Optical Mouse Scanner

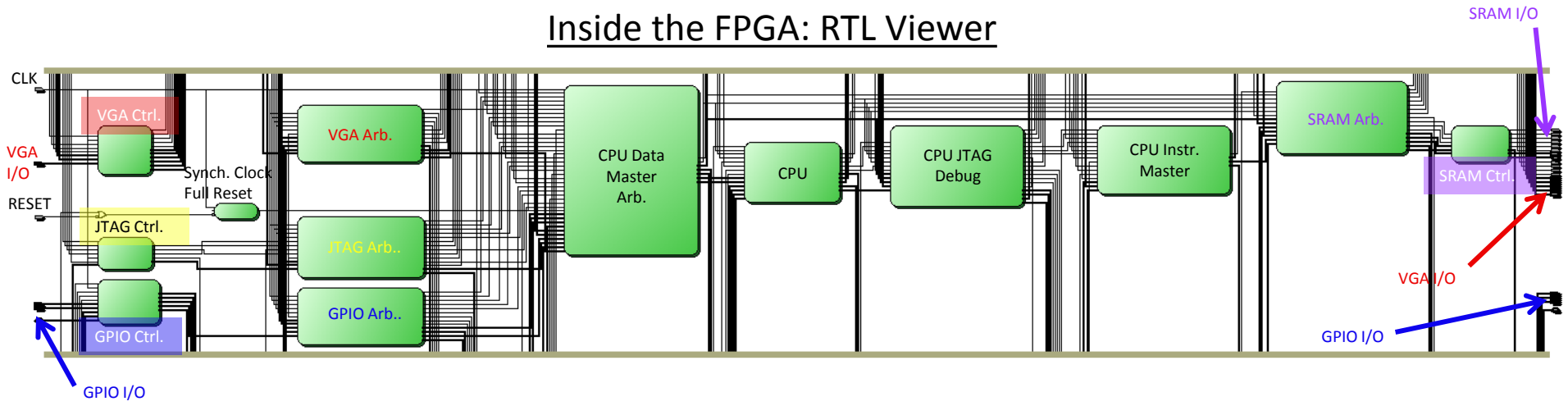
CSEE4840: Embedded Systems Design

David Calhoun
Kishore Padmaraju
Serge Yegiazarov

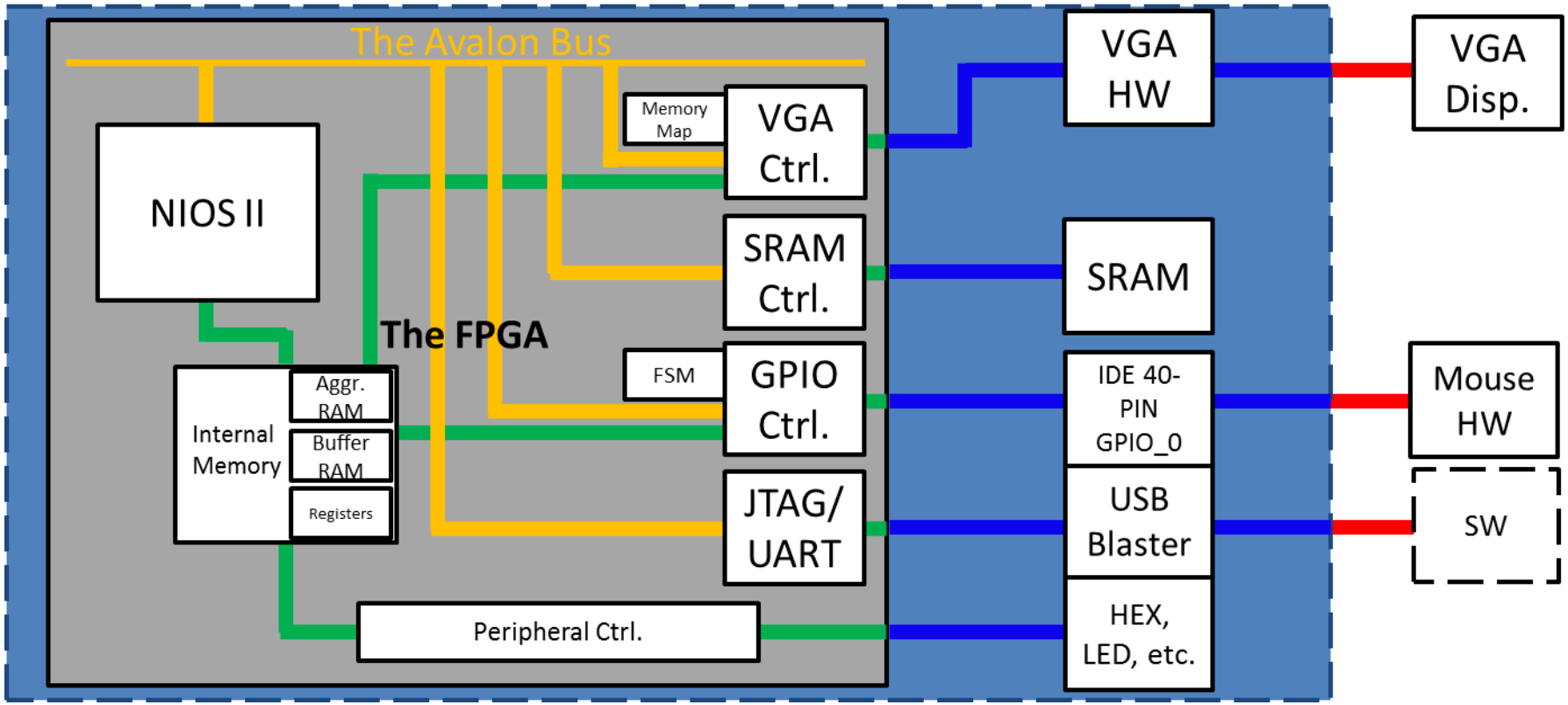
An Overview

- Our project was…
  - Relatively balanced in terms of SW/HW.
  - Relatively simple in terms of HW.
- Our project has become…
  - Much more HW than SW because of speed concerns.
  - More UI and visual aid-centric.
- Systems engineering played a key role, since both HW and SW are largely indispensable.
- Reverse engineering was heavily used, as the ADNS-2051 datasheet needed to be converted entirely into usable logic.

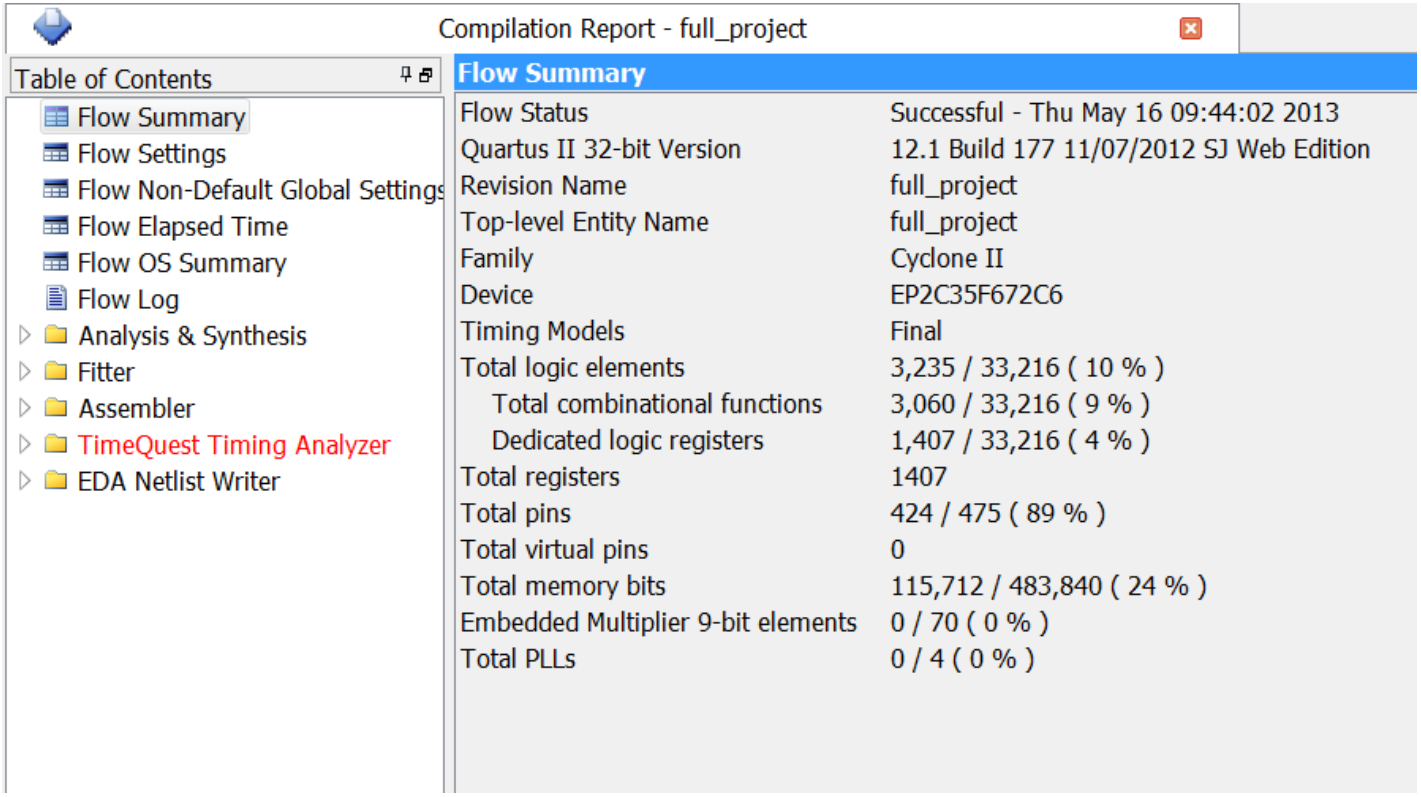
Inside the FPGA: RTL Viewer



The Architecture: As We Designed It

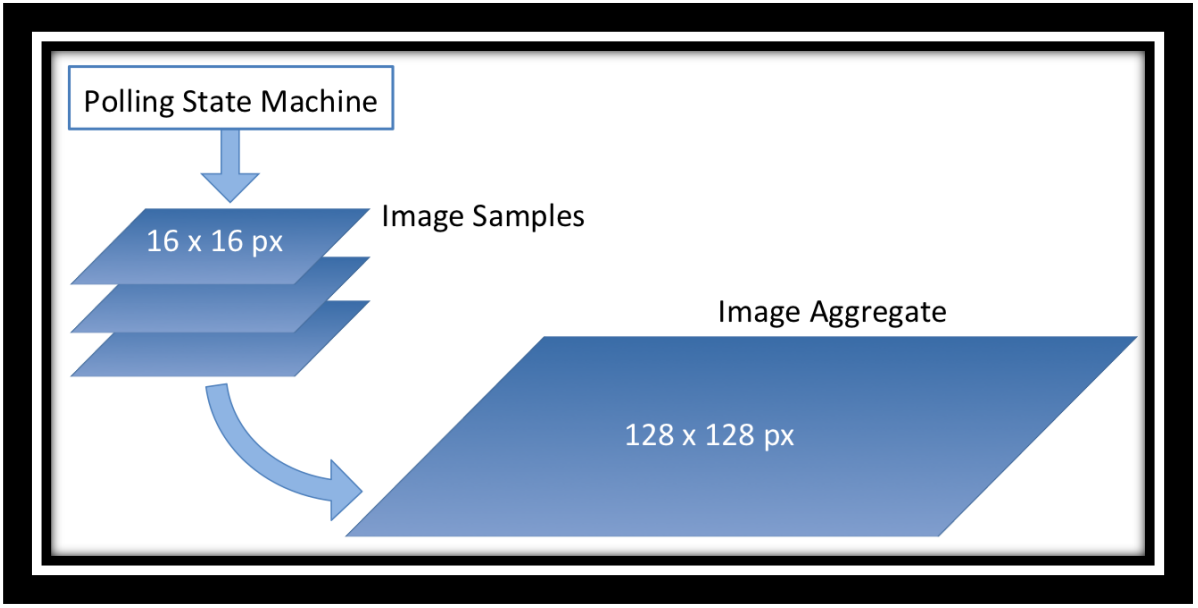


Compilation Report



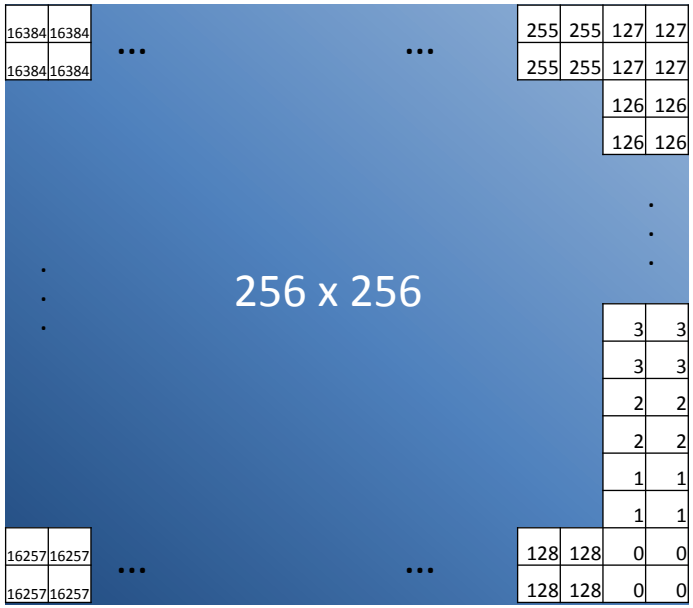
- Image storage and aggregation memory
  - Internal to FPGA using ALTSYNCRAM megafunction
  - Actual memory storage uses 128*128*6 + 16*16*6*4 = 104448 bits
  - Compilation reports 115,712 bits, which means some other bits were used for other peripheral registers

Image Acquisition and Aggregation

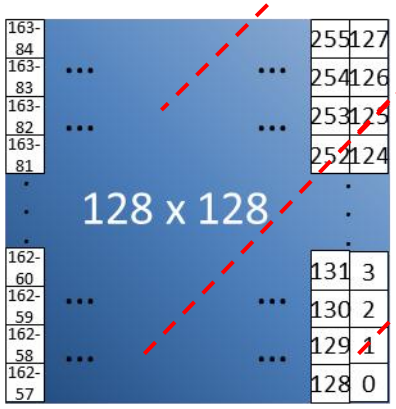


Polling State Machine

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255



Pixel Doubling



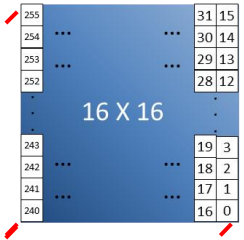
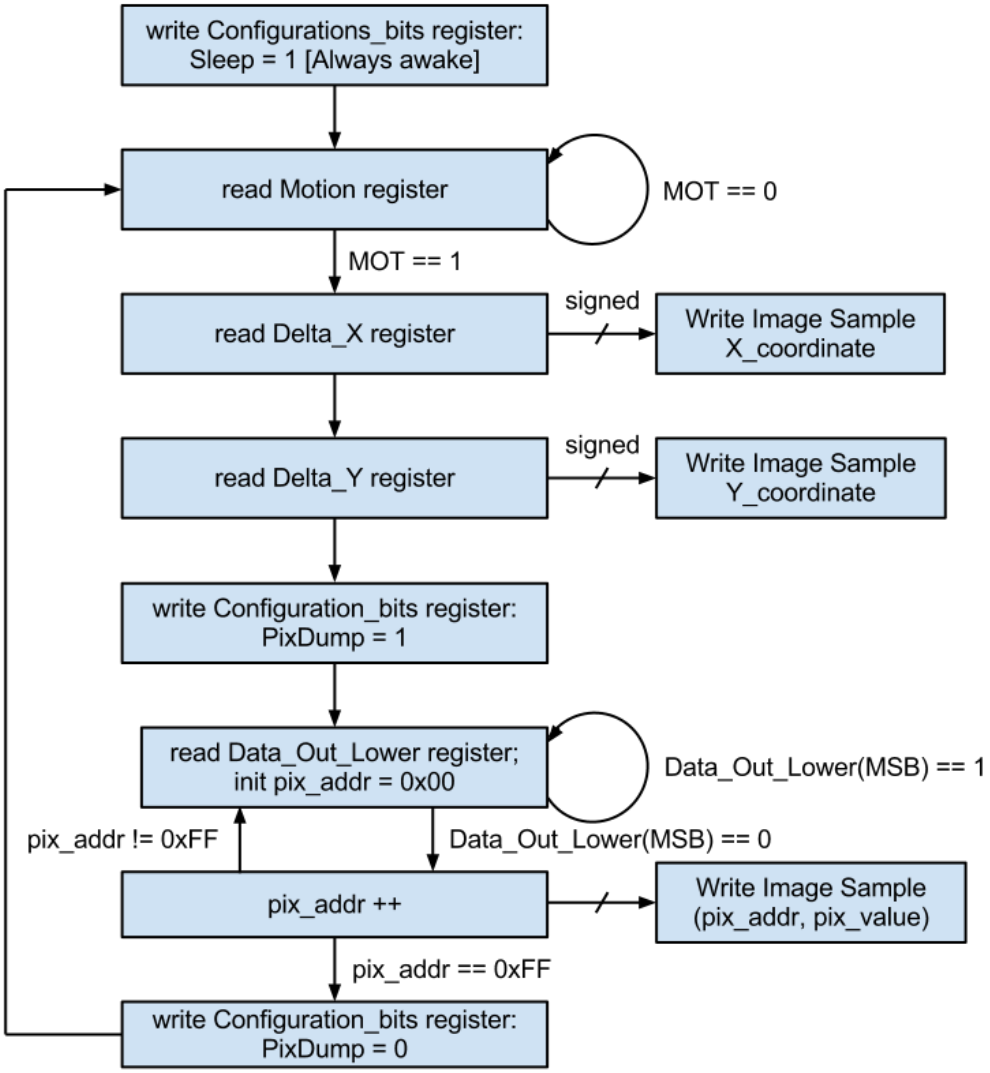


Image Index

Memory Mapping

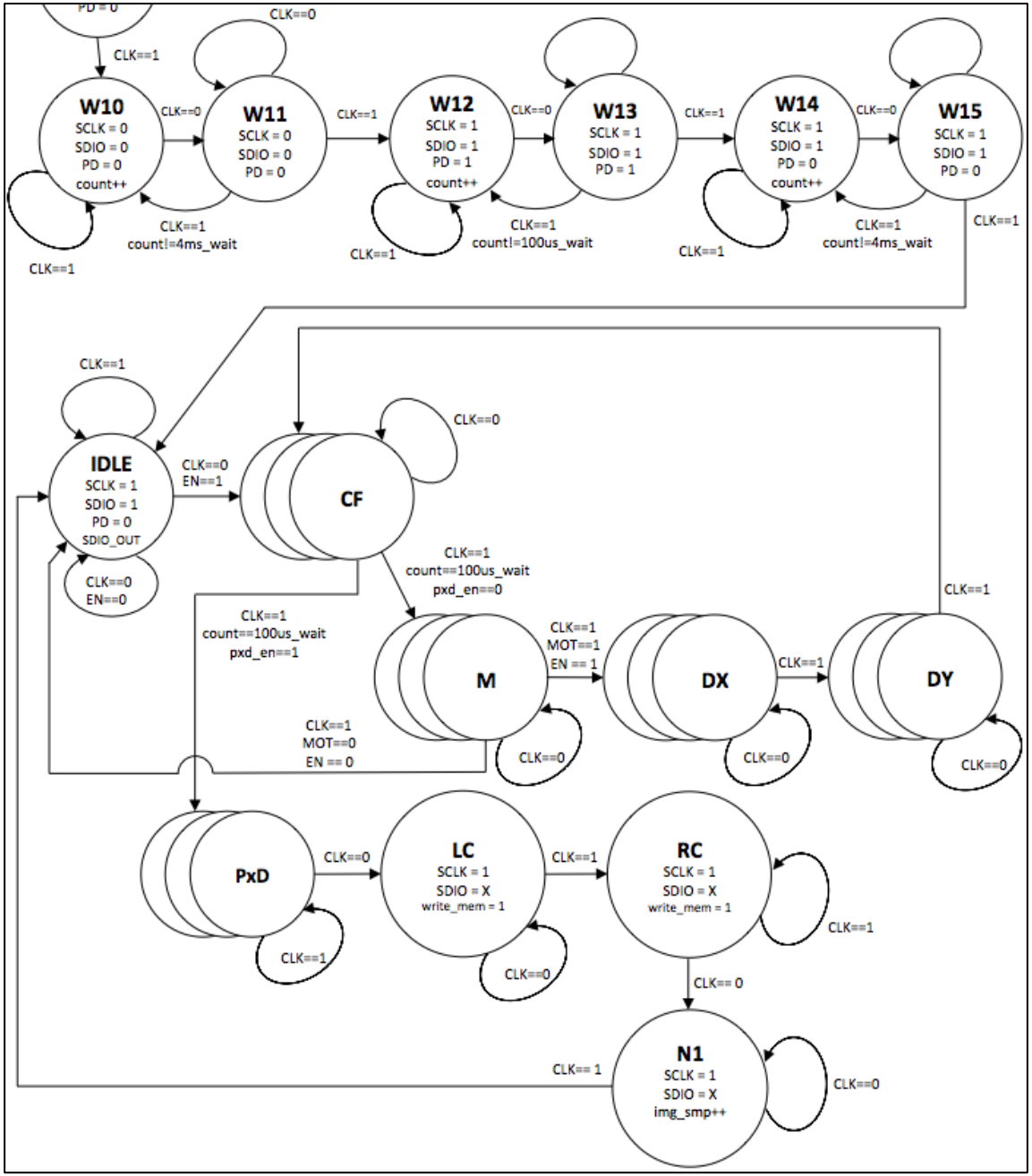


Software Implementation

- We had implemented this algorithmic version to test the optical processor
- Works, but very slow

Algorithmic Procedure

1. Polls for Mouse motion
2. Retrieve coordinates of movement
3. Retrieve image from mouse



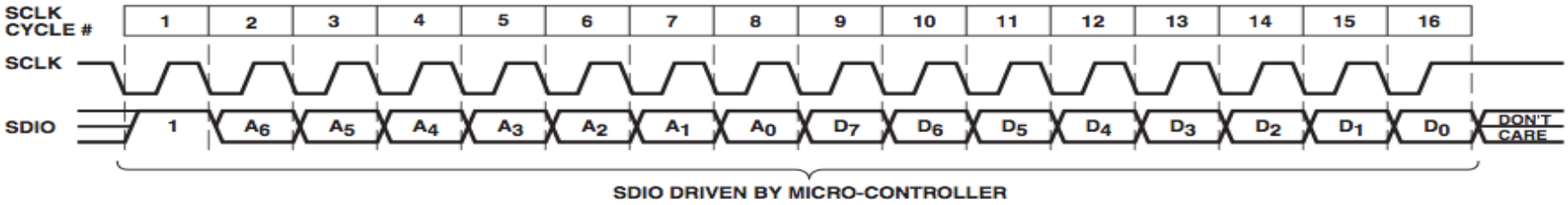
Finite State Machine

- Implemented as an FSM with 75 states
- Runs much faster than software implementation

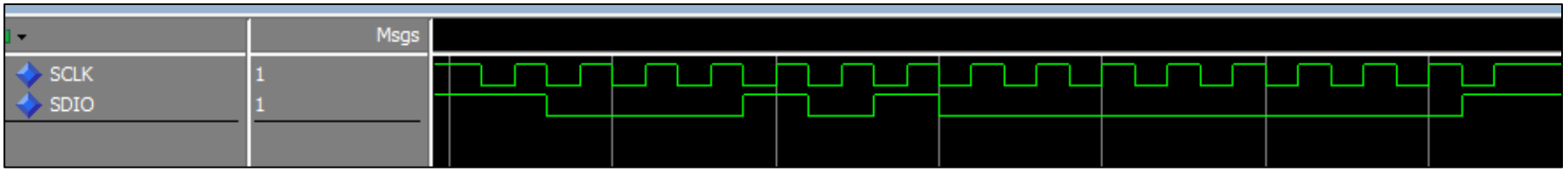
Mouse Optical Processor (ADNS-2051) interfaced using SPI protocol

- SCLK line always driven by FGPA
- Control of SDIO line toggled during read/write operations
- Additional PD (power-down) line was required to initialize and resync communication

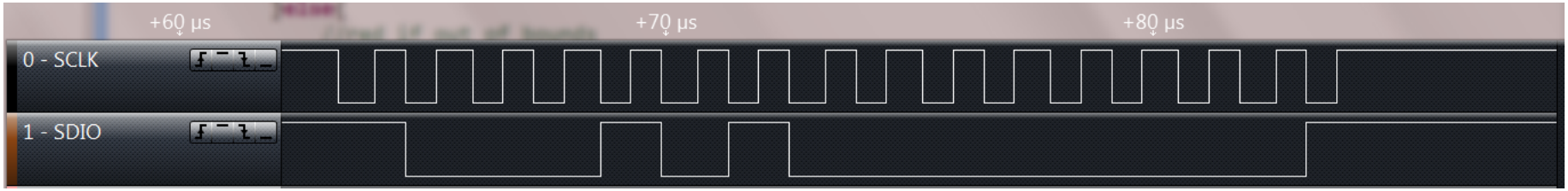
Datasheet Waveforms



Simulated Waveforms



Captured Waveforms using Logic Analyzer



Software

- The software performs/assists with the following tasks:
  - Coordinates the left/right click functionality.
  - Ensures new samples are unique.
  - Performs aggregation by telling hardware where to write the next samples.
  - Coordinates the location/color of highlight box.
  - Checks various boundary conditions and allows for/tracks out of bounds traversal.

Ensuring Uniqueness

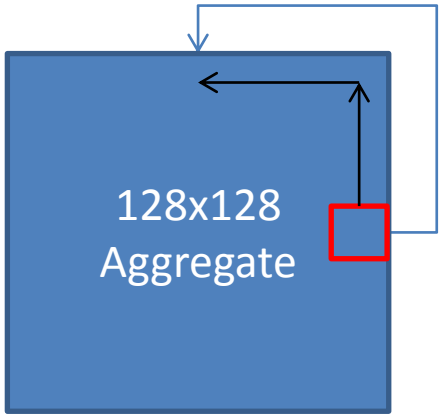
- Read 16 bit “select number” from hardware on every loop iteration.
- If this number is equivalent to the last such number, ignore and continue to next iteration.
- If it’s not equivalent, figure out which portion of the value differs, and write that to the hardware’s “read select”.
- This value determines the next sample.

Aggregation

- Since deltaX and deltaY are relative movement coordinates, software needs to keep track of absolute coordinates.
- Reads deltaX and deltaY, adds them to global position, checks boundaries, and writes back.
- The value written back is normalized to the following form in order to map to RAM: *ycoordinate+(xcoordinate*128)*

Boundary Checking

- Firstly, checks when user is about to leave boundaries and warns with red box.
- Secondly, allows out of bounds traversal.
- Thirdly, tracks the out of bounds movement by moving red box along edge.
- Prevents strange bugs (such as splitting and syncing) with some corner case handling.

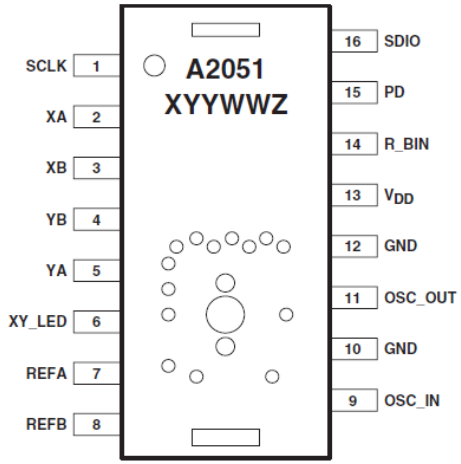


Experiences

- Power of the ADNS-2051
  - Or lack thereof…
  - dx and dy are calculated based on an image gradient, but they are also rounded arbitrarily
    - Consider dx of 0.625 => 1
      - This skews the image, although it provides sensitivity for mouse movement
  - Image blurring adds skew
    - Quick movements are not supported

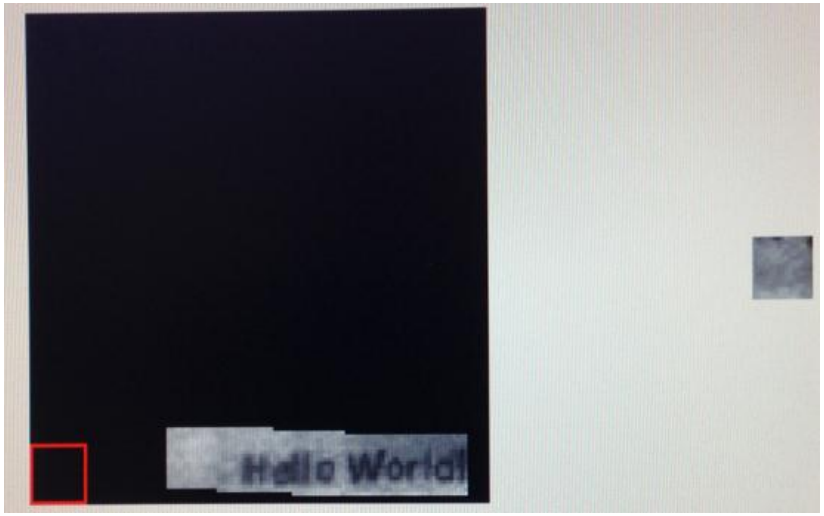
Slow Movement

Fast Movement



ADNS-2051 Optical Processor, Agilent/Avago Technologies

- Hardware interfacing is simple using the DE2
  - Several ways to approach this project
    - Could have created our own microprocessor core
      - Set up digital I/O pins (GPIO) with buffers, multiplexers, etc. for communicating with the mouse
      - Same memory on FPGA still required
      - This would enable a “fully software-defined” implementation
- Timing diagrams are a good aid, however…
  - They do not always reflect what will happen in real time
  - Simulation vs. synthesizable
  - Heed the warnings given by Quartus II
    - Jitter
    - Latches
    - Timing concerns, etc.



Issues Experienced

- Timing and synchronization
  - Image acquisition and software control are difficult to synchronize
    - Need to remove bottlenecks in software to get smooth acquisition and aggregation
  - It's hard to determine the response time of software with respect to our clock speed on the FPGA (our queue system helped resolve any issues we would face from this issue)
- State machine
  - Specification
    - Need to consider all conditions outlined in ADNS-2051 datasheet
      - Timing required between sending and receiving commands
      - Timing required between different types of commands
      - Layout of the state machine in an efficient way
      - Toggling Power-Down pin in order to reset and synchronize the serial communication
  - Timing
    - Data handling (outputs, changes) based on state changes vs. clock pulses
    - Simulations showed perfect behavior, actual communication generated by FPGA completely wrong [cannot trust simulation, had to use logic analyzer to verify what was going on]