location of the glob
24
16*2*16
line color
24
control
2
fill
Frame Decoder
GPU
buffer_switch
data_ready
frame done
fill color
3
node number
Read frame
Computation done
Controller
2


GPU Block Diagram
Frame Storage Register
vertices
3x16x16
color
24
control
3
read_input
read_frame
data_read
Draw Logic
control bus out
control bus in
fill_coord
32
line_coord
32
draw_line
Coordinate To Address
address
20
GPU Controller
data_ready
coputation_done

Draw Logic

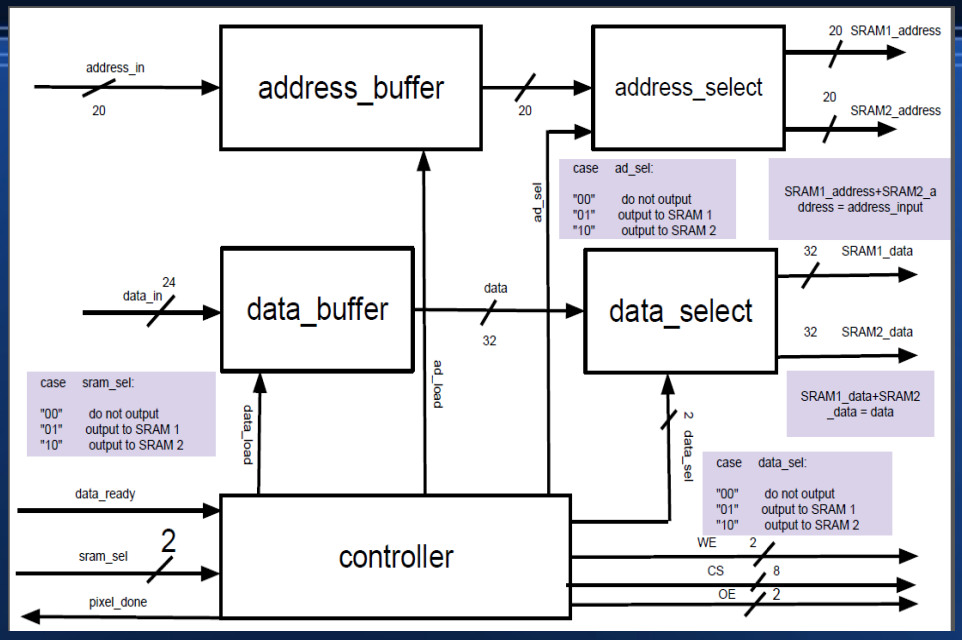

DrawLogic Block Diagram
start
vertices
795
Coord Selector
sequence_drawn
count
4
start
two_points
64
next_line
line_drawn
GPU Timer
done1
next_pixel
DrawLine
pixel_ready
line_coord
32
vertices[0:63]
64
new_data
draw_line
PixelFill
done2
fill_coord

Controller

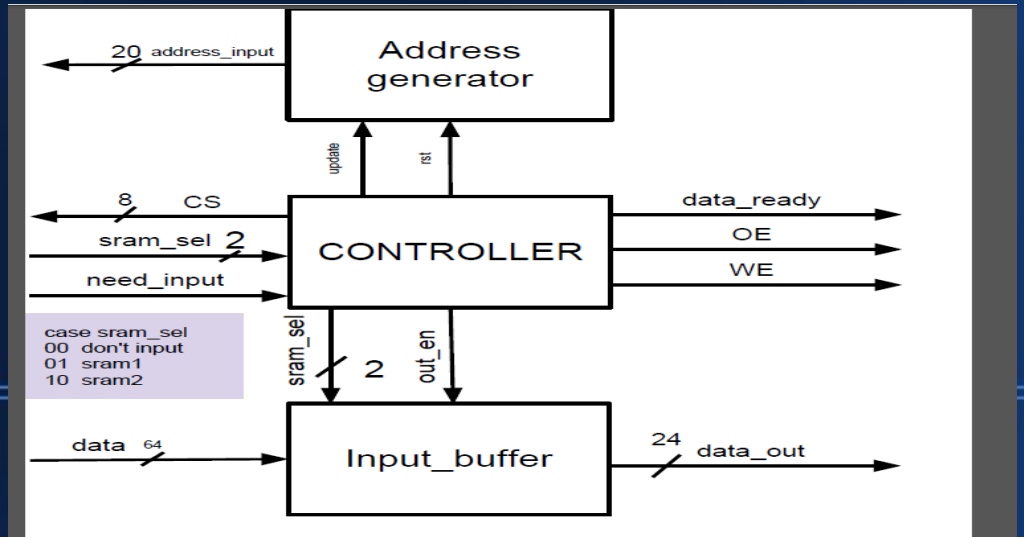

DECODE
busy = 1
data_ready = 1
Read_frame=0
sram_sel=sram_sel
Frame_done=0
Frame_done=1
data_received=0
data_received=1
IDLE_ZERO
busy = 0
data_ready = 0
Read_frame=0
sram_sel=0
IDLE
busy = 0
data_ready = 0
Read_frame=0
sram_sel=sram_sel
computation_done=0
computation_done=1
IDLE
busy = 1
data_ready = 0
Read_frame=0
sram_sel= not
sram_sel
buffer_switch = 1
GPU_GET
busy = 1
data_ready = 0
Read_frame=1
sram_sel=sram_sel
computation_done=0
buffer_switch = 0
computation_done=1
IDLE
busy = 1
data_ready = 0
Read_frame=0
sram_sel=sram_sel

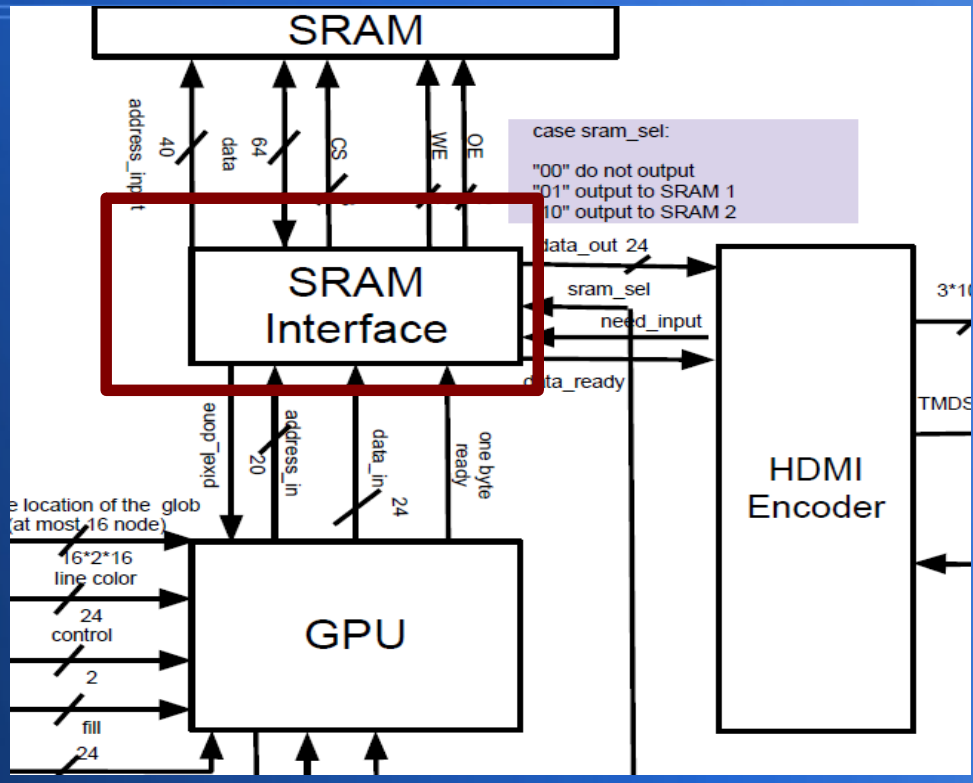
SRAM Interface

Sending data

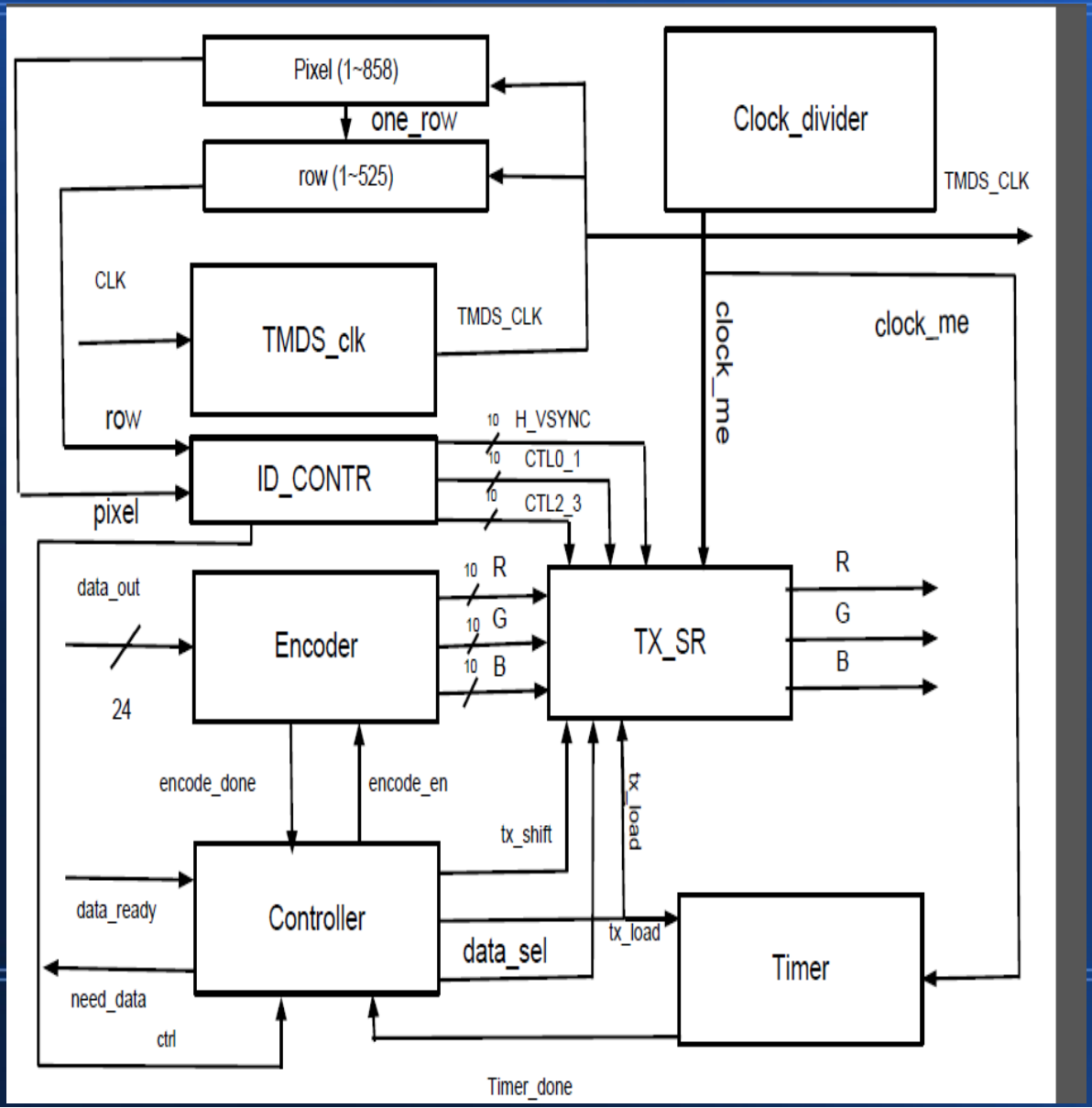


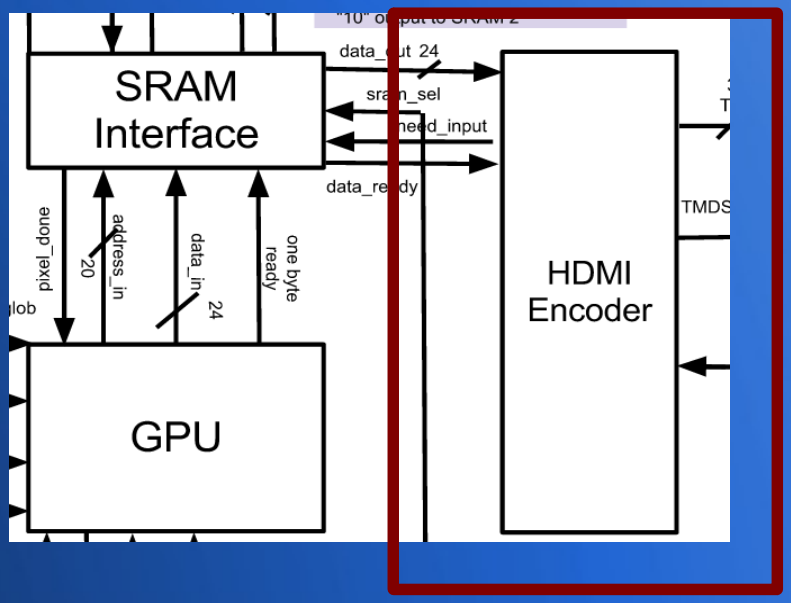
Extracting Data





HDMI Encoder





Area Budget

Core Area Calculation

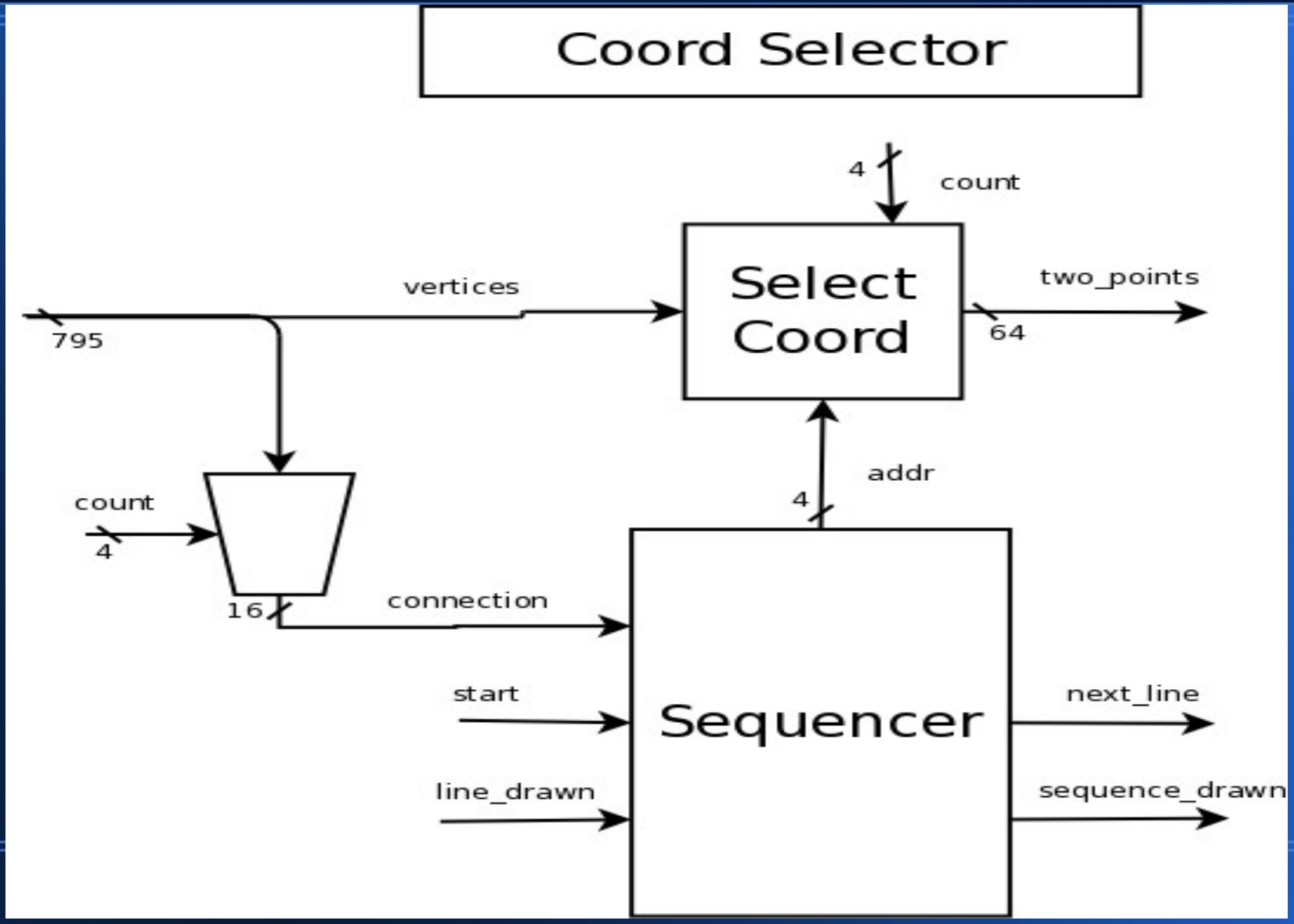
Name of Block	Category	FF count	Area()	Commints
UART RECEIVER	Reg.w/ Reset	38	91200	
Frame Decoder	Reg.w/ Reset	620	1488000	
GPU	Reg.w/ Reset	613	28448700	Flip Flop + Combinational Logic
SRAM Interface	Reg. w/ Reset	80	168000	
Controller	Reg.w/ Reset	3	7200	
HDMI Encoder	Reg. w/ Reset	56	130000	

Timing Budget

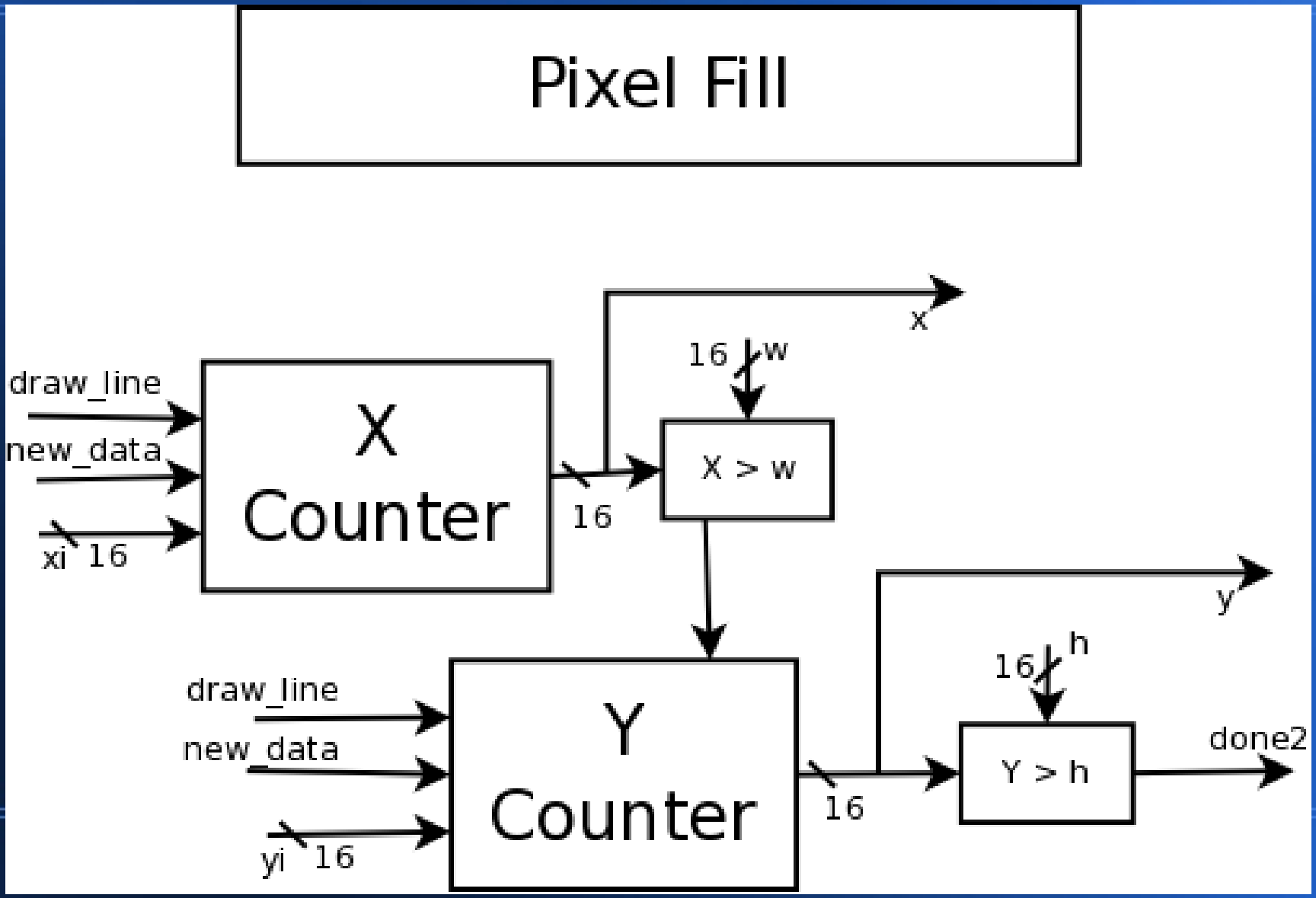
Starting component	Tp	Combination Logic	Tp	Ending Component	Tsetup/Tp	Total Delay
encoder output	0.1 ns	0.3 ns	0 ns	TX_SR	0.2	0.6 ns
address buffer	0.1	0 ns	0 ns	address_select	0.2	0.3 ns
controller register output	0.1	0.2	0	controller register input	0.2	0.5ns
UART shift register output	0.1	0.2	0	UART main controller register input	0.2	0.5ns
frame decoder register output	0.1	0.3	0	frame decoder register input	0.2	0.6ns

Q & A

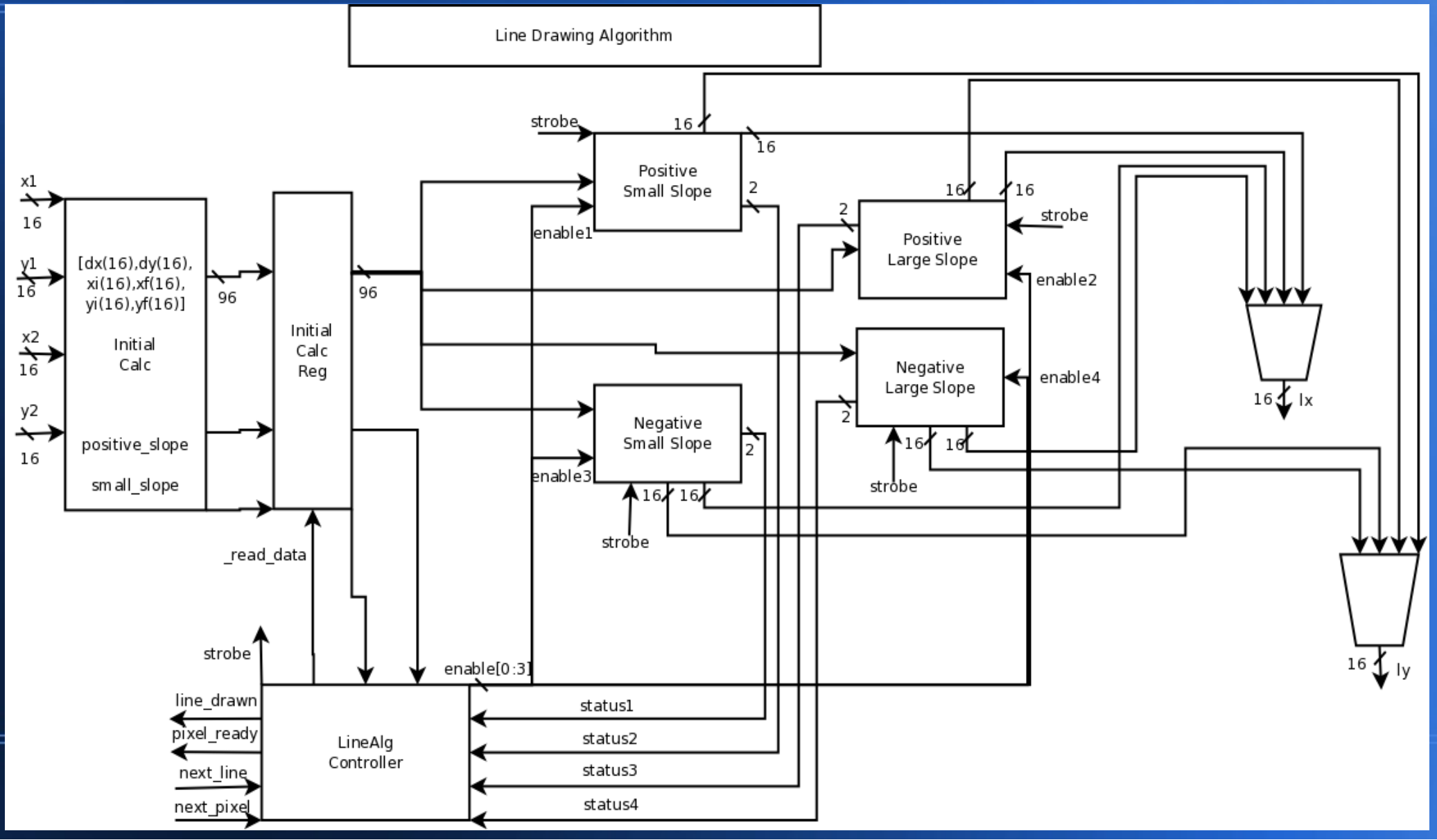
Coord Selector



Pixel Fill


Pixel Fill
x
16
w
draw_line
new_data
X
Counter
X > w
xi
16
16
y
16
h
draw_line
new_data
Y
Counter
Y > h
done2
yi
16
16

Line Drawing Algorithm



Positive Small Slope

Bresenham's algorithm is used to calculate the points. A variation of the pseudo code below is used:

```
Increment x from xi to xf
        Write x,y pixel to buffer
        err += dy
        if (err << 1) >= dx
            y += 1
            err -= dx
```

Coord Flattener