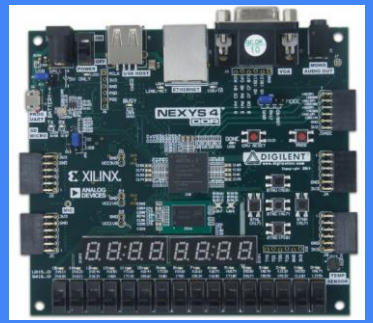
Visual Graphic Array Floating Point Calculator

Kelley Harris, Benjamin Hayes, Benjamin Jackson, William Strand



Oakland UNIVERSITY

VIVADO

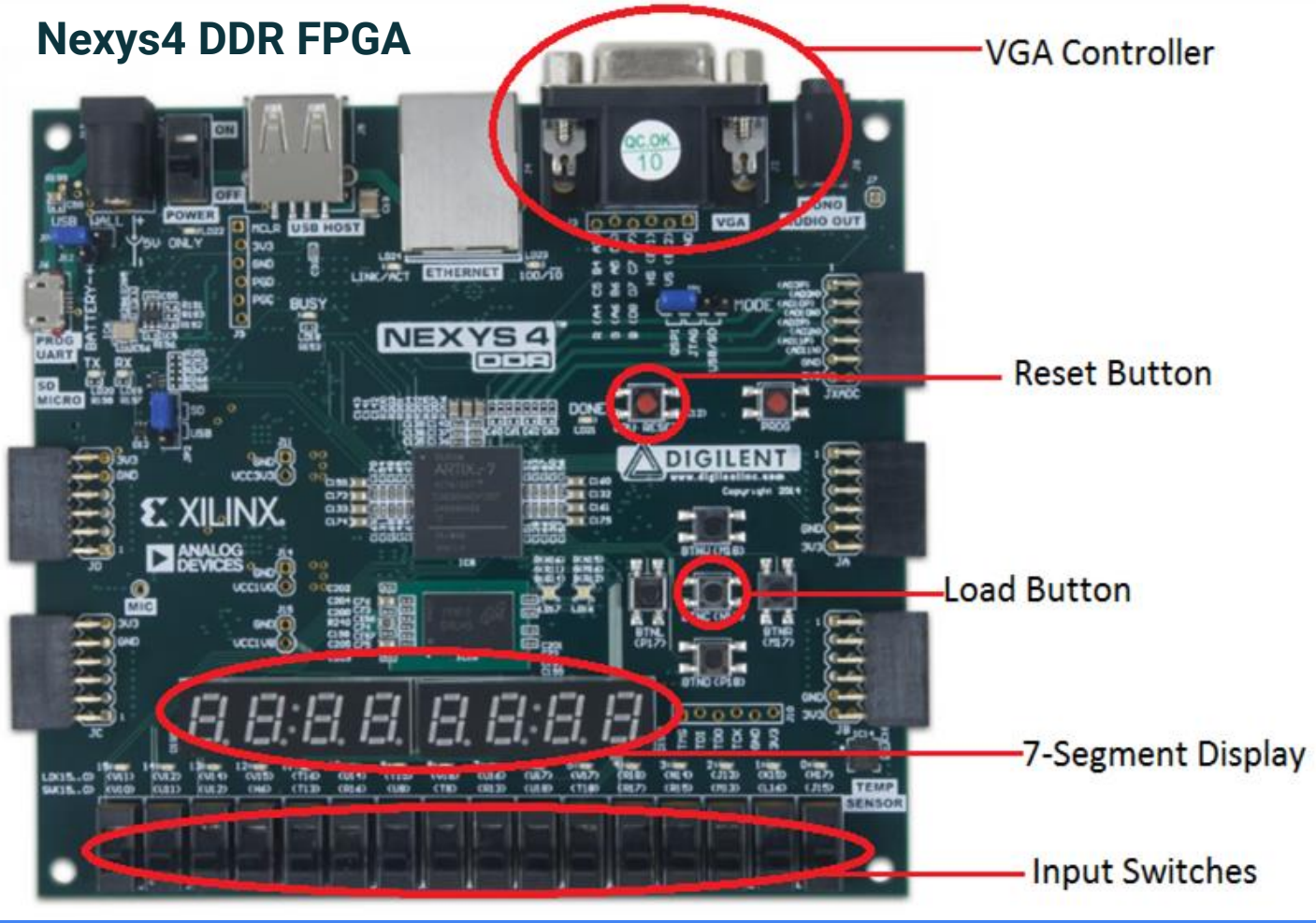
Introduction tidbits

Using a Nexys4 DDR Artix-7 Field Programmable Gate Array (FPGA), a 32-bit Floating-Point Calculator was implemented using the hardware description language, VHDL and the Vivado HL Webpack software program.

Images uploaded into MATLAB and converted into 64x64 text files are downloaded onto the VGA controller.

A Video Graphics Array (VGA) controller is then used to display the outputs generated by the floating point arithmetic unit on a monitor.

Nexys4 DDR FPGA



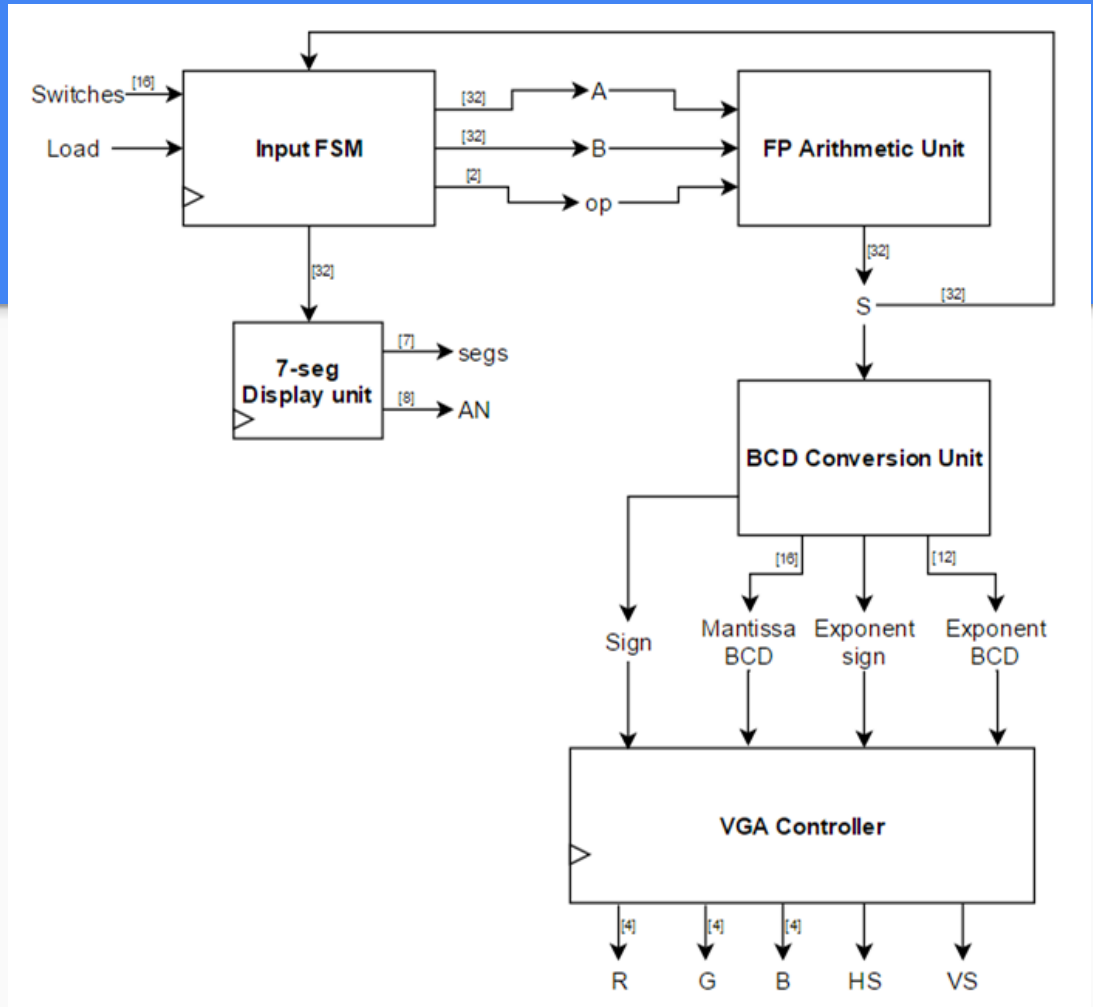
Set Up

1 - Input FSM and 7seg display

2 - FP Arithmetic

3 - BCD Conversion

4 - VGA Controller



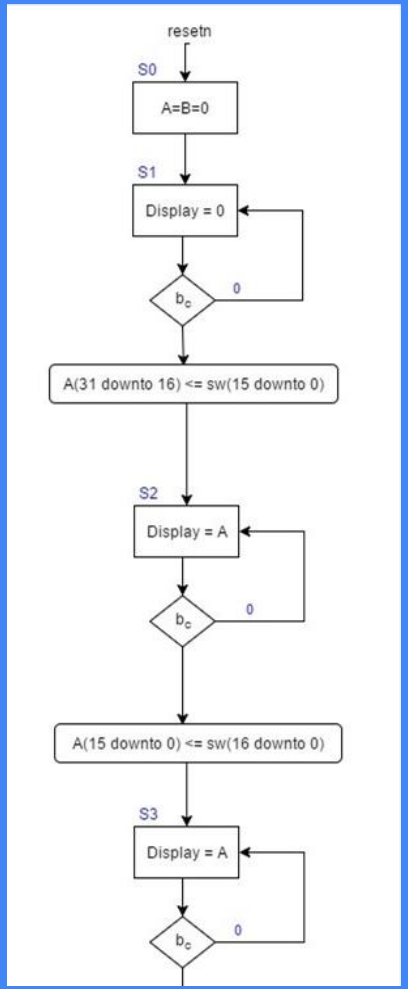
1- Input FSM and 7 Segment Display

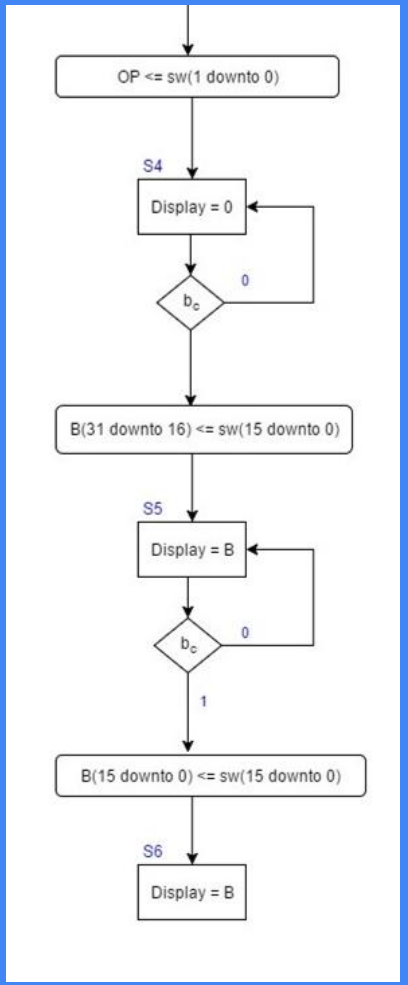
Resetn initializes the calculator and clears everything.

Two 32-bit Single floating point numbers (A, B) along with a 2-bit operand code (+, -, x, ÷) are entered into the NEXYS 4 DDR using the switches. The 32 bits for the FP numbers are entered 16 bits at a time.

While inputting A and B, their current value will be displayed in hex on the 7 segment display. After input, the calculator output will be shown on the seven segments and on a vga monitor.

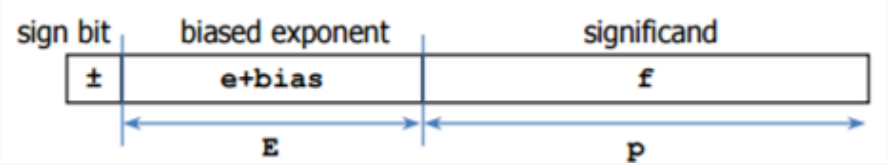
Finite State Machine used for inputting the 32-bit values into the calculator program.

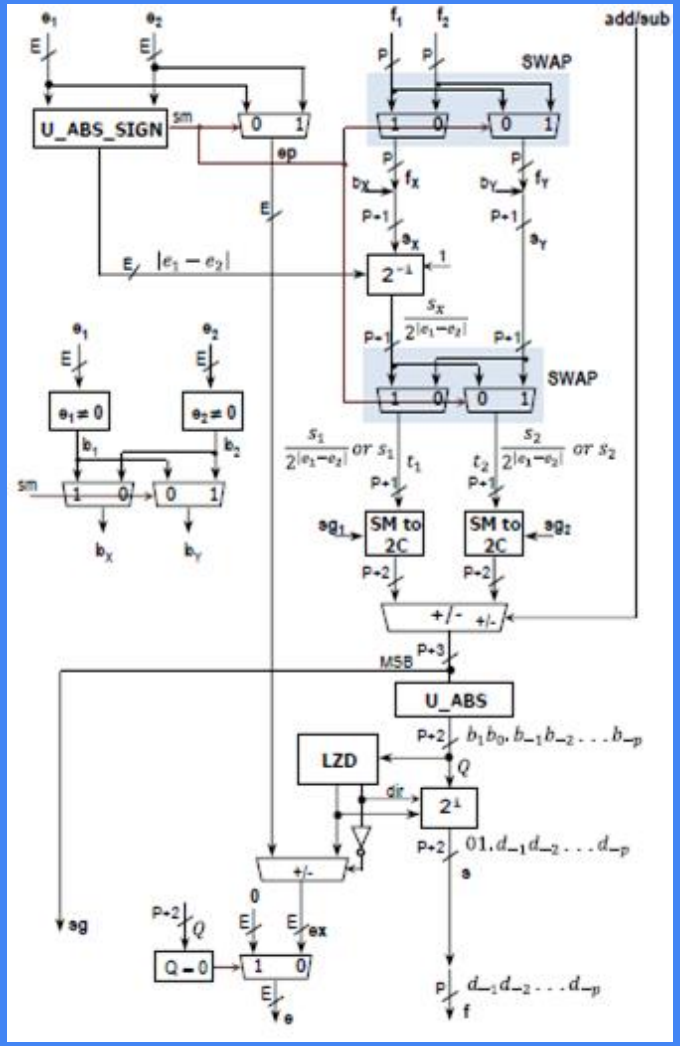


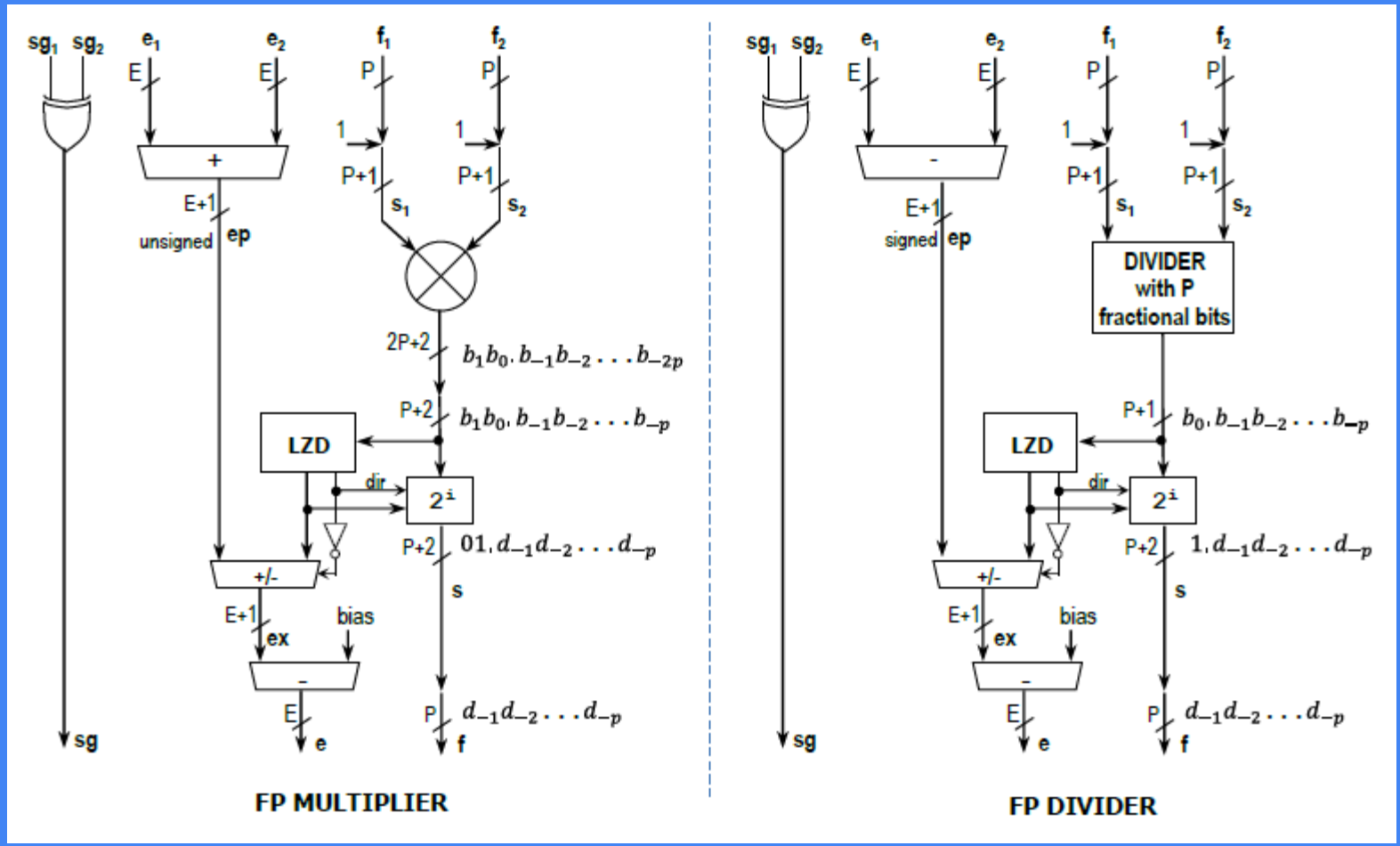


2 - Floating Point Arithmetic Unit

The floating point arithmetic unit takes in two, 32-bit floating point numbers along with a 2-bit operation code. It then feeds the inputs into the appropriate operational unit and outputs to a single 32-bit floating point number.

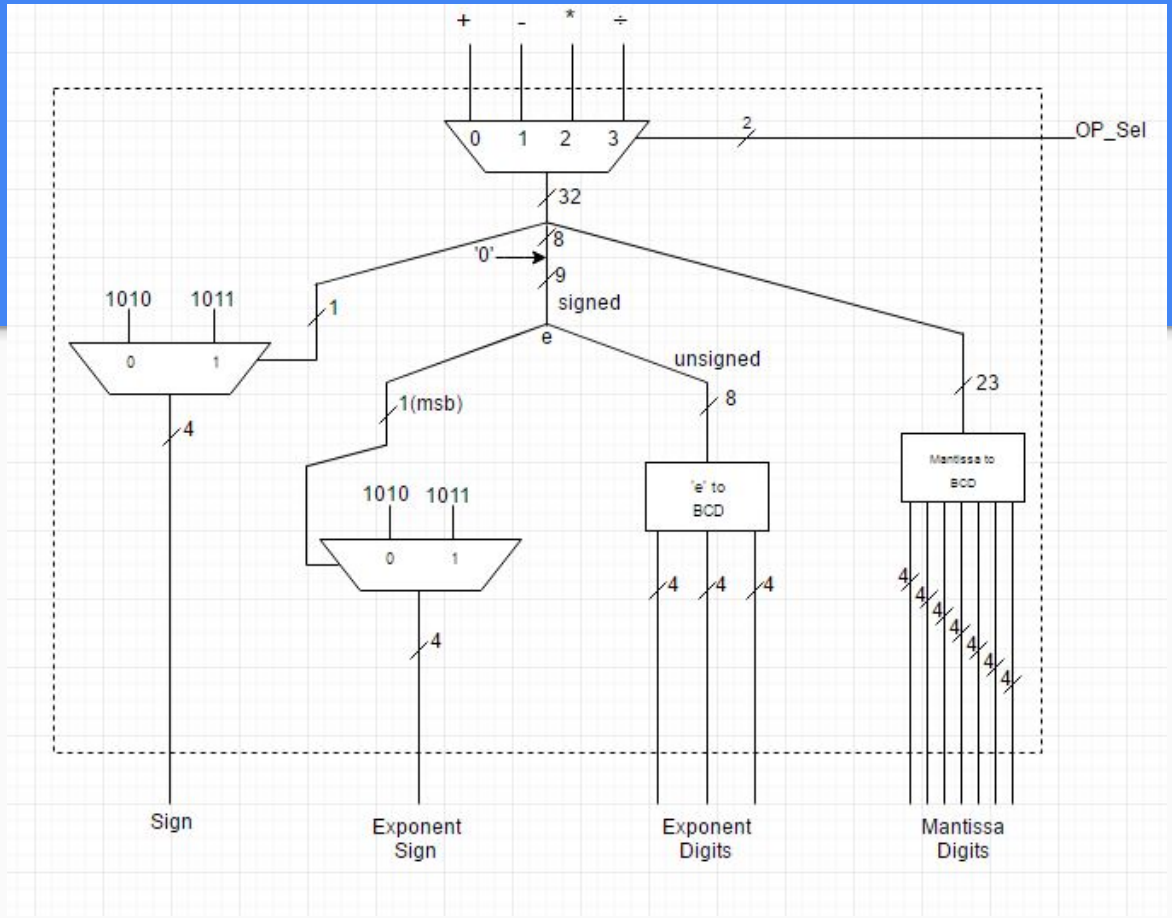



add/sub
SWAP
U_ABS_SIGN
sm
ep
SWAP
SM to 2C
SM to 2C
U_ABS
MSB
LZD
dir
sg
ex
e
f
s


sg1 sg2
e1
e2
E
E
+
E+1
unsigned
ep
f1
f2
P
P
1
1
P+1
P+1
s1
s2
2P+2
b1b0. b−1b−2 . . . b−2p
P+2
b1b0. b−1b−2 . . . b−p
LZD
dir
2^i
P+2
01. d−1d−2 . . . d−p
s
+/-
E+1
ex
bias
−
E
e
P
d−1d−2 . . . d−p
f
sg
FP MULTIPLIER
sg1 sg2
e1
e2
E
E
-
E+1
signed
ep
f1
f2
P
P
1
1
P+1
P+1
s1
s2
DIVIDER
with P
fractional bits
P+1
b0. b−1b−2 . . . b−p
LZD
dir
2^i
P+2
1. d−1d−2 . . . d−p
s
+/-
E+1
ex
bias
−
E
e
P
d−1d−2 . . . d−p
f
sg
FP DIVIDER

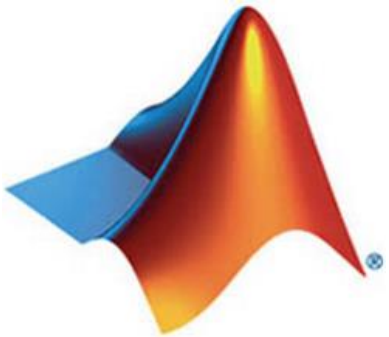
3- BCD Conversion

The Floating Point to BCD converter circuit receives the output of the Arithmetic Unit, splits it up into the sign, e+bias, and mantissa, and uses/alters these signals to produce the appropriate BCD representation of the number.



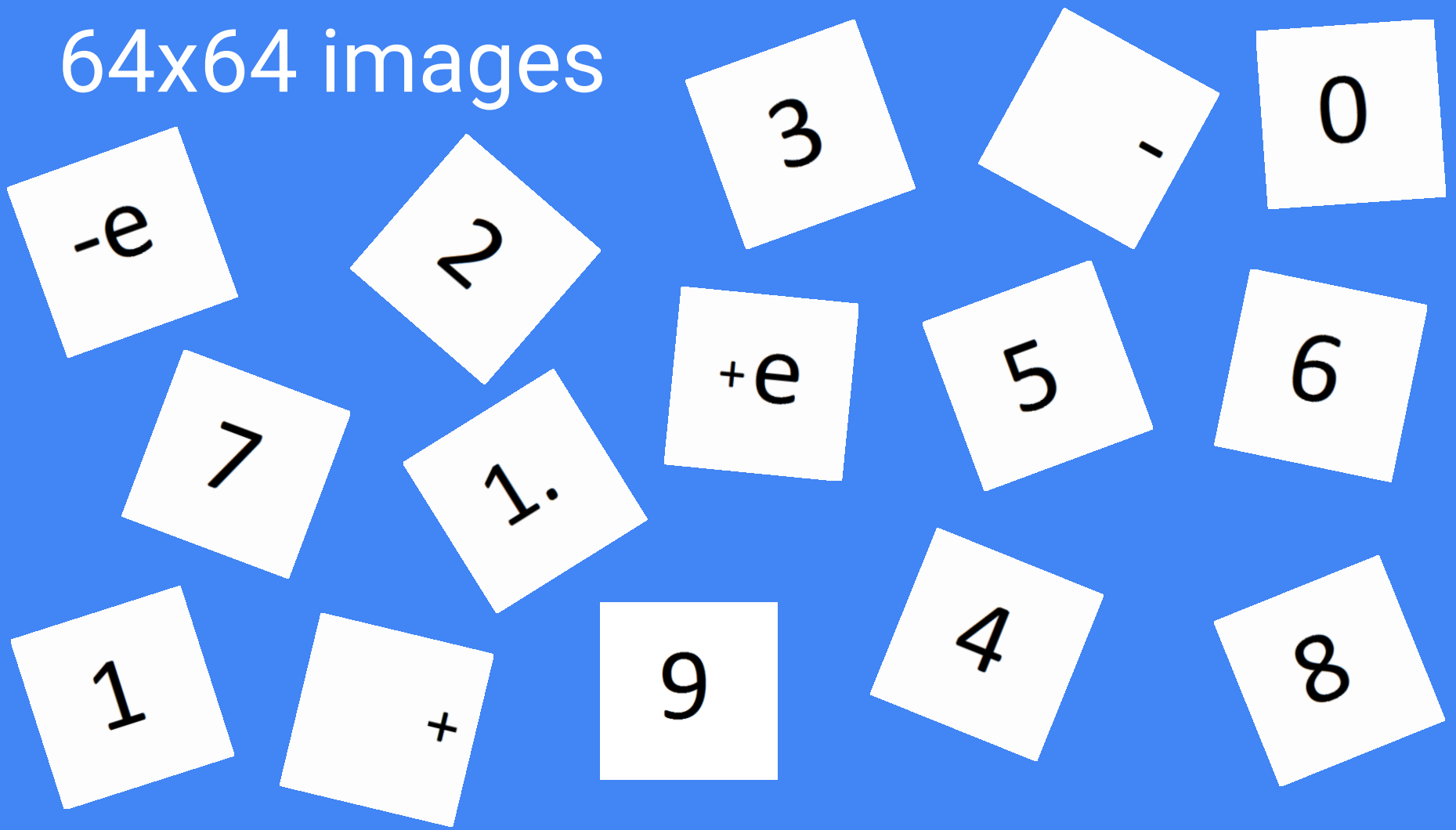
+/- 1 . - - - +/- e - - -

The provided MATLAB code used for converting PNG images into a usable TXT files



```
clear all; close all; clc;

I = imread ('negative.png'); % RGB image
figure; imshow(I);

% Resizing the image to 256x256:
IP = imresize(I,[64 64]);
figure; imshow (IP);

% 24-bit RGB image: we will convert it to a 12-bit RGB image:
for i = 1:3
    IN(:,:,i) = IP(:,:,i)/16; % every plane converted to 4 bits. right shift
end

figure; imshow(IN*16); % This is just so that 'imshow' can display the image properly

% ------------------------------------------------------------------------
% Converting to text file. Format: 0|R|G|B in hexadecimal
q = quantizer ('ufixed', 'round', 'saturate', [4 0]);
textfile = 'negative.txt';
fid = fopen (textfile, 'wt'); % generates text file in write mode

for i = 1:64
    for j = 1:64
        R = IN(i,j,1); G = IN(i,j,2); B = IN(i,j,3);
        Rh = num2hex(q, double(R)); Gh = num2hex(q, double(G)); Bh = num2hex(q, double(B));
        fprintf(fid, '0%s%s%s\n',Rh, Gh, Bh);
    end
end
```

64x64 images



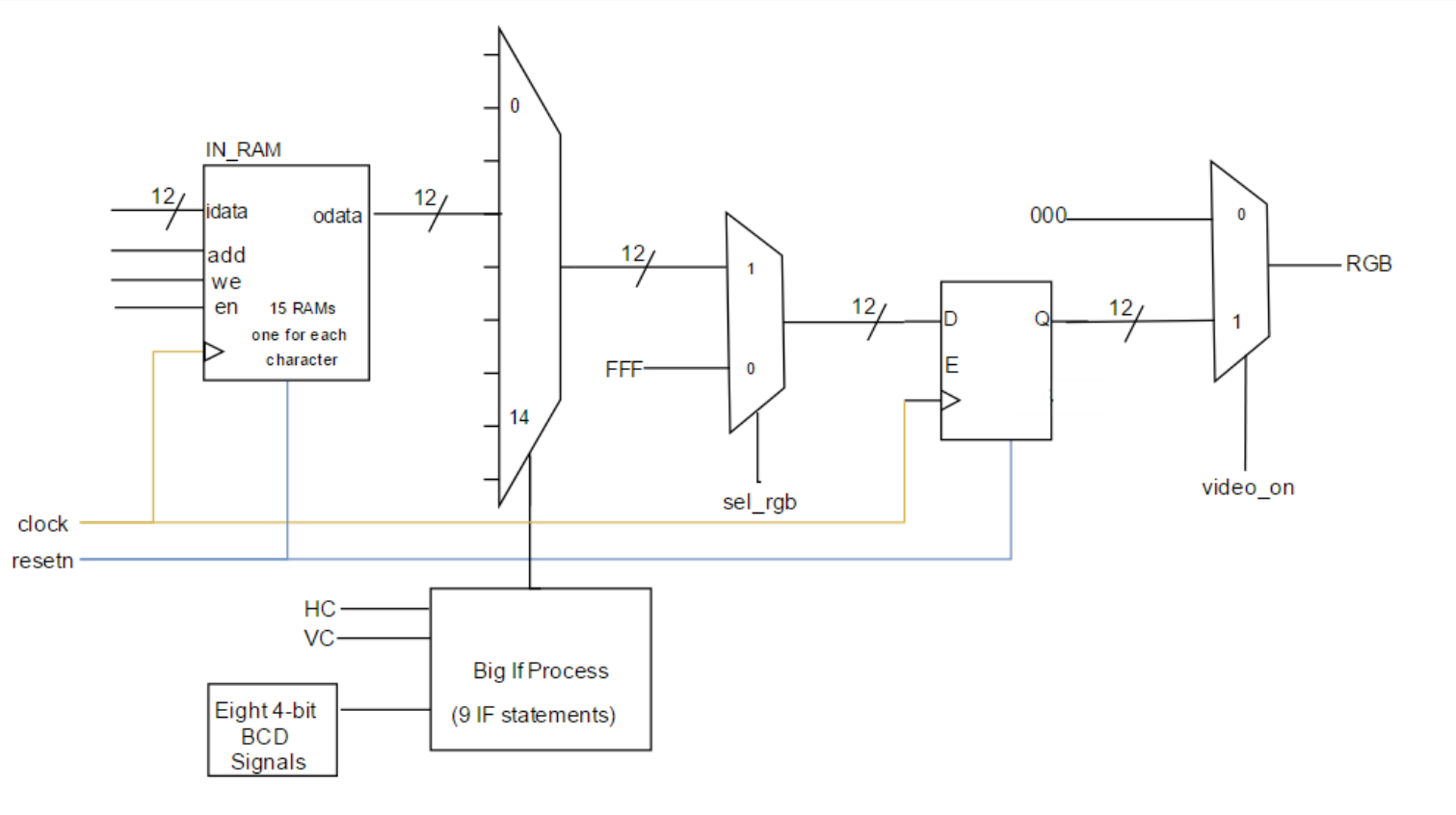
4 - VGA Controller

The VGA controller takes in the truncated digits of the calculator output after their conversion to BCD. The text files for 0-9, +, -, -e, +e, ‘1.’ are stored in separate RAMs, the output of each is the input of a 15-to-1 bus mux. This mux is selected by a series of If statements that depend on the current values of VC, HC and the the BCD value of the character to be displayed on that particular 64x64 area.

The first 4 significant digits of each number and its three digit exponent on the monitor, in the format of ±1.xxx ±e xxx.

VGA demo default screen

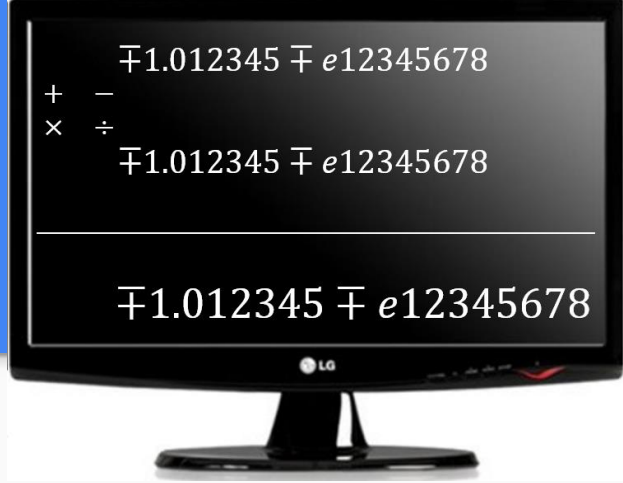
VGA circuit


IN_RAM
12
idata
odata
12
add
we
en
15 RAMs
one for each
character
0
14
12
1
0
FFF
sel_rgb
12
D
Q
E
12
000
0
1
RGB
video_on
clock
resetn
HC
VC
Big If Process
(9 IF statements)
Eight 4-bit
BCD
Signals

Examples/demo

Input A	OP	Input B	Result	VGA Display Theoretical	VGA Display Actual	% Error max (0.6%)
40B0 0000	+	C2FA 8000	-1.871*2^6	-1.871+e006	-1.867+e006	0.21%
50DA D000	-	D0FA D000	1.834*2^35	+1.834+e035	+1.828+e035	0.33%
7AB8 0000	*	8180 0000	1.437*2^-6	+1.437-e006	+1.437-e006	0%
FA39 0000	÷	4840 0000	-1.921*2^99	-1.921+e099	-1.921+e099	0%

Plausible Improvements



Utilize the keyboard for both input #’s and operator

Display the inputs and operator on VGA as well as the result

Increase number of precision bits of the result for accuracy

Include Denormal, Infinite (±∞), and Not a Number (NaN) as possible outputs

Option to show the answer in full decimal format or as a single

FIN