6502

Akira Baruah
Chaiwen Chou
Phil Schiffrin
Sean Liu

Our Goals

- Initially set out to emulate the NES
- Implement the 6502 in SystemVerilog
- Synthesize the processor onto the FPGA
- Create software to interface with the processor
- Load programs into memory and read output of the processor in a user program

High Level design

CPU - Contains control signals, registers, and wires

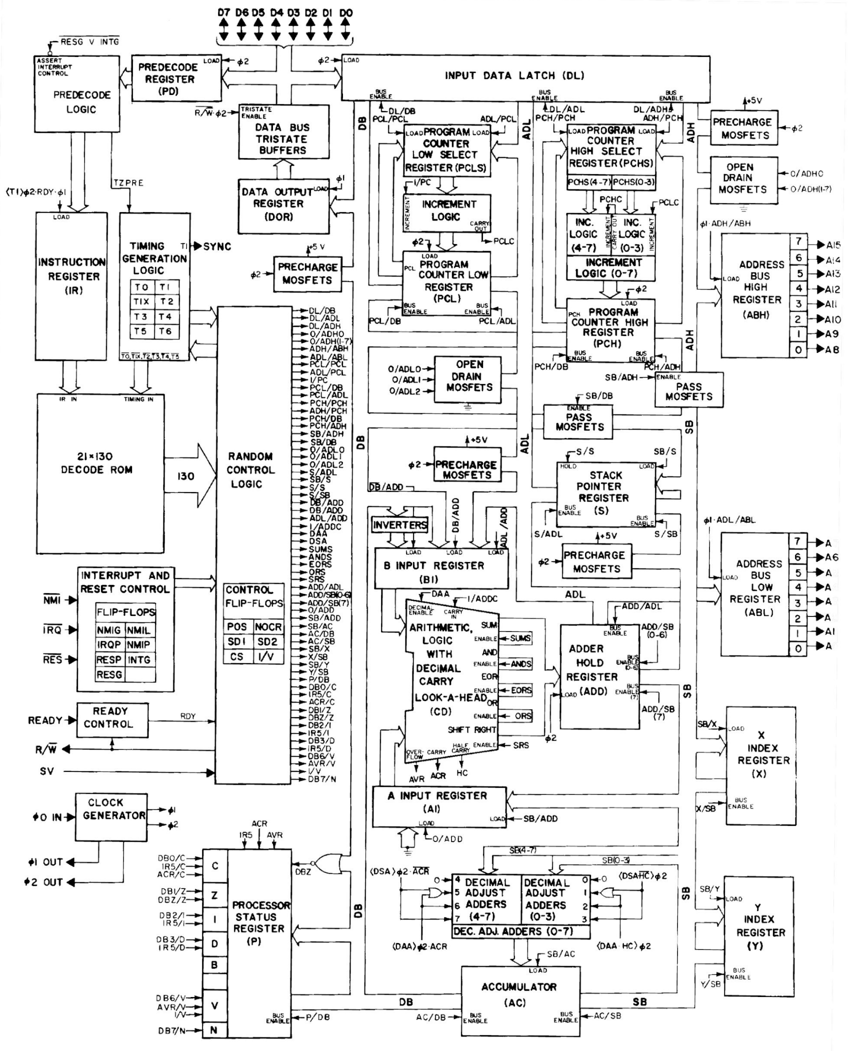
ALU - Computes all arithmetic operations for CPU

Memory - Basic read/write functionality

Original block diagram

Main changes:

- Single clock
- Control logic: Mealy finite state machine

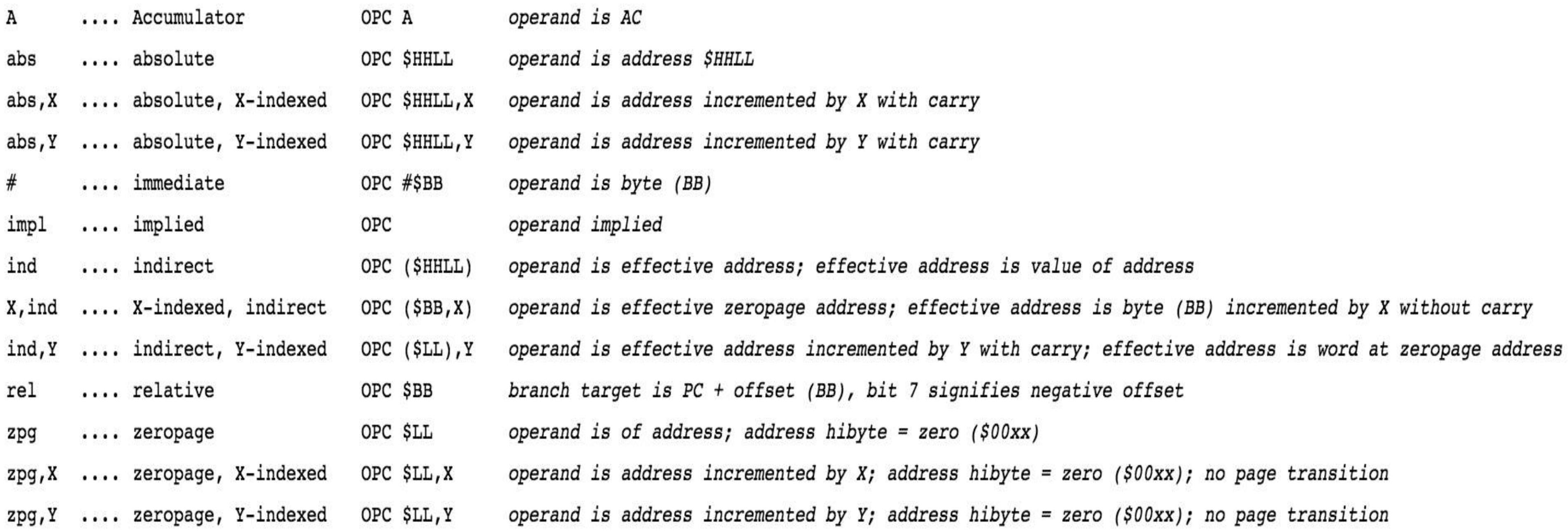

INPUT DATA LATCH (DL)
PREDECODE REGISTER (PD)
PREDECODE LOGIC
DATA BUS TRISTATE BUFFERS
DATA OUTPUT REGISTER (DOR)
INSTRUCTION REGISTER (IR)
TIMING GENERATION LOGIC
21×130 DECODE ROM
RANDOM CONTROL LOGIC
INTERRUPT AND RESET CONTROL
READY CONTROL
CLOCK GENERATOR
PROCESSOR STATUS REGISTER (P)
PROGRAM COUNTER LOW SELECT REGISTER (PCLS)
PROGRAM COUNTER HIGH SELECT REGISTER (PCHS)
PROGRAM COUNTER LOW REGISTER (PCL)
PROGRAM COUNTER HIGH REGISTER (PCH)
ADDRESS BUS HIGH REGISTER (ABH)
ADDRESS BUS LOW REGISTER (ABL)
STACK POINTER REGISTER (S)
B INPUT REGISTER (BI)
A INPUT REGISTER (AI)
ARITHMETIC, LOGIC WITH DECIMAL CARRY LOOK-A-HEAD (CD)
ADDER HOLD REGISTER (ADD)
DECIMAL ADJUST ADDERS
ACCUMULATOR (AC)
X INDEX REGISTER (X)
Y INDEX REGISTER (Y)
PRECHARGE MOSFETS
OPEN DRAIN MOSFETS
PASS MOSFETS
INVERTERS

Architecture

Control signals

Combinational/sequential blocks

Addressing Modes

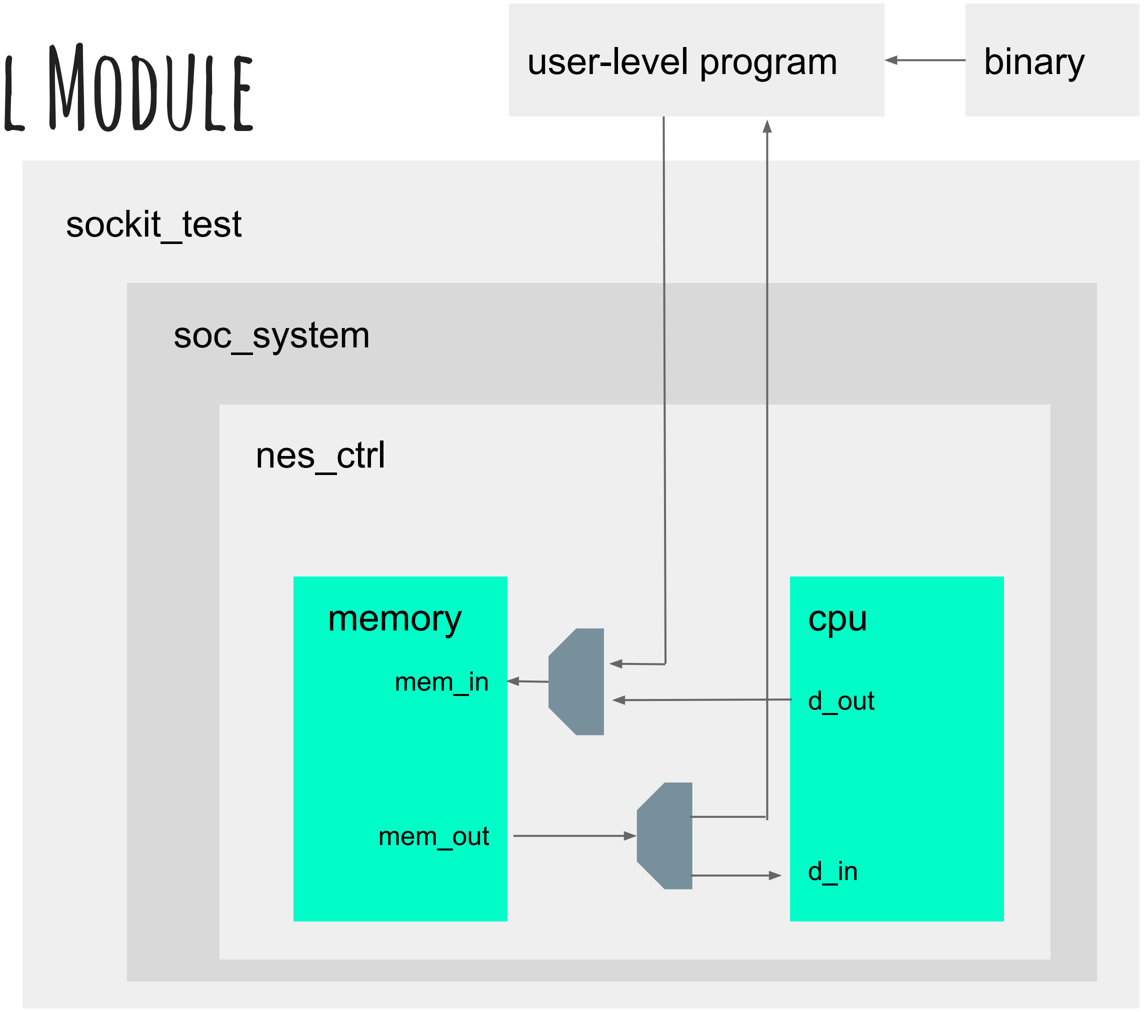


A	.... Accumulator	OPC A	*operand is AC*
abs	.... absolute	OPC $HHLL	*operand is address $HHLL*
abs,X	.... absolute, X-indexed	OPC $HHLL,X	*operand is address incremented by X with carry*
abs,Y	.... absolute, Y-indexed	OPC $HHLL,Y	*operand is address incremented by Y with carry*
#	.... immediate	OPC #$BB	*operand is byte (BB)*
impl	.... implied	OPC	*operand implied*
ind	.... indirect	OPC ($HHLL)	*operand is effective address; effective address is value of address*
X,ind	.... X-indexed, indirect	OPC ($BB,X)	*operand is effective zeropage address; effective address is byte (BB) incremented by X without carry*
ind,Y	.... indirect, Y-indexed	OPC ($LL),Y	*operand is effective address incremented by Y with carry; effective address is word at zeropage address*
rel	.... relative	OPC $BB	*branch target is PC + offset (BB), bit 7 signifies negative offset*
zpg	.... zeropage	OPC $LL	*operand is of address; address hibyte = zero ($00xx)*
zpg,X	.... zeropage, X-indexed	OPC $LL,X	*operand is address incremented by X; address hibyte = zero ($00xx); no page transition*
zpg,Y	.... zeropage, Y-indexed	OPC $LL,Y	*operand is address incremented by Y; address hibyte = zero ($00xx); no page transition*

Absolute Addressing (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Effective Address
T2	PC + 2	ADH	1	Fetch high order byte of Effective Address
T3	ADH, ADL	Data	0	Write internal register to memory
T0	PC + 3	OP CODE	1	Next Instruction

Top-Level Module



Quartus, Qsys, and the Software Interface

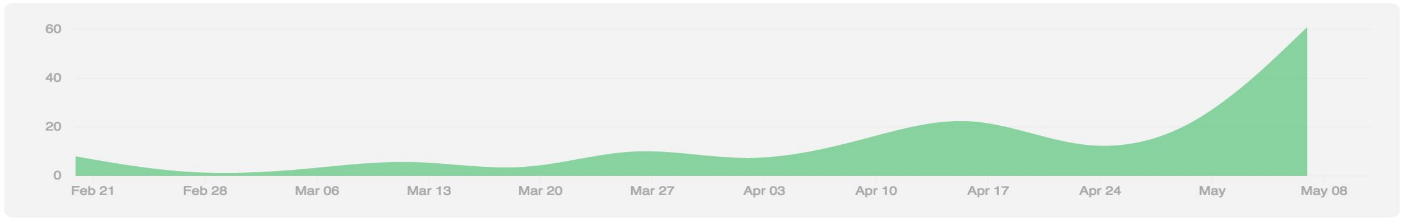
- Attempted kernel module for interfacing with hardware
- Hacky solution that worked for us: mmap to “/dev/mem”
- Created user-space program that writes into NES memory the contents of a binary file containing instructions for the processor
- 16 bits - top 8 bits for our own “opcodes”, bottom 8 for data

Work Flow

Feb 21, 2016 – May 12, 2016

Contributions to master, excluding merge commits

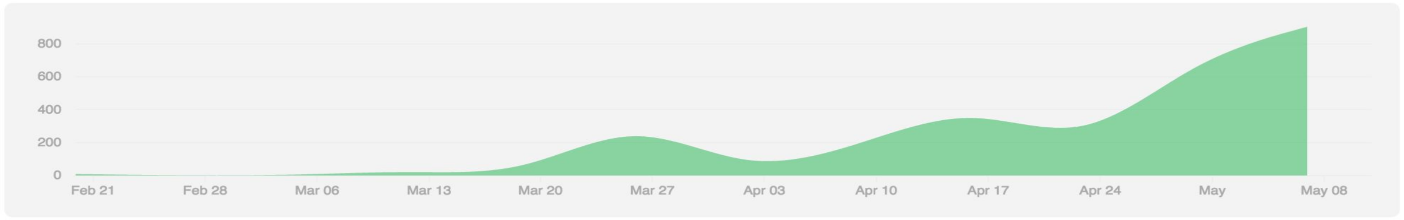
Contributions: **Commits** ▾



Feb 21, 2016 – May 12, 2016

Contributions to master, excluding merge commits

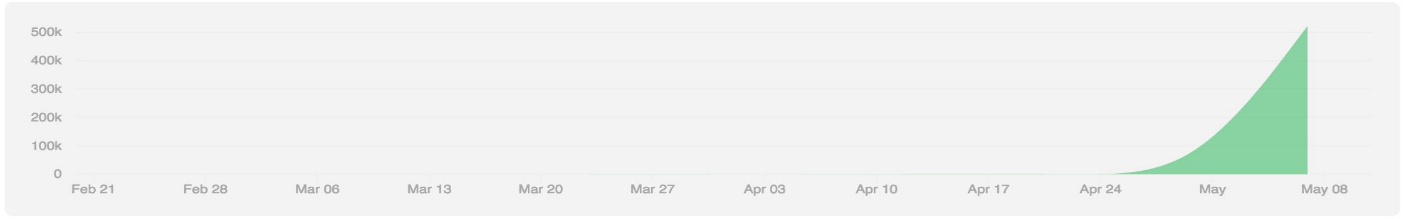
Contributions: **Deletions** ▾



Feb 21, 2016 – May 12, 2016

Contributions to master, excluding merge commits

Contributions: **Additions** ▾



Lessons Learned

- Time management and planning is key
- Look for help early
- FPGA Board is very delicate
- Testing takes more time than you expect