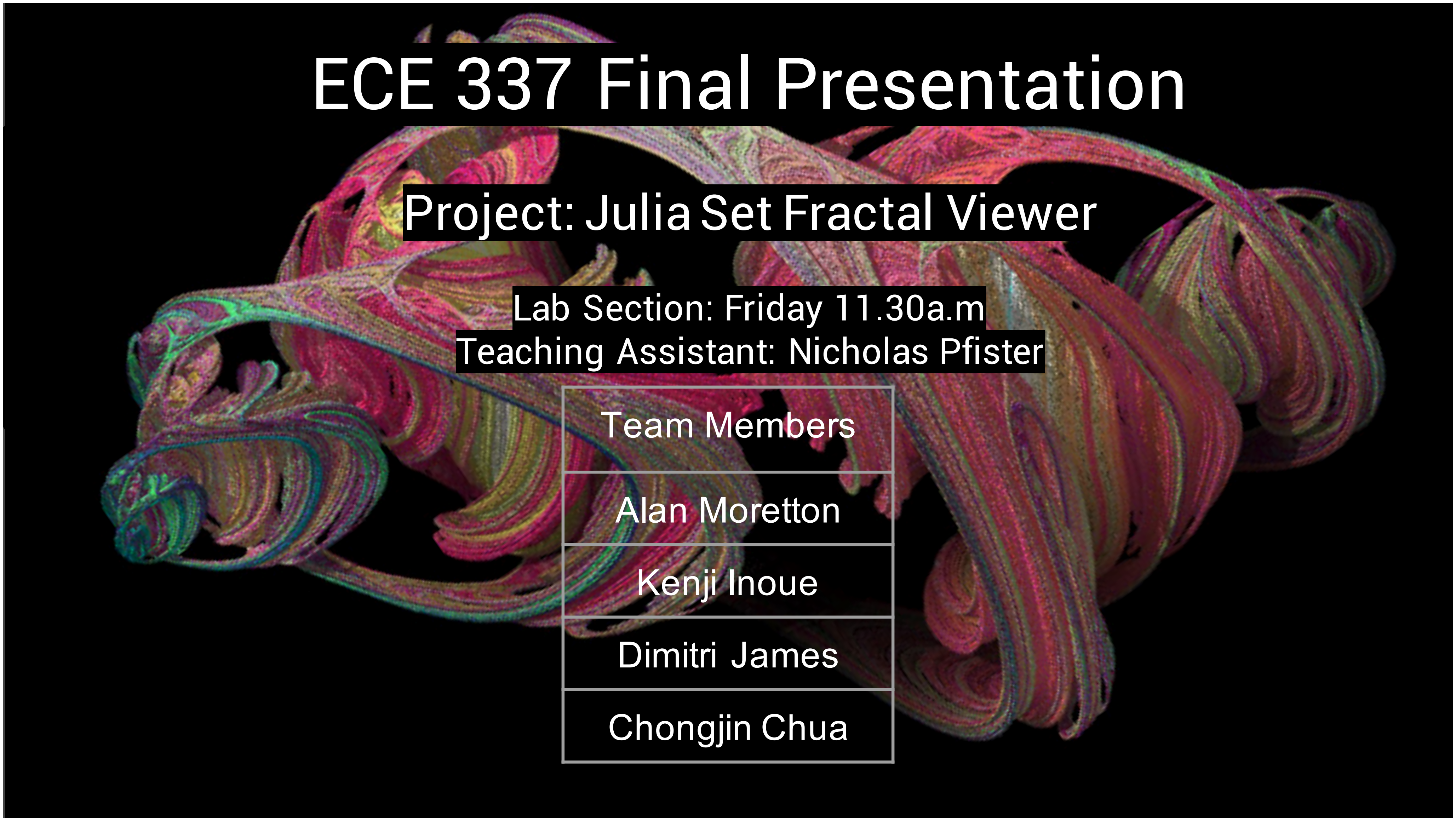
ECE 337 Final Presentation

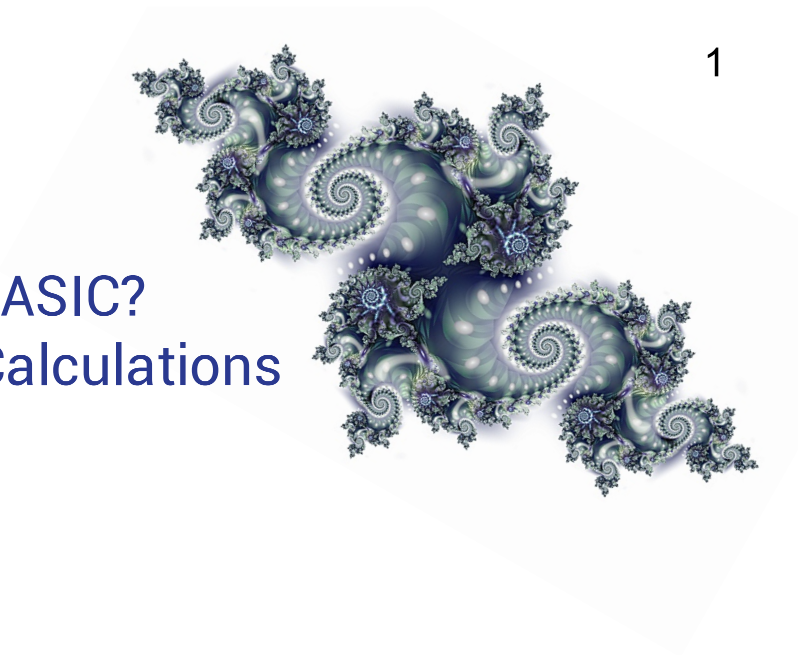
Project: Julia Set Fractal Viewer

Lab Section: Friday 11.30a.m
Teaching Assistant: Nicholas Pfister

Team Members
Alan Moretton
Kenji Inoue
Dimitri James
Chongjin Chua



Overview - Julia Set

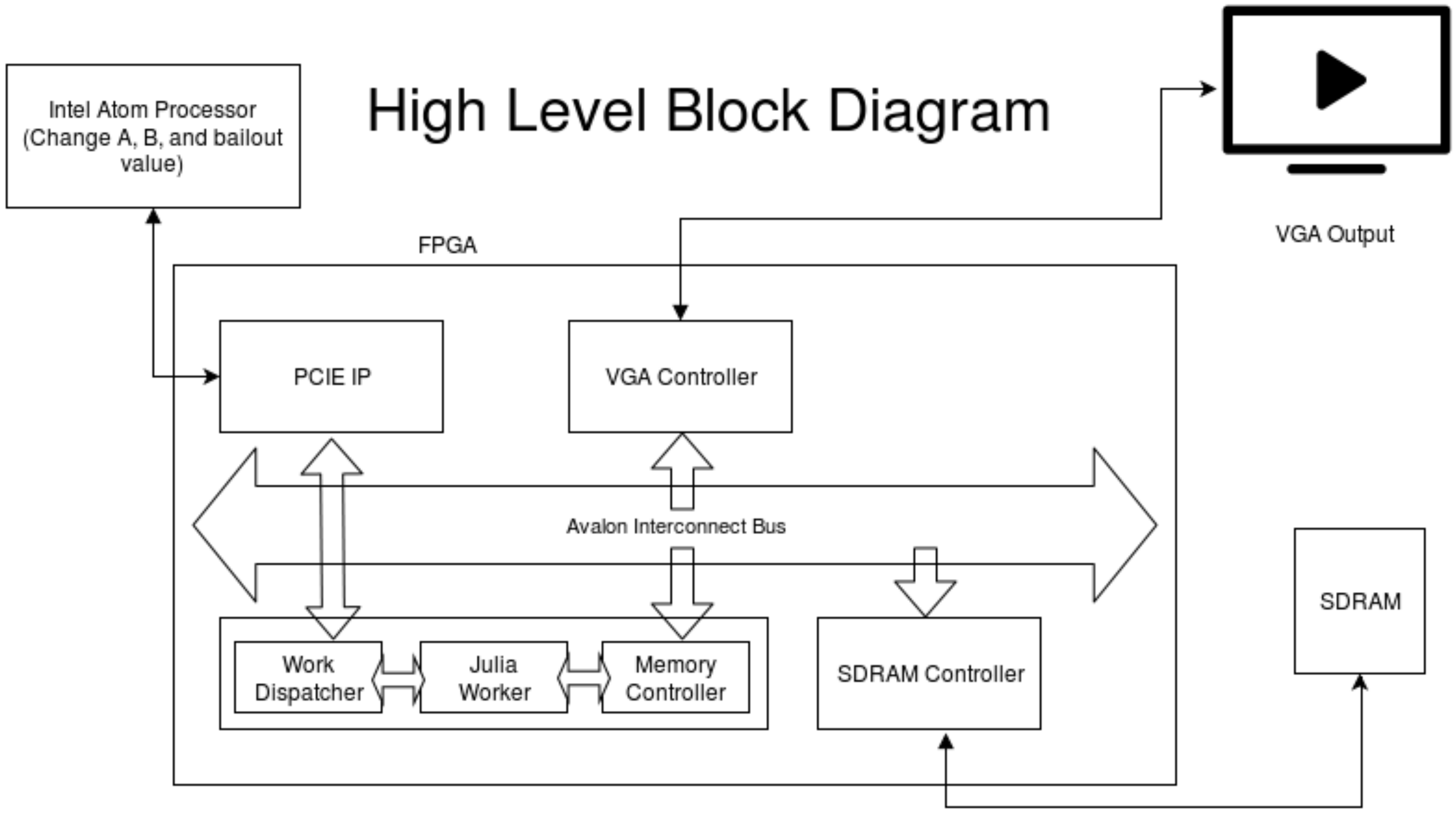


What is a Julia Set?
- Fractal
- Subset of Mandelbrot
- Chaotic

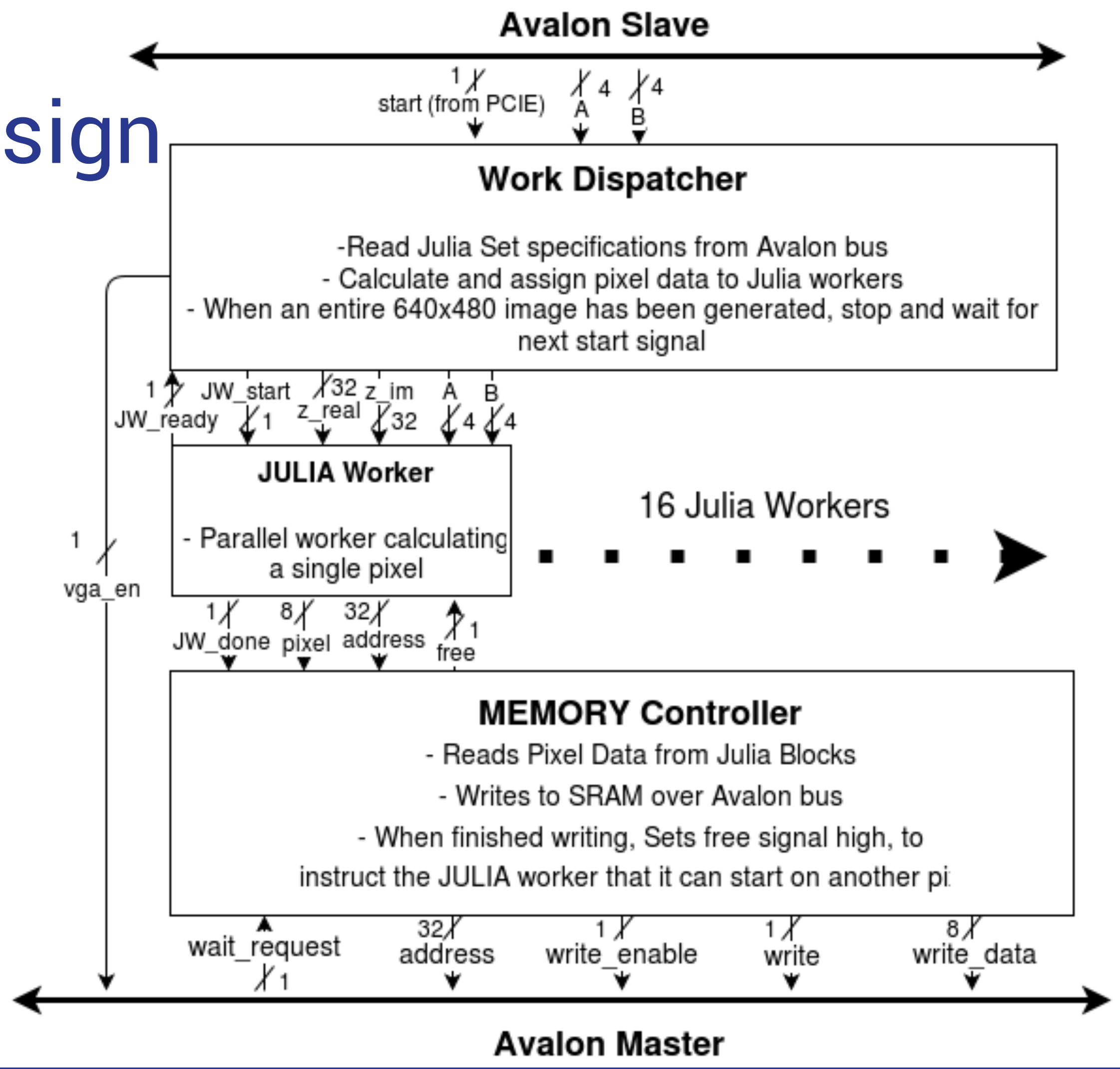
Why use an ASIC?
- Faster Calculations

What are its functions?
- CGI
- Simulate biological processes, interaction
- Future image compression

System Design



System Design


Avalon Slave
1
start (from PCIE)
4
A
4
B
Work Dispatcher
-Read Julia Set specifications from Avalon bus
- Calculate and assign pixel data to Julia workers
- When an entire 640x480 image has been generated, stop and wait for next start signal
1
JW_ready
JW_start
1
32
z_real
z_im
32
A
4
B
4
JULIA Worker
- Parallel worker calculating a single pixel
16 Julia Workers
1
vga_en
1
JW_done
8
pixel
32
address
1
free
MEMORY Controller
- Reads Pixel Data from Julia Blocks
- Writes to SRAM over Avalon bus
- When finished writing, Sets free signal high, to instruct the JULIA worker that it can start on another pi
wait_request
1
32
address
1
write_enable
1
write
8
write_data
Avalon Master

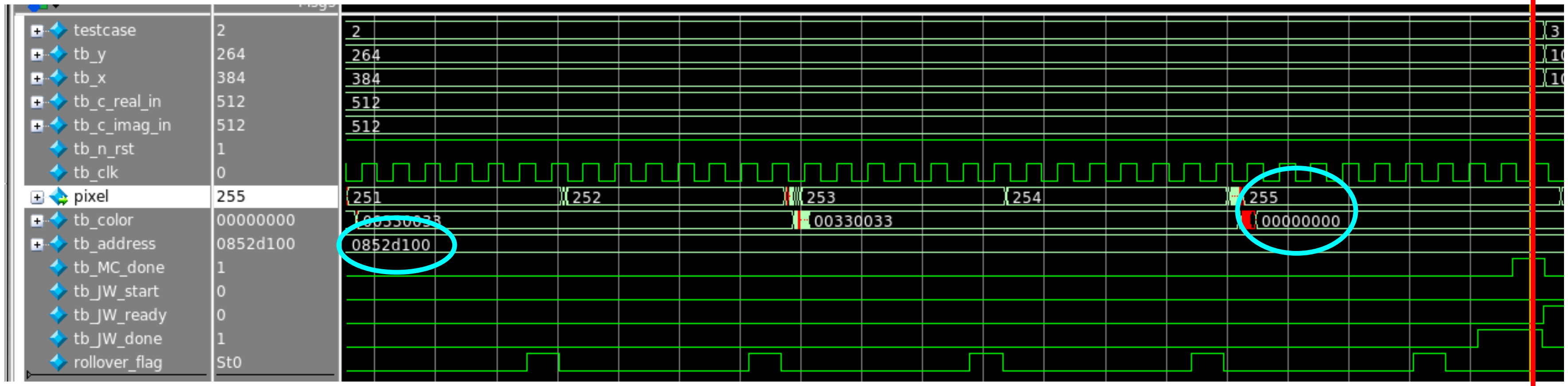
Fixed Criteria	P/F
Test benches exist for all top level components and the entire design. The test benches for the entire design can be demonstrated or documented to cover all of the functional requirements given in the design specific success criteria.	P
Entire design synthesizes completely, without any inferred latches, timing arcs, and, sensitivity list warnings.	P
Source and mapped version of the complete design behave the same for all test cases. The mapped version simulates without timing errors except at time zero.	P&F
A complete IC layout is produced that passes all geometry and connectivity checks.	P
The entire design complies with targets for area, pin count, throughput (if applicable), and clock rate.	P

DSSC	P/F
Calculate a working Julia set block with code that accurately calculates next values and address of memory location	P
Dispatcher correctly assigns work to workers that are available	P
Memory controller correctly assigns calculated values to memory	P
Julia set updates the display with the new zoomed image manually	F
Successfully integrate FPGA and Avalon Bus Interface	P&F
Different patterns using coefficients	P
Vga controller successfully read data from SDRAM and display data on screen	P

4

Results (julia worker) DSSC 1

Julia set algorithm:
Z_new = (Z_old)^2 + C



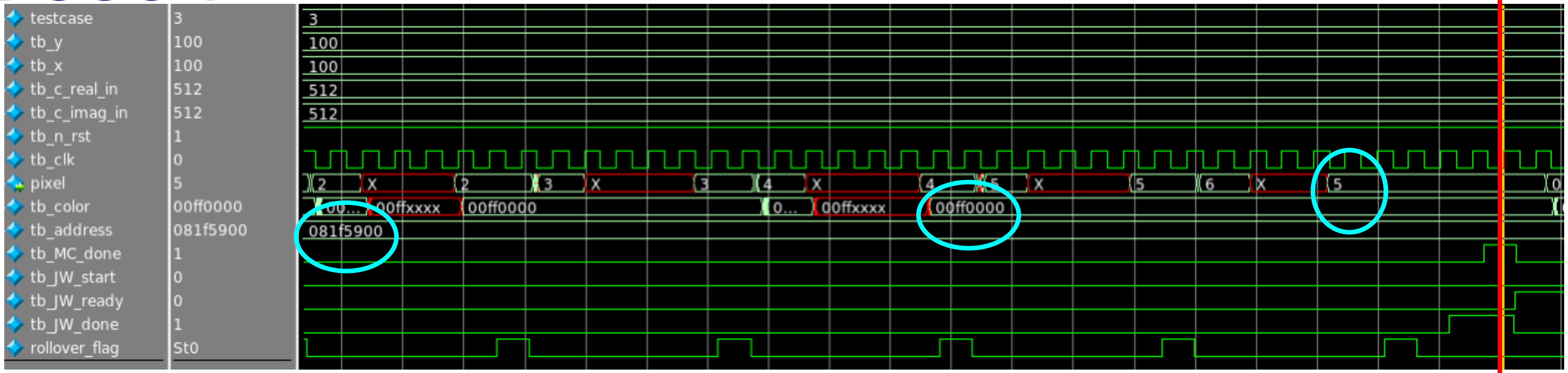
Red line: The positive clock edge where valid color and address are passed to Memory Controller.

Blue circle: Number of iterations = 255, Color = 00000000 (black)

Address = 0852d100 (x = 384, y = 264)

Results (julia worker) DSSC 1

Julia set algorithm:
Z_new = (Z_old)^2 + C



Red line: The positive clock edge where valid color and address are passed to Memory Controller.

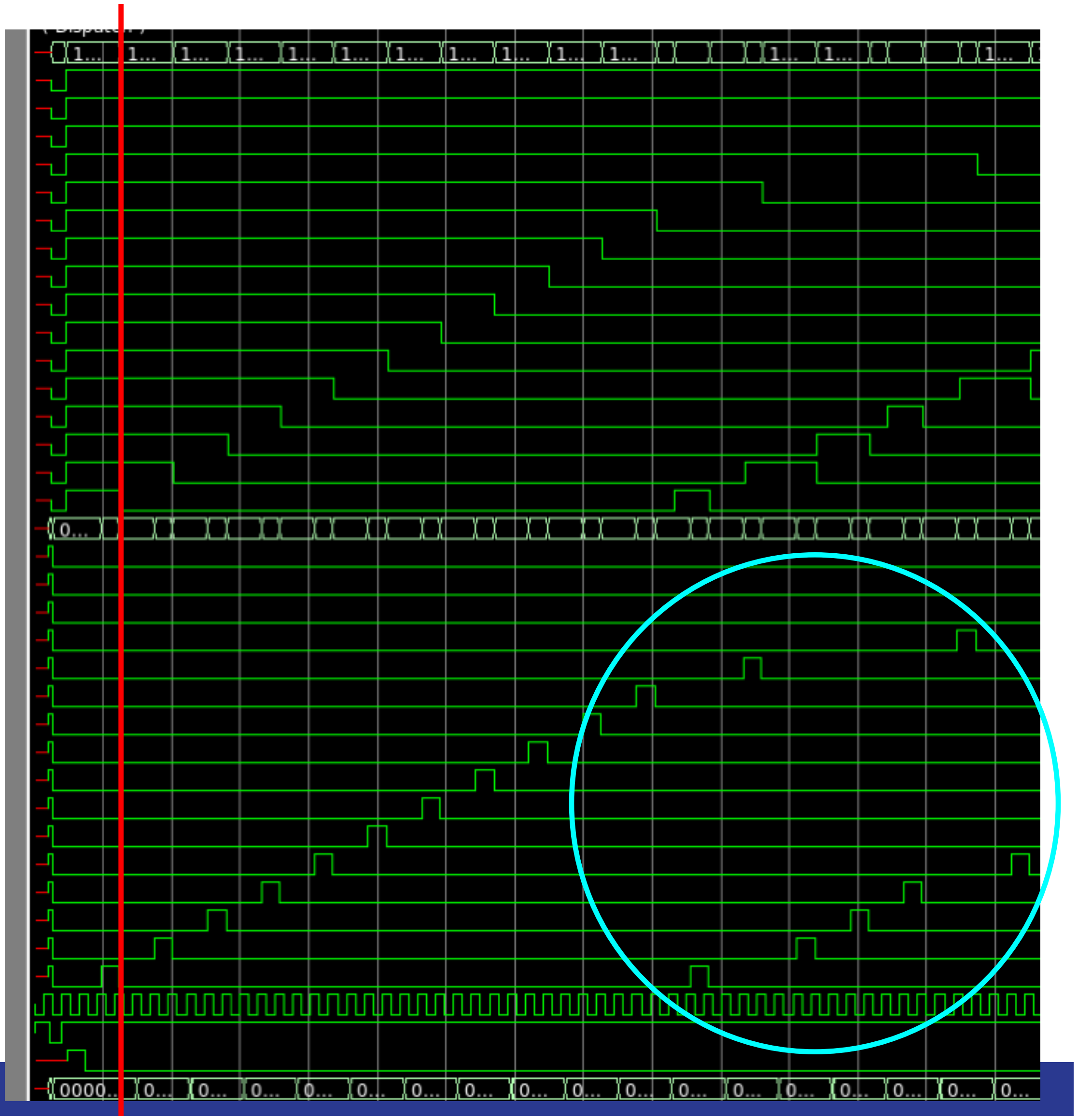
Blue circle: Number of iterations = 5, Color = 00FF0000 (red)

Address = 081f5900 (x = 100, y = 100)

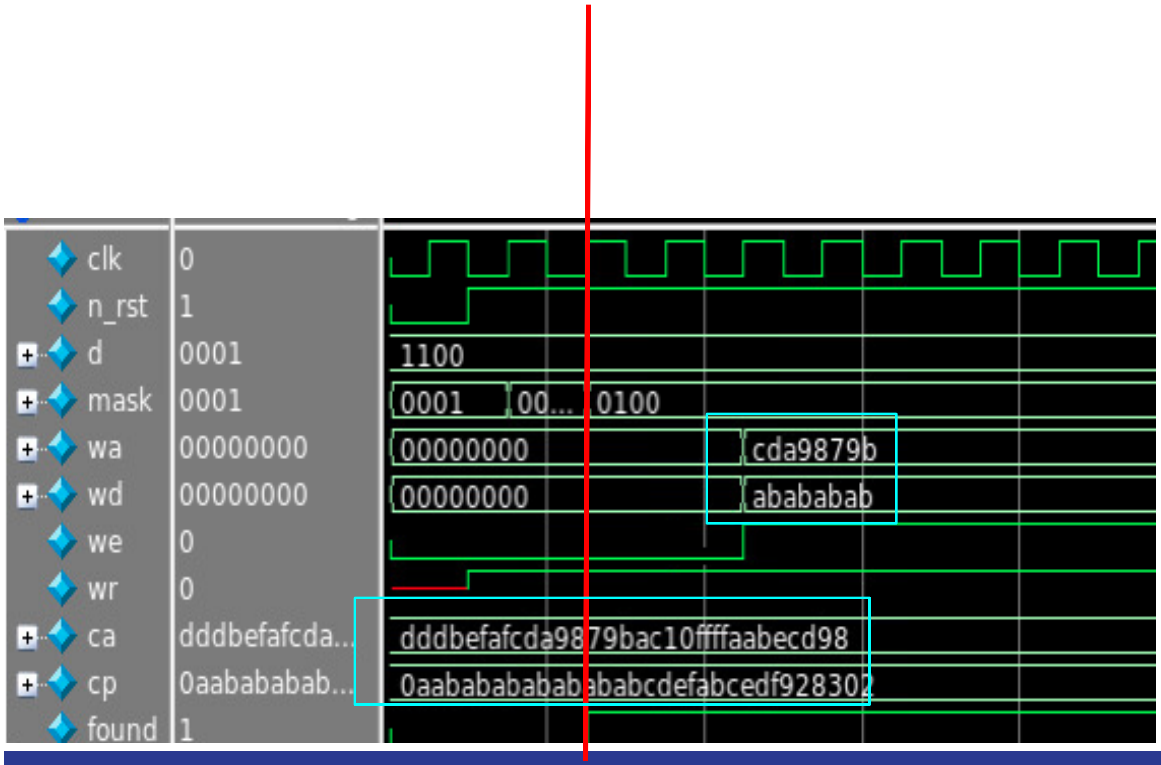
Results (dispatcher) DSSC 2

Red line: Shows dispatcher assigning jobs to workers as they are available and de-asserting the “done flag”

Blue circle: Shows dispatcher logic taking effect as jobs become more complicated



Results (memory ctrl.) DSSC 3

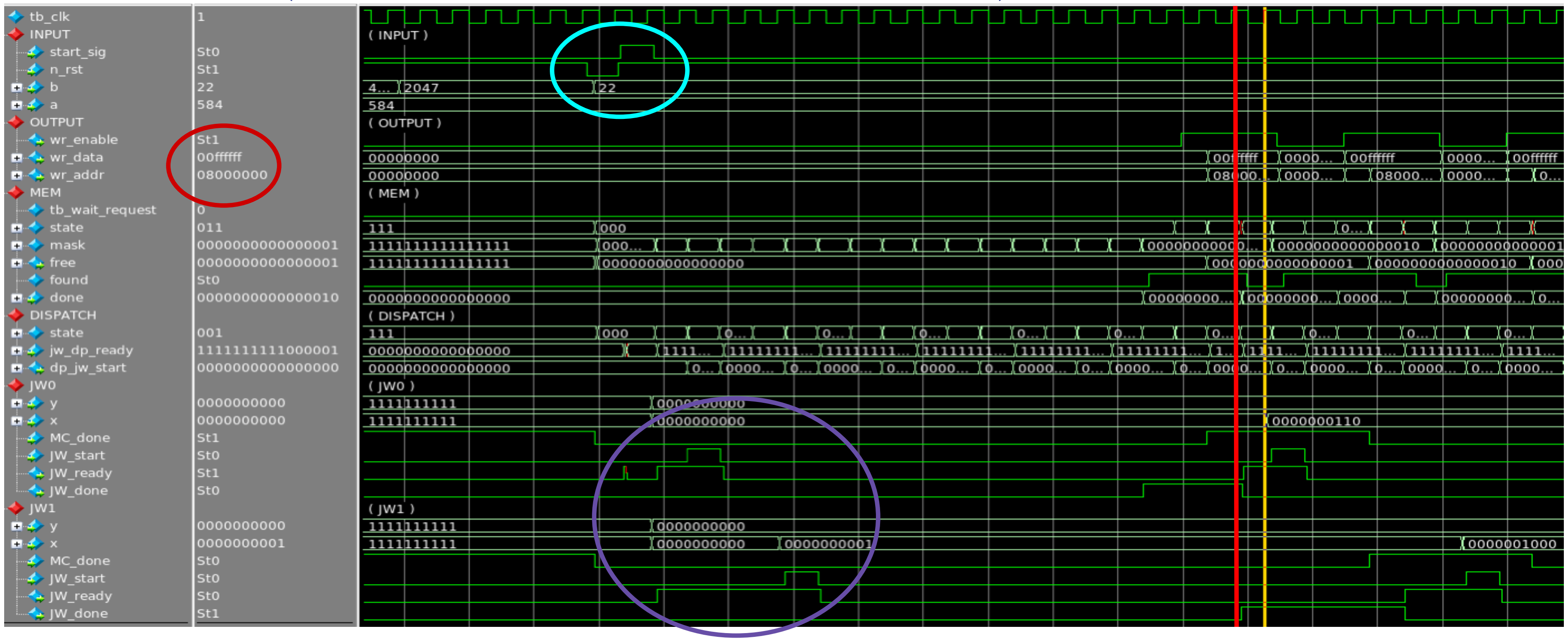


```
clk    0
n_rst  1
d      0001          1100
mask   0001          0001  00...  0100
wa     00000000      00000000            cda9879b
wd     00000000      00000000            abababab
we     0
wr     0
ca     dddbefafcda...  dddbefafcda9879bac10ffffaabecd98
cp     0aababababab...  0aababababababababcdefabcedf928302
found  1
```

Red line: Next done Julia worker found!

Blue: Pixel and address data muxed from concatenated line.

Results (custom_master_slave)

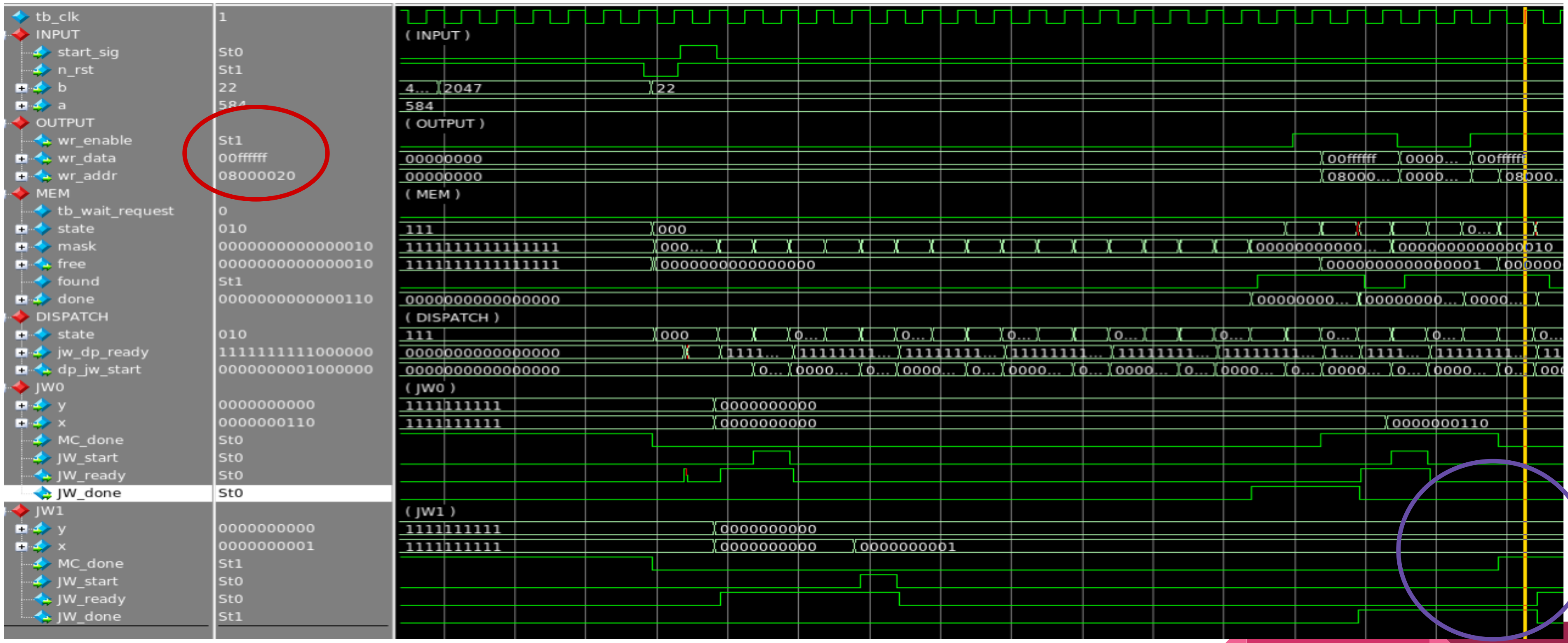


Blue circle: Custom logic reset followed by start signal

Purple circle: Julia Workers asking for task/assigned with task

Red circle: Output (Address = 08000000 , Color = 00FFFFFF)

Results (custom_master_slave)



```
tb_clk              1
INPUT
  start_sig         St0
  n_rst             St1
  b                 22
  a                 584
OUTPUT
  wr_enable         St1
  wr_data           00ffffff
  wr_addr           08000020
MEM
  tb_wait_request   0
  state             010
  mask              0000000000000010
  free              0000000000000010
  found             St1
  done              0000000000000110
DISPATCH
  state             010
  jw_dp_ready       1111111111000000
  dp_jw_start       0000000001000000
JW0
  y                 0000000000
  x                 0000000110
  MC_done           St0
  JW_start          St0
  JW_ready          St0
  JW_done           St0
JW1
  y                 0000000000
  x                 0000000001
  MC_done           St1
  JW_start          St0
  JW_ready          St0
  JW_done           St1
```

Purple circle: JW1 write to SDRAM after JW0 is done writing

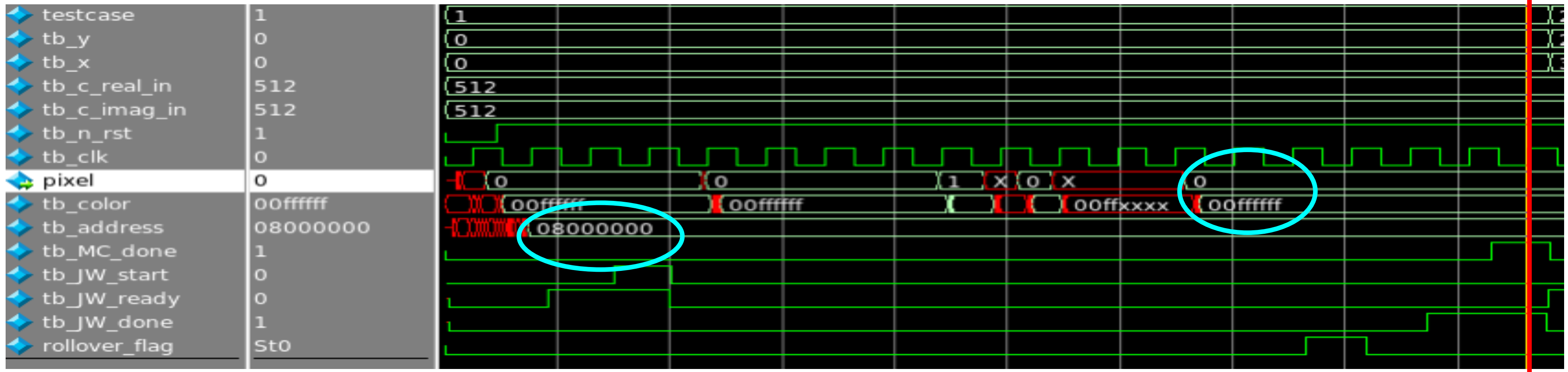
Red circle: Output (Address = **08000020** , Color = 00FFFFFF)

Conclusions

1. The biggest challenges in doing this design:
   a. FPGA integration
   b. Understanding how the Avalon bus system works.
   c. Writing data to SDRAM on the FPGA
2. How we would approach our design differently if we had to do it over again:
   a. Use a more efficient round robin approach
   b. More Julia Workers
3. What improvements we would make given more time:

Questions?

Results (julia worker) [BACKUP SLIDE] DSSC 1



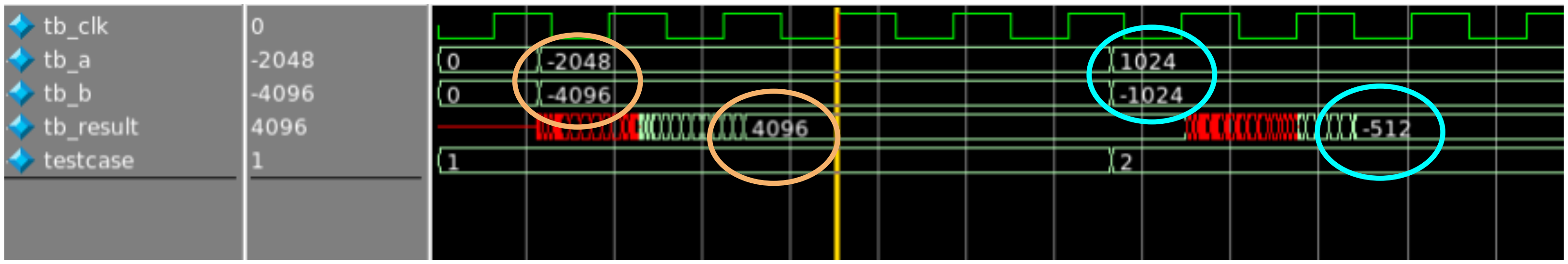
testcase	1
tb_y	0
tb_x	0
tb_c_real_in	512
tb_c_imag_in	512
tb_n_rst	1
tb_clk	0
pixel	0
tb_color	00ffffff
tb_address	08000000
tb_MC_done	1
tb_JW_start	0
tb_JW_ready	0
tb_JW_done	1
rollover_flag	St0

Red line: The positive clock edge where valid color and address are passed to Memory Controller.

Blue circle: Number of iterations = 0, Color = 00FFFFFF (white)

Address = 08000000 (x = 0, y = 0)

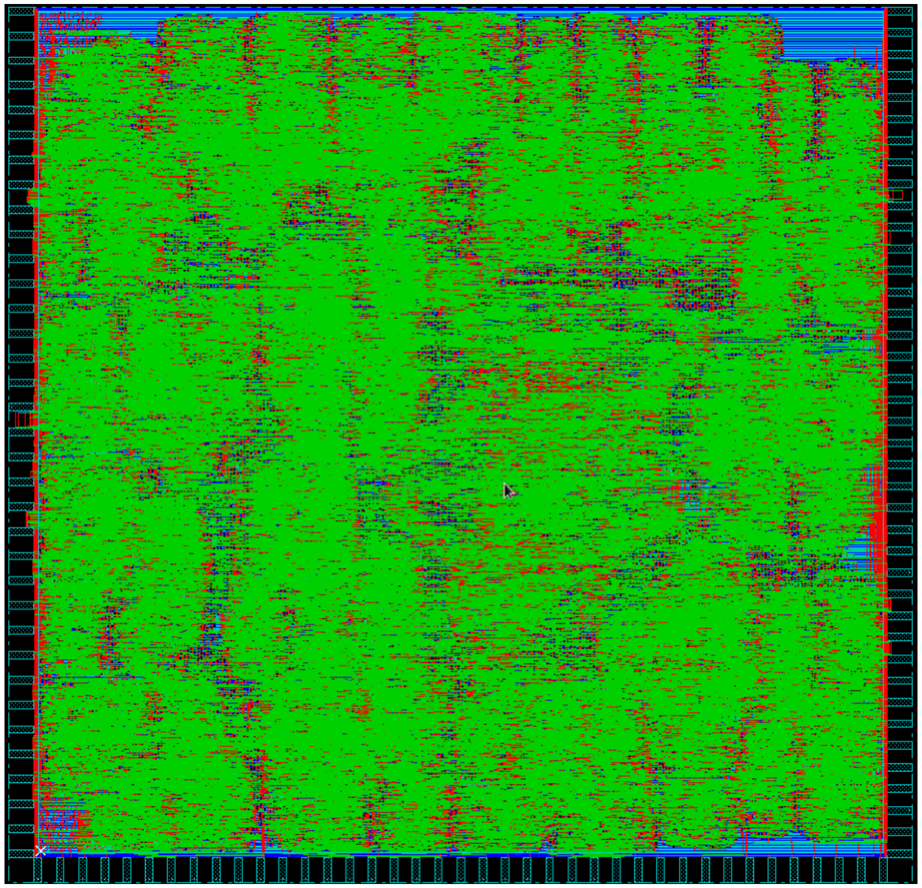
Results (Julia Worker)[BACKUP SLIDE] DSSC 1



Fundamental block in each Julia Workers

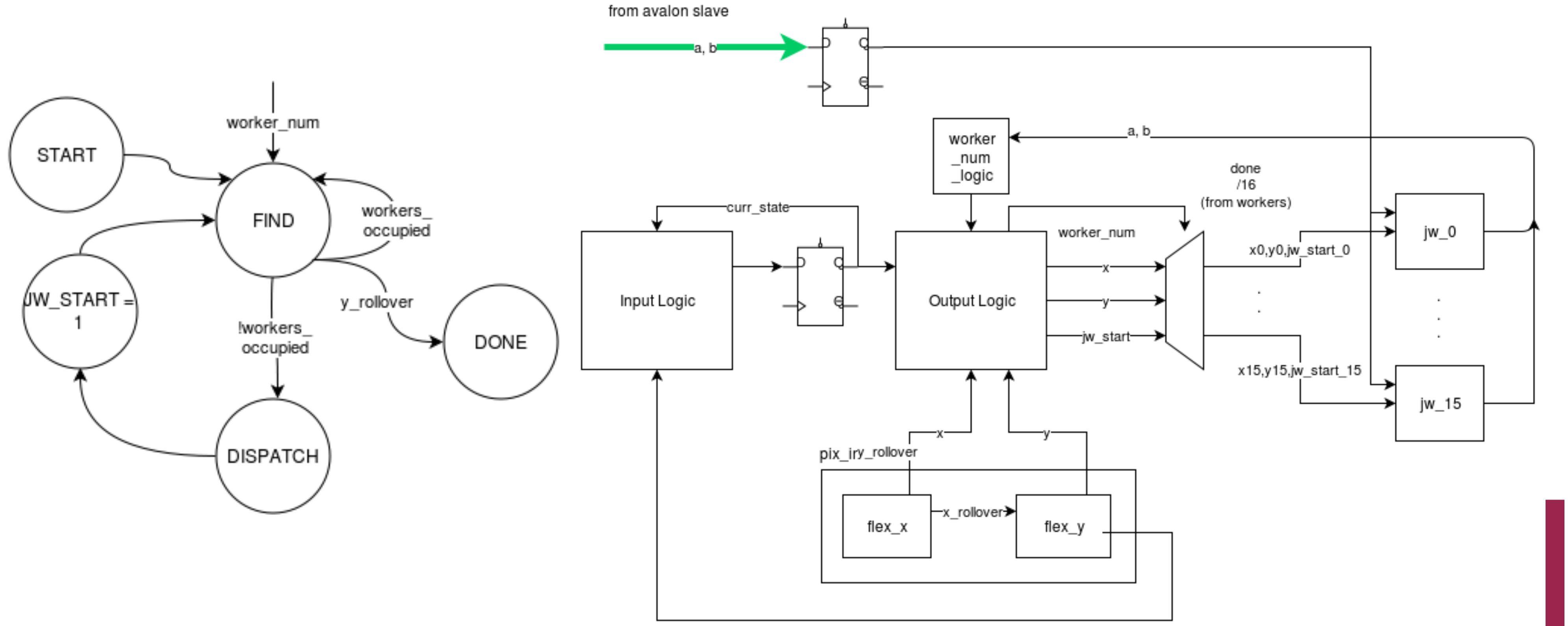
- Executes fixed point multiplication
- Julia Set algorithm: Znew = Zold^2 + Constant
- Metastable state will propagate to other part of the design

Layout

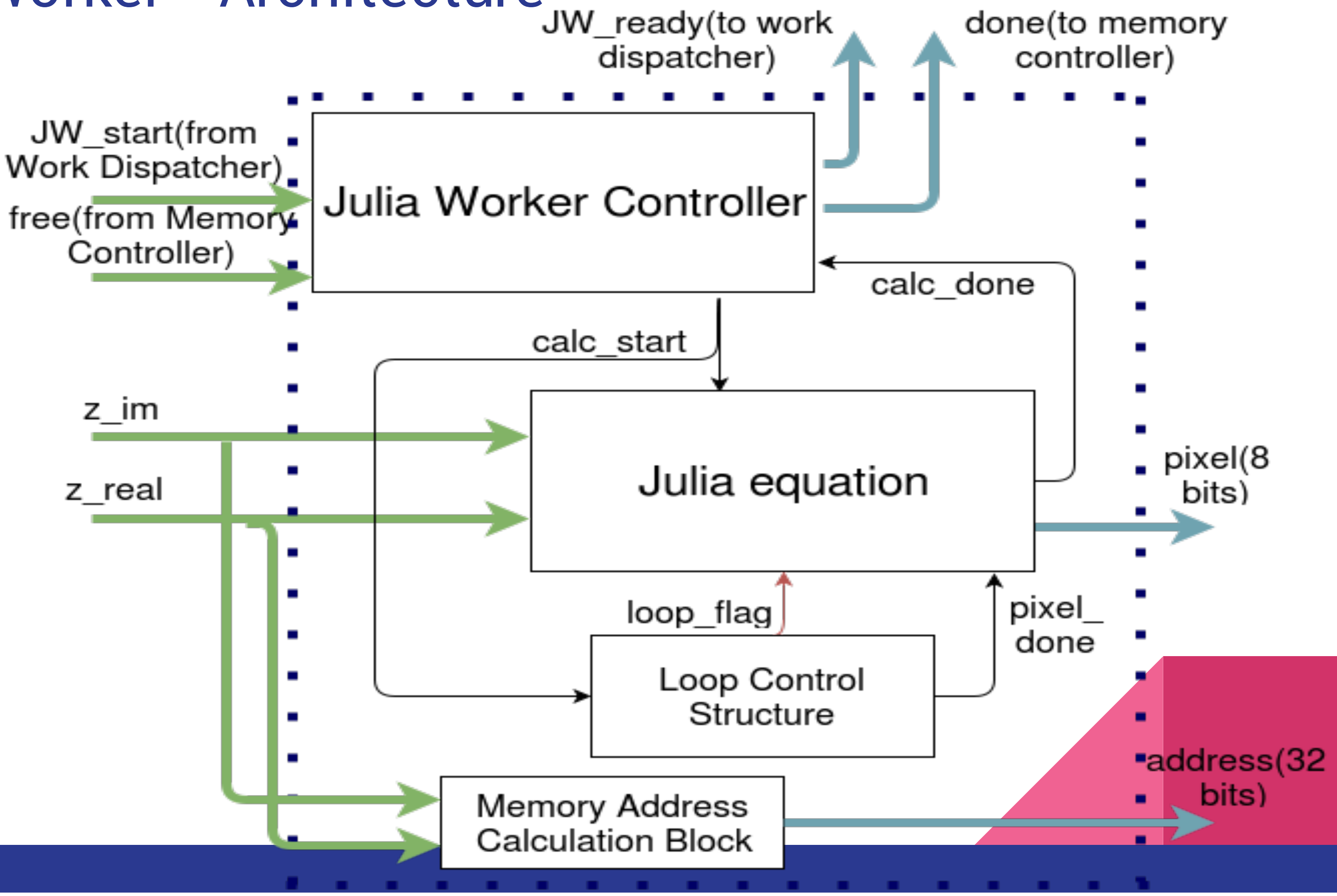


Synthesis Critical Path Delay : 3.5ns
(from Memory Controller state register
to master address)

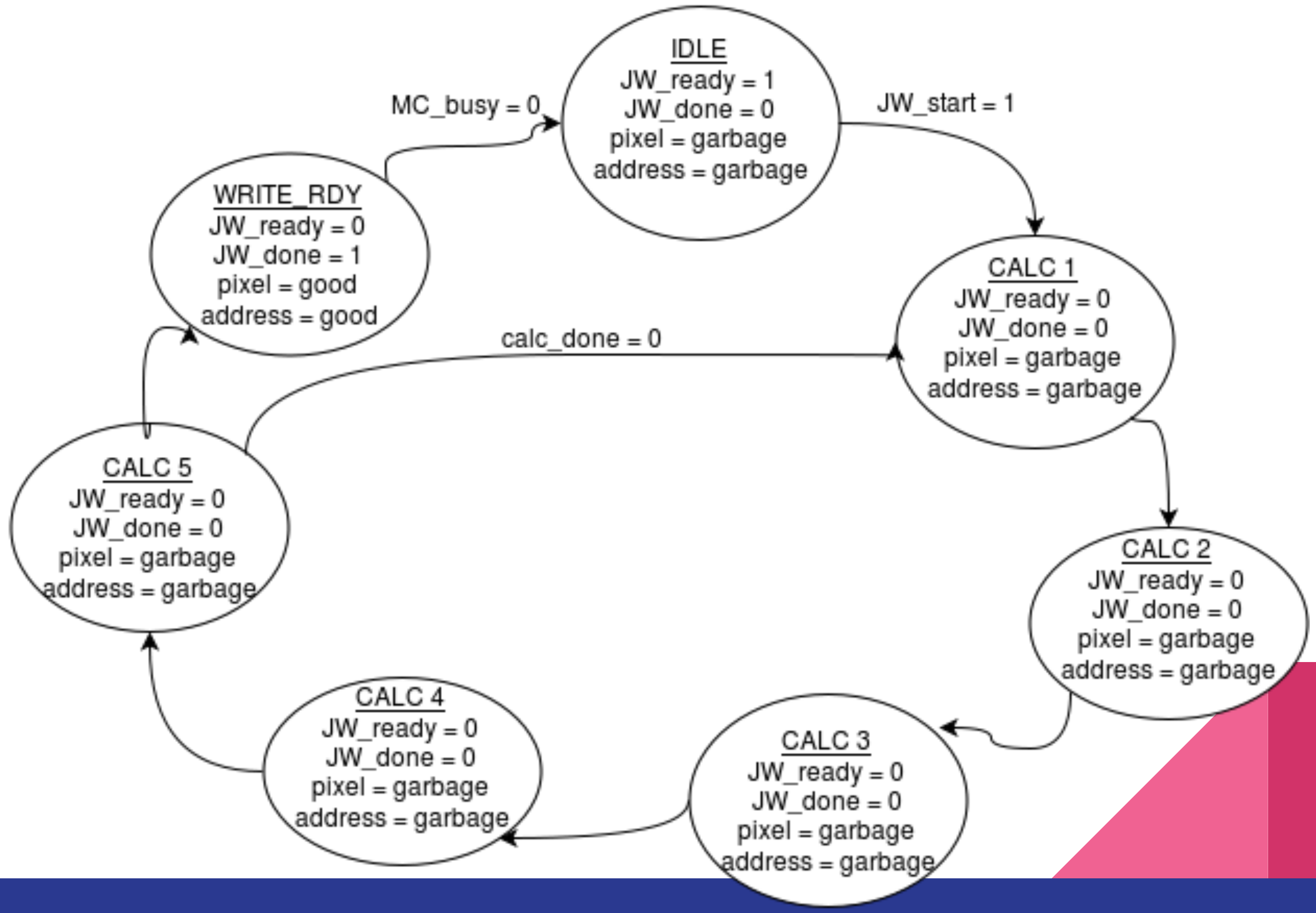
Work Dispatcher - Architecture and Logic



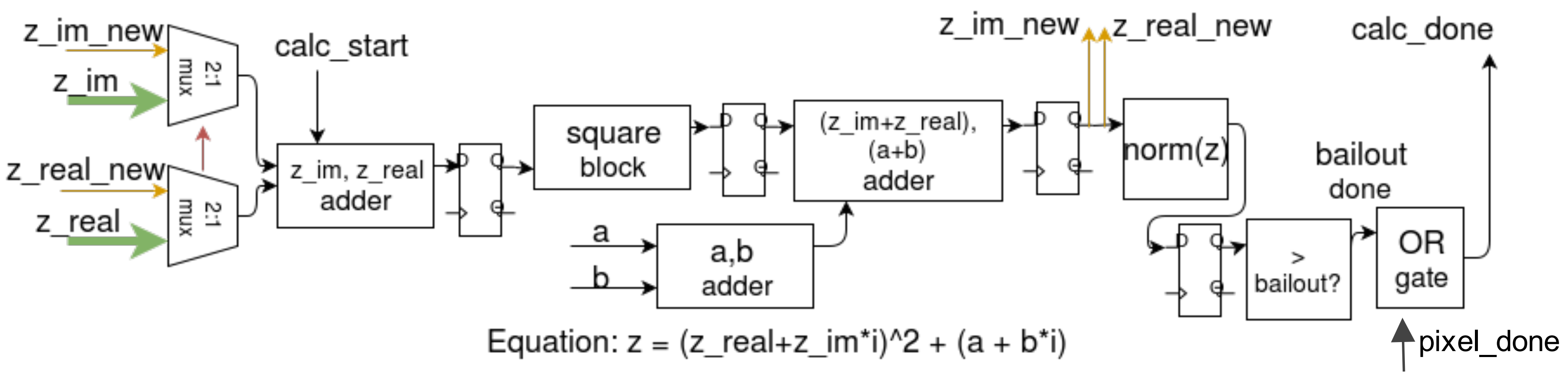
Julia Worker - Architecture


JW_ready(to work dispatcher)
done(to memory controller)
JW_start(from Work Dispatcher)
free(from Memory Controller)
Julia Worker Controller
calc_done
calc_start
z_im
z_real
Julia equation
pixel(8 bits)
loop_flag
pixel_ done
Loop Control Structure
address(32 bits)
Memory Address Calculation Block

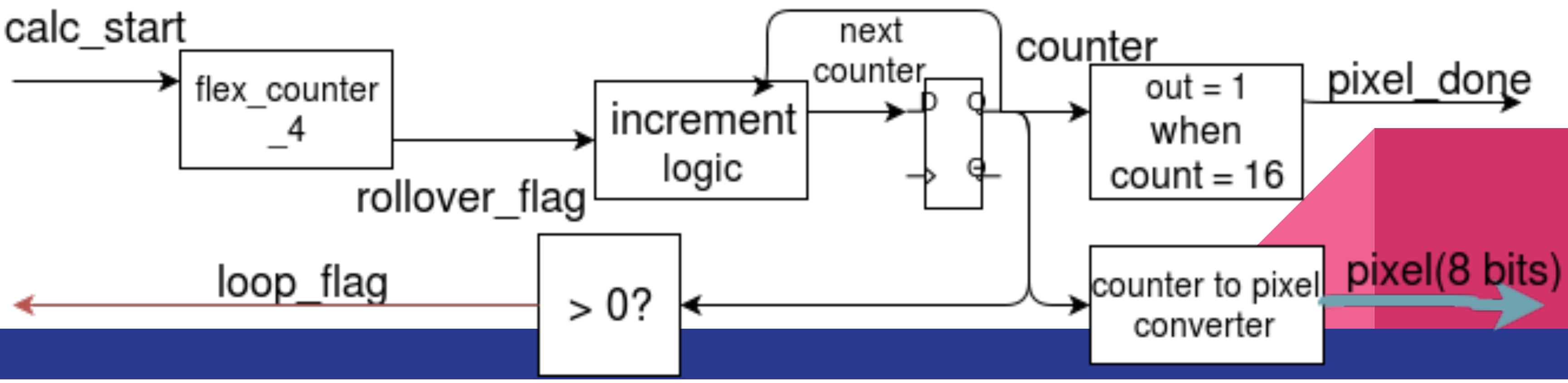
Julia Worker - Logic



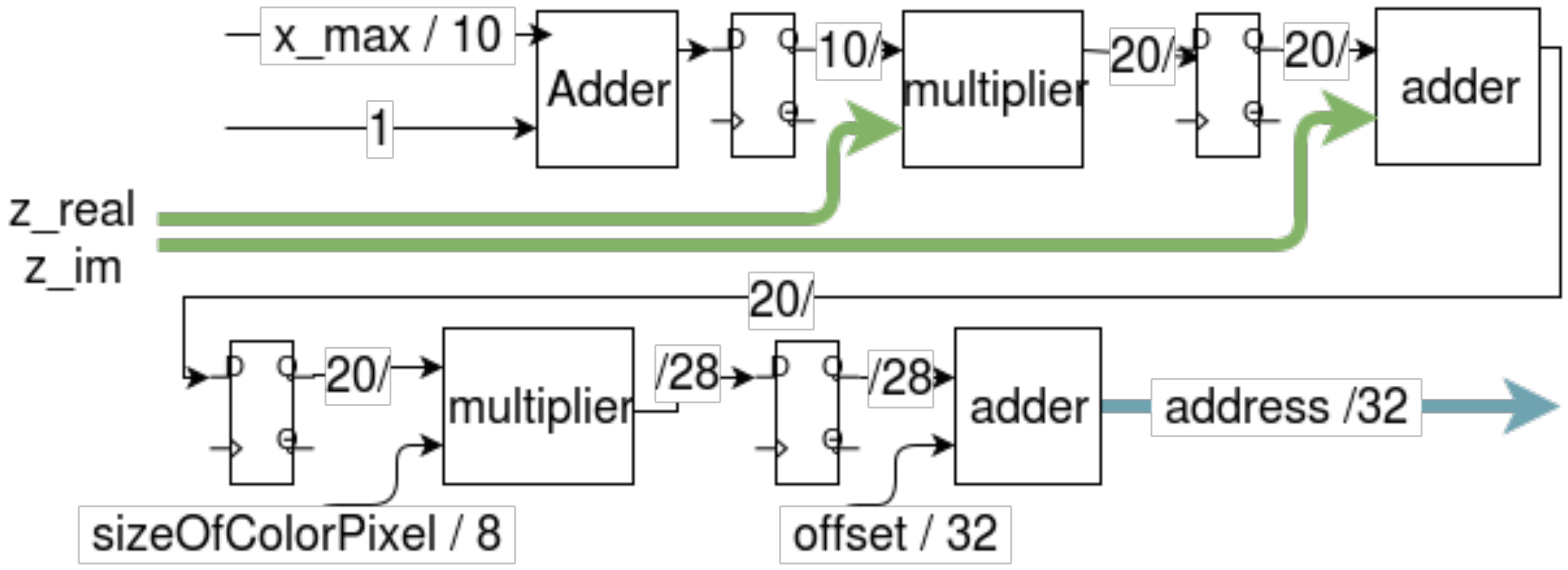
Julia Worker - Julia Equation



Julia Worker - Loop Control Structure

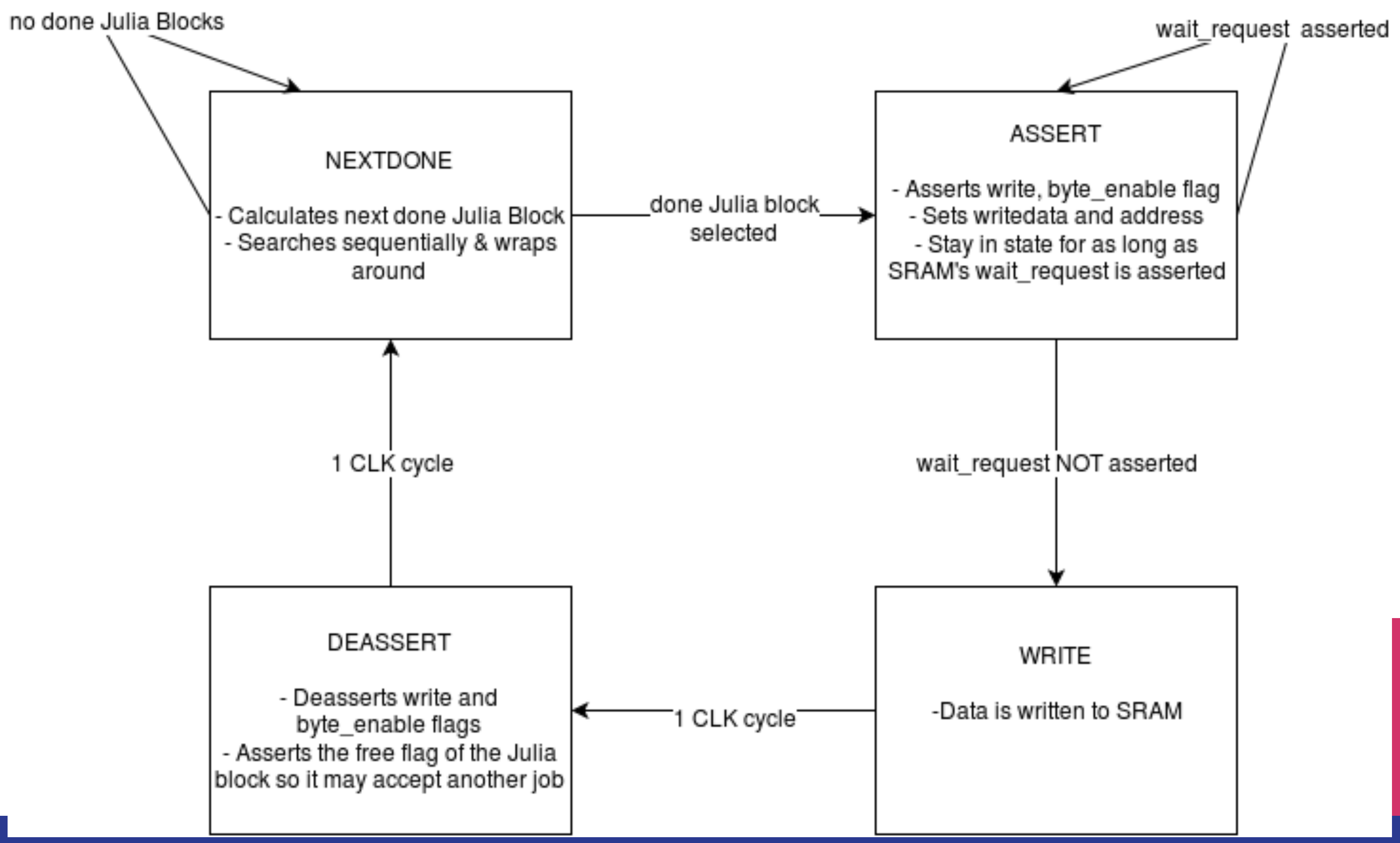


Julia Worker - Memory Address Calculation Block

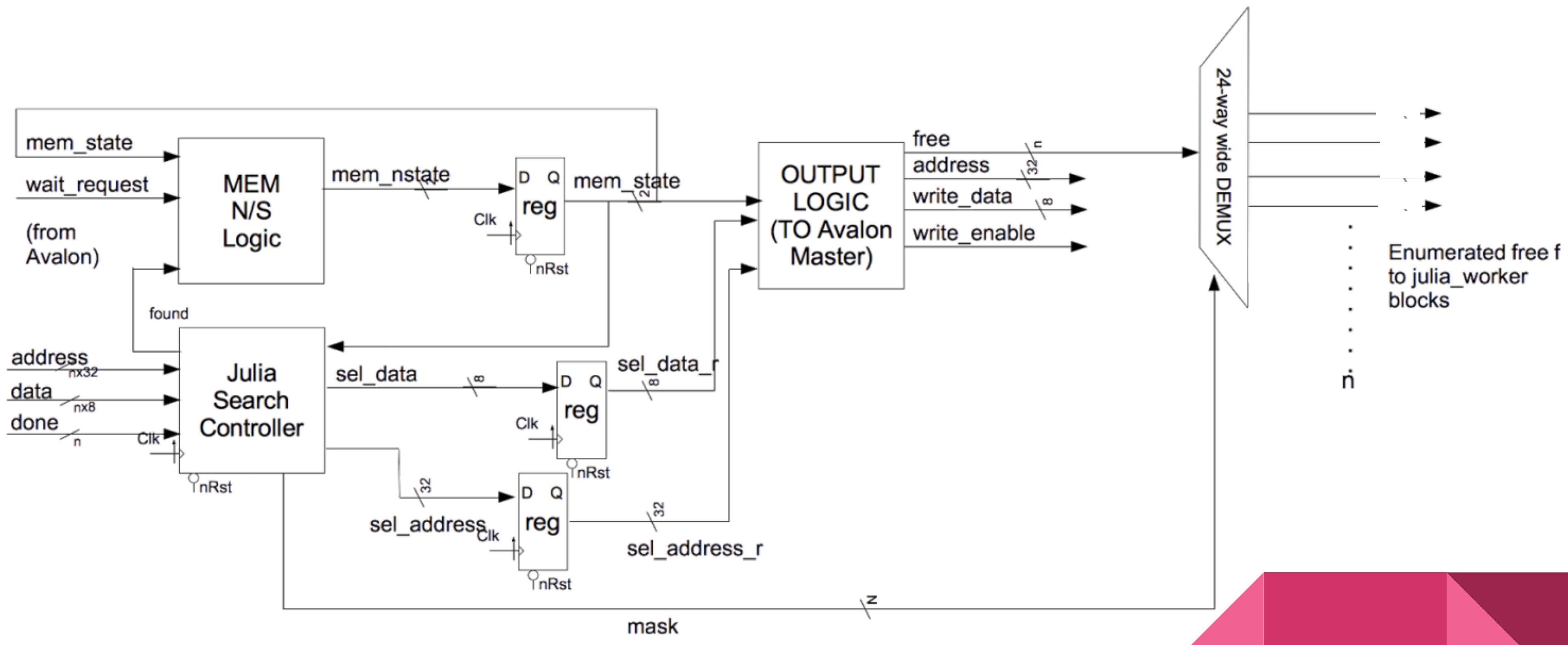


Equation: (sizeOfColorPixel * ((x_max + 1) * y) + x) + offset = address of start of pixel

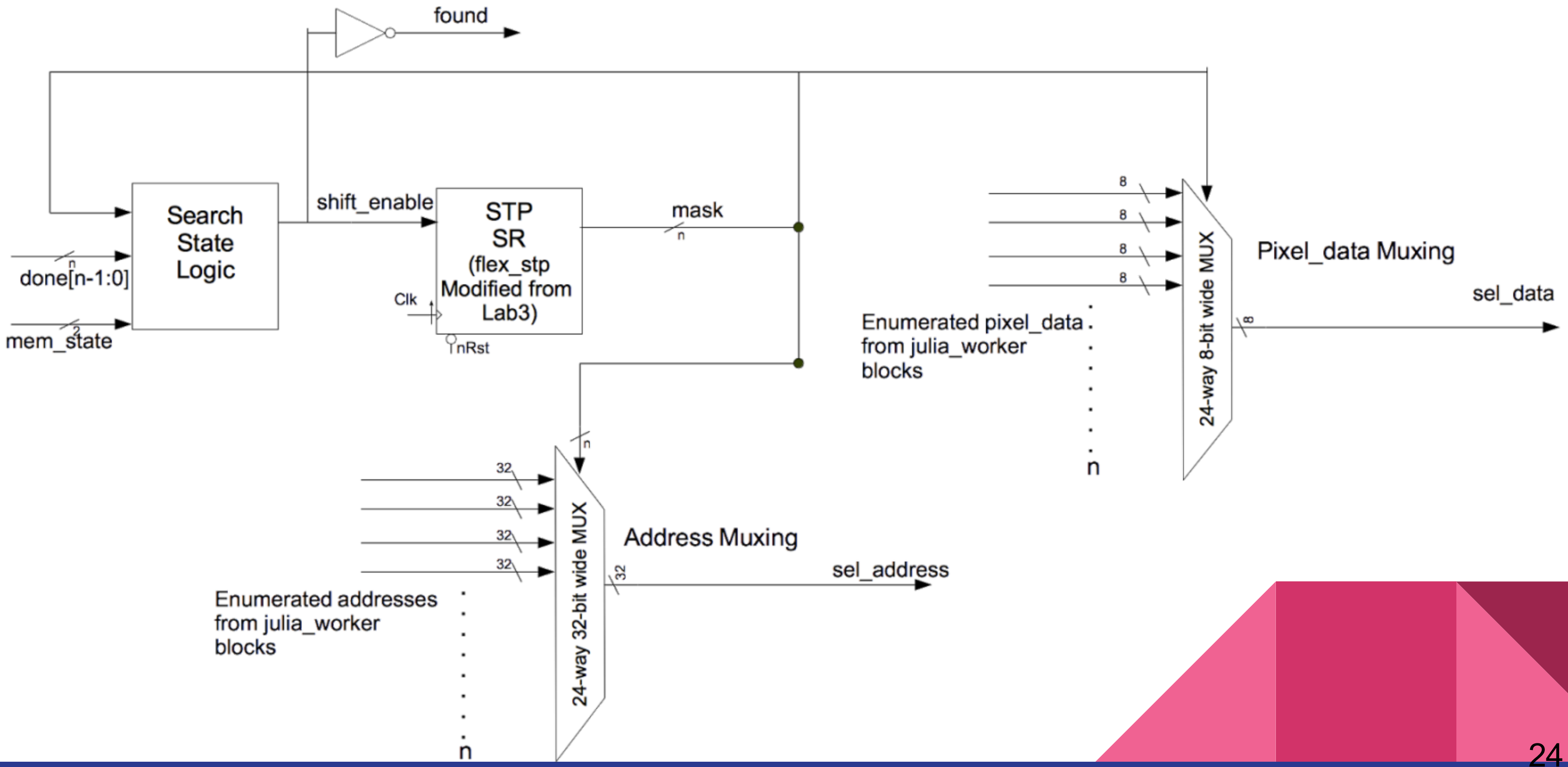
Memory Controller - Flow


no done Julia Blocks
NEXTDONE
- Calculates next done Julia Block
- Searches sequentially & wraps around
done Julia block selected
ASSERT
- Asserts write, byte_enable flag
- Sets writedata and address
- Stay in state for as long as SRAM's wait_request is asserted
wait_request asserted
wait_request NOT asserted
WRITE
-Data is written to SRAM
1 CLK cycle
DEASSERT
- Deasserts write and byte_enable flags
- Asserts the free flag of the Julia block so it may accept another job
1 CLK cycle

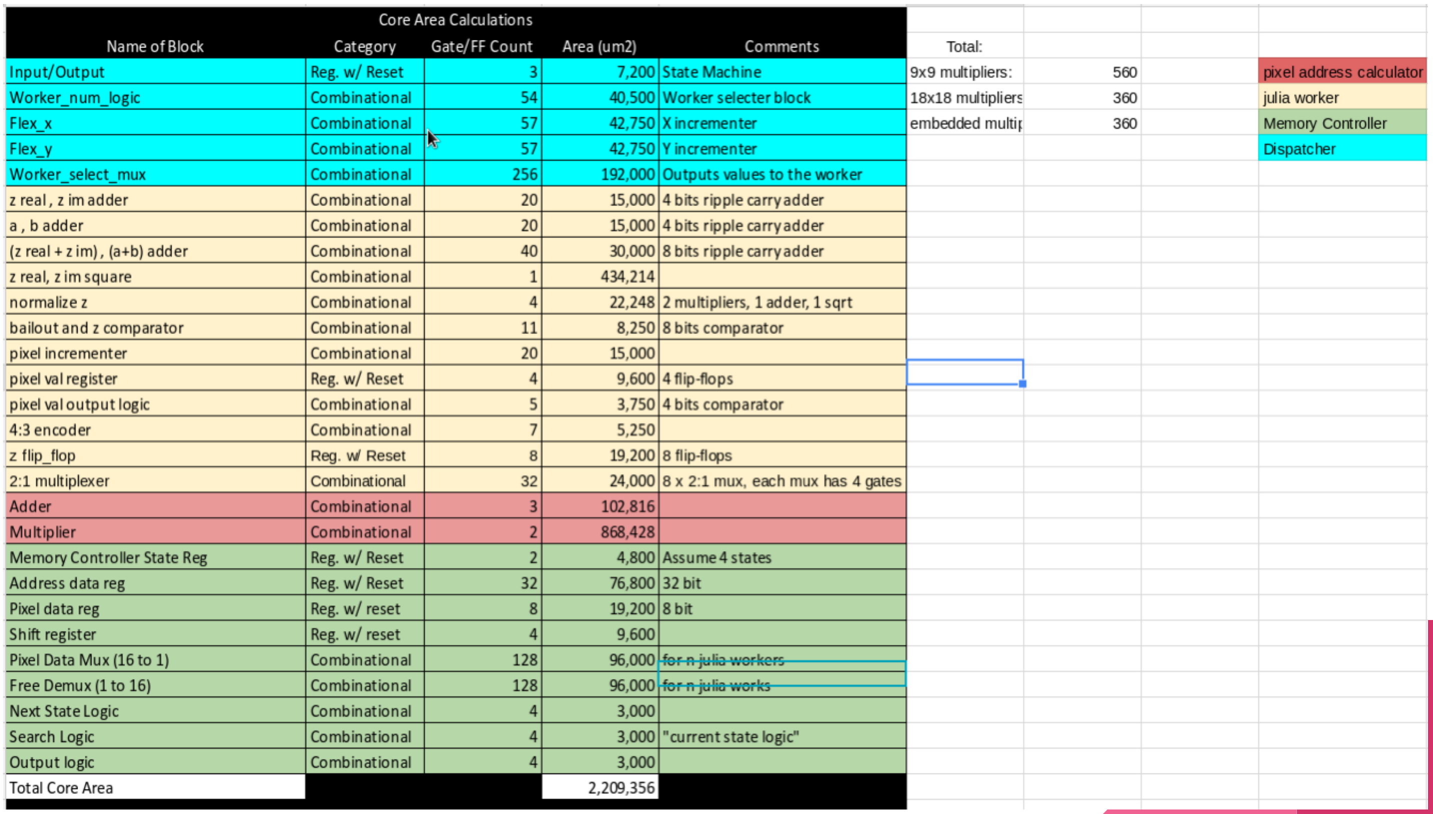
Memory Controller - Architecture



Memory Controller - Julia Search Controller Block



Area Budget



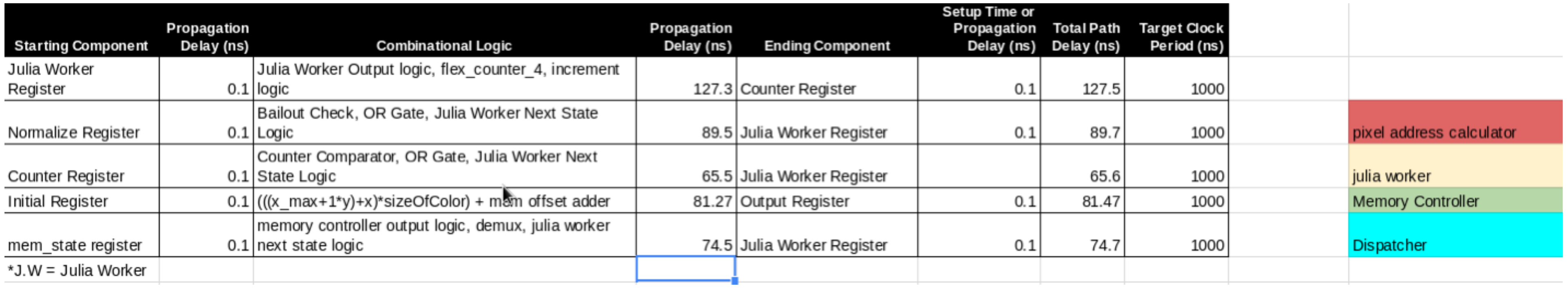
Core Area Calculations

Name of Block	Category	Gate/FF Count	Area (um2)	Comments
Input/Output	Reg. w/ Reset	3	7,200	State Machine
Worker_num_logic	Combinational	54	40,500	Worker selecter block
Flex_x	Combinational	57	42,750	X incrementer
Flex_y	Combinational	57	42,750	Y incrementer
Worker_select_mux	Combinational	256	192,000	Outputs values to the worker
z real , z im adder	Combinational	20	15,000	4 bits ripple carry adder
a , b adder	Combinational	20	15,000	4 bits ripple carry adder
(z real + z im) , (a+b) adder	Combinational	40	30,000	8 bits ripple carry adder
z real, z im square	Combinational	1	434,214	
normalize z	Combinational	4	22,248	2 multipliers, 1 adder, 1 sqrt
bailout and z comparator	Combinational	11	8,250	8 bits comparator
pixel incrementer	Combinational	20	15,000	
pixel val register	Reg. w/ Reset	4	9,600	4 flip-flops
pixel val output logic	Combinational	5	3,750	4 bits comparator
4:3 encoder	Combinational	7	5,250	
z flip_flop	Reg. w/ Reset	8	19,200	8 flip-flops
2:1 multiplexer	Combinational	32	24,000	8 x 2:1 mux, each mux has 4 gates
Adder	Combinational	3	102,816	
Multiplier	Combinational	2	868,428	
Memory Controller State Reg	Reg. w/ Reset	2	4,800	Assume 4 states
Address data reg	Reg. w/ Reset	32	76,800	32 bit
Pixel data reg	Reg. w/ reset	8	19,200	8 bit
Shift register	Reg. w/ reset	4	9,600	
Pixel Data Mux (16 to 1)	Combinational	128	96,000	for n julia workers
Free Demux (1 to 16)	Combinational	128	96,000	for n julia works
Next State Logic	Combinational	4	3,000	
Search Logic	Combinational	4	3,000	"current state logic"
Output logic	Combinational	4	3,000	
Total Core Area			2,209,356	

Total:	
9x9 multipliers:	560
18x18 multipliers	360
embedded multip	360

🟥	pixel address calculator
🟨	julia worker
🟩	Memory Controller
🟦	Dispatcher

Timing Budget



Starting Component	Propagation Delay (ns)	Combinational Logic	Propagation Delay (ns)	Ending Component	Setup Time or Propagation Delay (ns)	Total Path Delay (ns)	Target Clock Period (ns)		
Julia Worker Register	0.1	Julia Worker Output logic, flex_counter_4, increment logic	127.3	Counter Register	0.1	127.5	1000		
Normalize Register	0.1	Bailout Check, OR Gate, Julia Worker Next State Logic	89.5	Julia Worker Register	0.1	89.7	1000		pixel address calculator
Counter Register	0.1	Counter Comparator, OR Gate, Julia Worker Next State Logic	65.5	Julia Worker Register		65.6	1000		julia worker
Initial Register	0.1	(((x_max+1*y)+x)*sizeOfColor) + mem offset adder	81.27	Output Register	0.1	81.47	1000		Memory Controller
mem_state register	0.1	memory controller output logic, demux, julia worker next state logic	74.5	Julia Worker Register	0.1	74.7	1000		Dispatcher
*J.W = Julia Worker									

Data Sheet and Documentation

VGA chip data sheet

http://www.analog.com/media/en/technical-documentation/data-sheets/ADV7123.pdf

Avalon reference manual

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec.pdf

PCIe

https://sites.google.com/site/de2i150atpurdue/how-to-write-the-softw

https://sites.google.com/site/de2i150atpurdue/designing-your-ow

DE2i-150 FPGA System User Manual download link