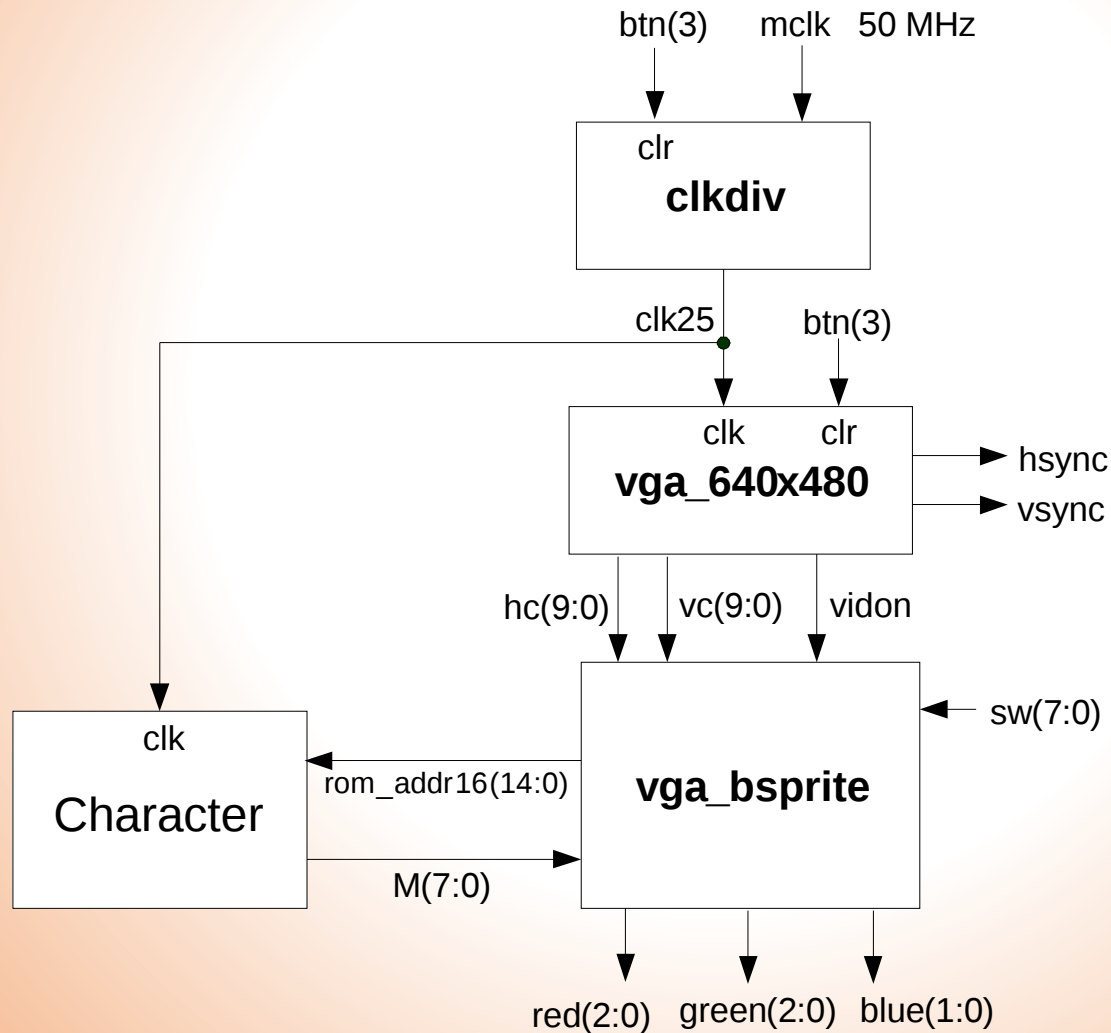


Presented by:
Michael Harris
Eric Yeh



Goal of the Game is to have tom capture jerry by having his image overlap jerry while jerry is trying to run away

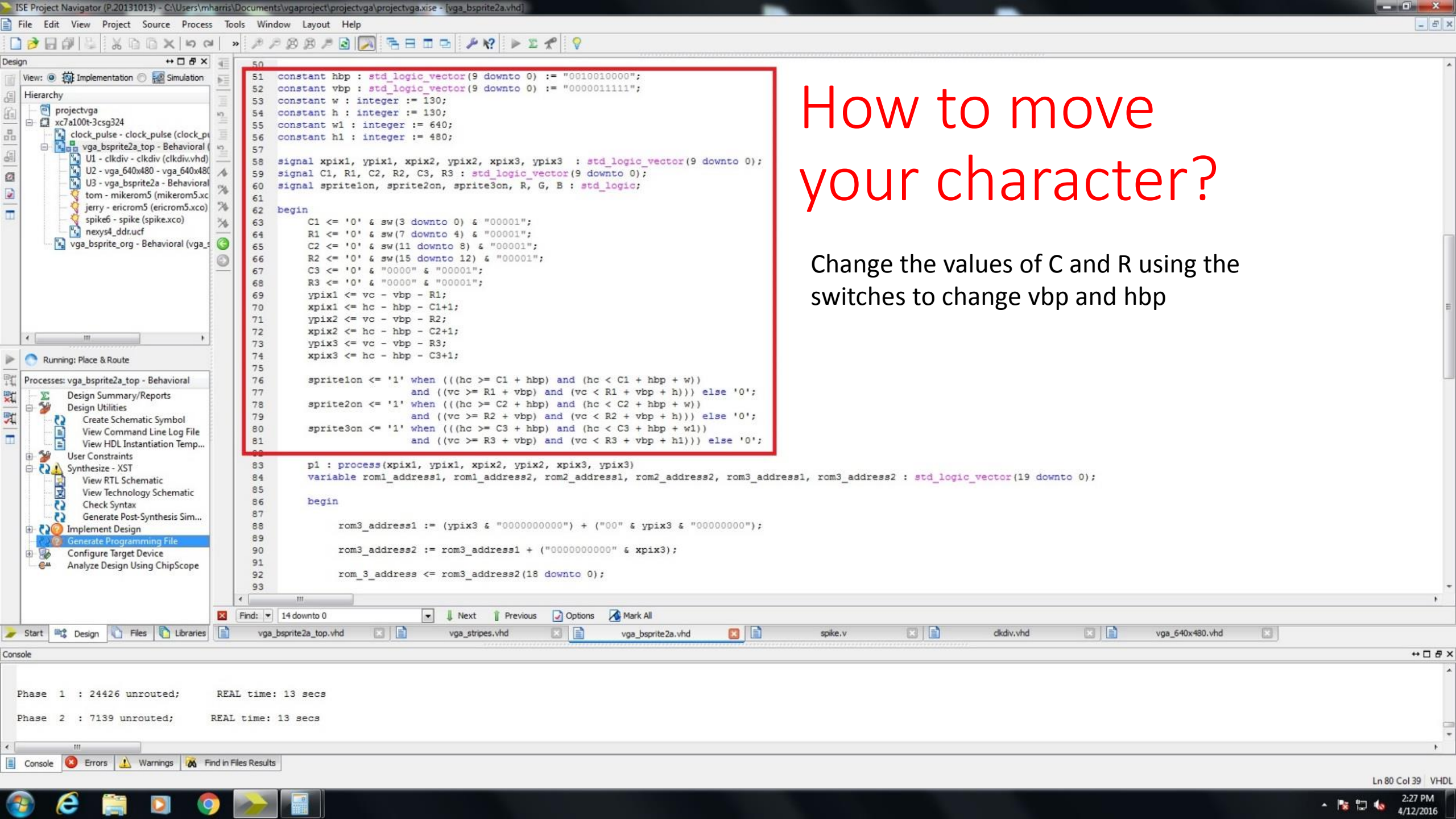
Top Level Design



Creating the Characters

We Used Microsoft Paint to remove Our Heads from a picture and placing them over tom and jerry





How to move your character?

Change the values of C and R using the switches to change vbp and hbp

```
50
51 constant hbp : std_logic_vector(9 downto 0) := "0010010000";
52 constant vbp : std_logic_vector(9 downto 0) := "0000011111";
53 constant w : integer := 130;
54 constant h : integer := 130;
55 constant w1 : integer := 640;
56 constant h1 : integer := 480;
57
58 signal xpix1, ypix1, xpix2, ypix2, xpix3, ypix3 : std_logic_vector(9 downto 0);
59 signal C1, R1, C2, R2, C3, R3 : std_logic_vector(9 downto 0);
60 signal spritelon, sprite2on, sprite3on, R, G, B : std_logic;
61
62 begin
63     C1 <= '0' & sw(3 downto 0) & "00001";
64     R1 <= '0' & sw(7 downto 4) & "00001";
65     C2 <= '0' & sw(11 downto 8) & "00001";
66     R2 <= '0' & sw(15 downto 12) & "00001";
67     C3 <= '0' & "0000" & "00001";
68     R3 <= '0' & "0000" & "00001";
69     ypix1 <= vc - vbp - R1;
70     xpix1 <= hc - hbp - C1+1;
71     ypix2 <= vc - vbp - R2;
72     xpix2 <= hc - hbp - C2+1;
73     ypix3 <= vc - vbp - R3;
74     xpix3 <= hc - hbp - C3+1;
75
76     spritelon <= '1' when ((hc >= C1 + hbp) and (hc < C1 + hbp + w))
77                     and ((vc >= R1 + vbp) and (vc < R1 + vbp + h)) else '0';
78     sprite2on <= '1' when ((hc >= C2 + hbp) and (hc < C2 + hbp + w))
79                     and ((vc >= R2 + vbp) and (vc < R2 + vbp + h)) else '0';
80     sprite3on <= '1' when ((hc >= C3 + hbp) and (hc < C3 + hbp + w1))
81                     and ((vc >= R3 + vbp) and (vc < R3 + vbp + h1)) else '0';
82
83     p1 : process(xpix1, ypix1, xpix2, ypix2, xpix3, ypix3)
84     variable rom1_address1, rom1_address2, rom2_address1, rom2_address2, rom3_address1, rom3_address2 : std_logic_vector(19 downto 0);
85
86     begin
87
88         rom3_address1 := (ypix3 & "0000000000") + ("00" & ypix3 & "0000000000");
89
90         rom3_address2 := rom3_address1 + ("0000000000" & xpix3);
91
92         rom_3_address <= rom3_address2(18 downto 0);
93
```

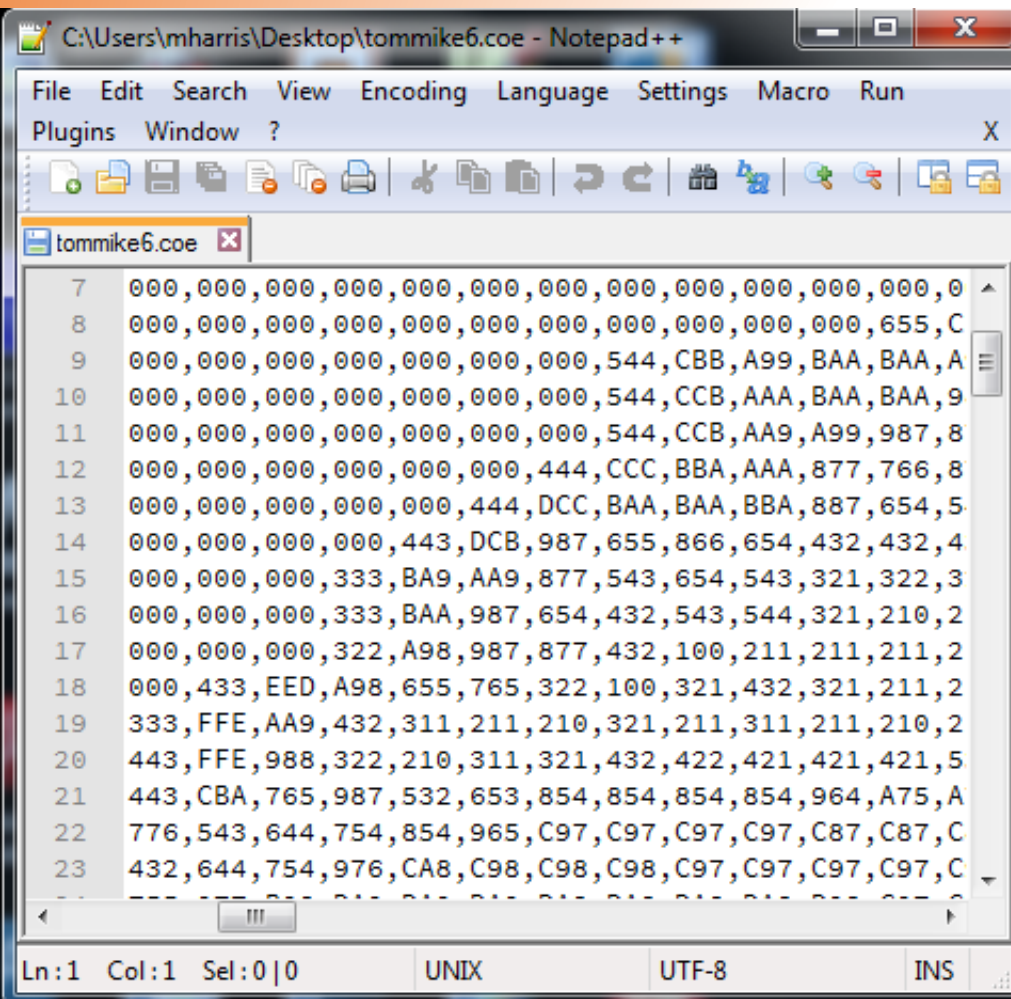
Phase 1 : 24426 unrouted; REAL time: 13 secs
Phase 2 : 7139 unrouted; REAL time: 13 secs

We Use the Matlab code to create the COE file For the Chosen Image

- % Read the image
- `img = imread(imgfile);`
- `h = size(img, 1); w = size(img, 2);`
- % Open the .coe file
- `s = fopen(outfile,'W');`
- % Print header
- `fprintf(s,'%s\n','; VGA Memory Map ');`
- `fprintf(s,'%s\n','; .COE file with hex coefficients ');`
- `fprintf(s,'; Height: %d, Width: %d\n\n', h, w);`
- `fprintf(s,'%s\n','memory_initialization_radix=16;');`
- `fprintf(s,'%s\n','memory_initialization_vector=');`
- % Convert color channels to binary
- `R = dec2bin(img(:,:,1)',8);`
- `G = dec2bin(img(:,:,2)',8);`
- `B = dec2bin(img(:,:,3)',8);`

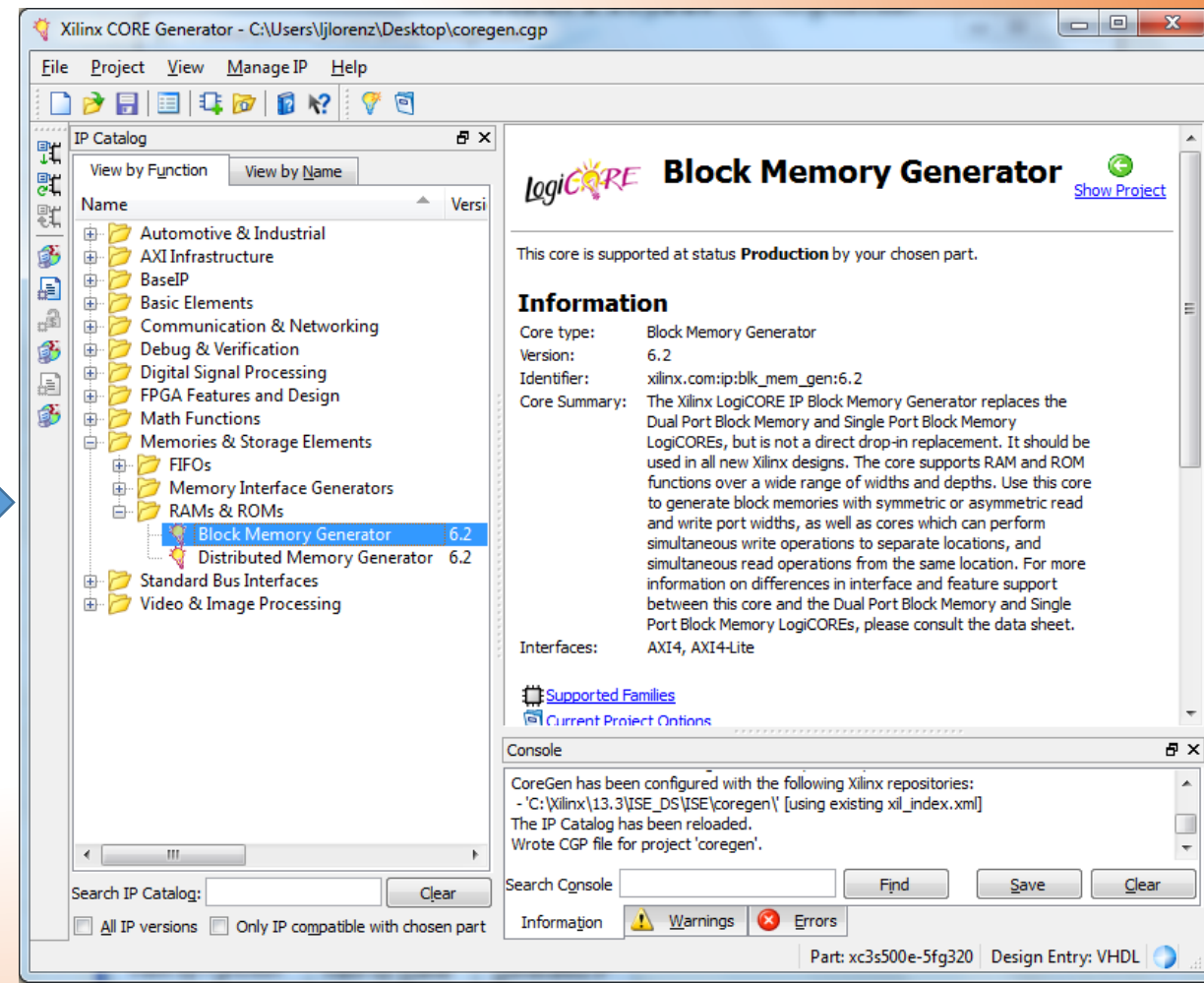
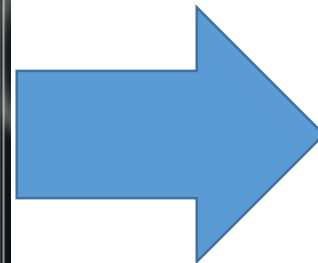
```
% Stitch together the output words
out = bin2dec([ R(:,1:4) G(:,1:4) B(:,1:4) ]);
img2 = img;
for i=1:h-1
    sol = i*w-w+1; % Start of line
    eol = i*w;     % End of line
    % Print out words
    fprintf(s,'%03X,',out(sol:eol,:));
    fprintf(s,'\n');
    % Save new image
    img2(i,:,1) = bin2dec(R(sol:eol,1:4))'*2^4';
    img2(i,:,2) = bin2dec(G(sol:eol,1:4))'*2^4';
    img2(i,:,3) = bin2dec(B(sol:eol,1:4))'*2^4';
end
% Print out the last row
fprintf(s,'%02X,',out(h*w-w+1:end-1,:));
fprintf(s,'%02X;',out(end,:));
img2(h,:,1) = bin2dec(R(h*w-w+1:end,1:4))'*2^4';
img2(h,:,2) = bin2dec(G(h*w-w+1:end,1:4))'*2^4';
img2(h,:,3) = bin2dec(B(h*w-w+1:end,1:4))'*2^4';
% Close the .coe file
fclose(s);
```

We Used the Core Generator to generate Image for the Created .COE file



A screenshot of the Notepad++ text editor. The title bar shows the file path: C:\Users\mharris\Desktop\tommike6.coe - Notepad++. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations. The text area displays a long sequence of hexadecimal values, one per line, starting from line 7. The status bar at the bottom indicates 'Ln:1 Col:1 Sel:0|0', 'UNIX', 'UTF-8', and 'INS'.

```
7 000,000,000,000,000,000,000,000,000,000,000,000,000,0
8 000,000,000,000,000,000,000,000,000,000,000,000,655,C
9 000,000,000,000,000,000,000,000,544,CBB,A99,BAA,BAA,A
10 000,000,000,000,000,000,000,000,544,CCB,AAA,BAA,BAA,9
11 000,000,000,000,000,000,000,000,544,CCB,AA9,A99,987,8
12 000,000,000,000,000,000,000,444,CCC,BBA,AAA,877,766,8
13 000,000,000,000,000,000,444,DCC,BAA,BAA,BBA,887,654,5
14 000,000,000,000,443,DCB,987,655,866,654,432,432,4
15 000,000,000,333,BA9,AA9,877,543,654,543,321,322,3
16 000,000,000,333,BAA,987,654,432,543,544,321,210,2
17 000,000,000,322,A98,987,877,432,100,211,211,211,2
18 000,433,EED,A98,655,765,322,100,321,432,321,211,2
19 333,FFE,AA9,432,311,211,210,321,211,311,211,210,2
20 443,FFE,988,322,210,311,321,432,422,421,421,421,5
21 443,CBA,765,987,532,653,854,854,854,854,964,A75,A
22 776,543,644,754,854,965,C97,C97,C97,C97,C87,C87,C
23 432,644,754,976,CA8,C98,C98,C98,C97,C97,C97,C97,C
```



A screenshot of the Xilinx CORE Generator application. The title bar shows the file path: C:\Users\jlorezn\Desktop\coregen.cgp. The menu bar includes File, Project, View, Manage IP, and Help. The IP Catalog on the left shows a tree view with 'Block Memory Generator' selected under 'RAMs & ROMs'. The right pane displays the 'Block Memory Generator' core details, including its status (Production), version (6.2), and a summary of its capabilities. The console at the bottom shows the configuration process.

Block Memory Generator

This core is supported at status **Production** by your chosen part.

Information

Core type: Block Memory Generator
Version: 6.2
Identifier: xilinx.com:ip:blk_mem_gen:6.2
Core Summary: The Xilinx LogiCORE IP Block Memory Generator replaces the Dual Port Block Memory and Single Port Block Memory LogiCOREs, but is not a direct drop-in replacement. It should be used in all new Xilinx designs. The core supports RAM and ROM functions over a wide range of widths and depths. Use this core to generate block memories with symmetric or asymmetric read and write port widths, as well as cores which can perform simultaneous write operations to separate locations, and simultaneous read operations from the same location. For more information on differences in interface and feature support between this core and the Dual Port Block Memory and Single Port Block Memory LogiCOREs, please consult the data sheet.

Interfaces: AXI4, AXI4-Lite

Supported Families
[Current Project Options](#)

Console

CoreGen has been configured with the following Xilinx repositories:
- 'C:\Xilinx\13.3\ISE_DS\ISE\coregen\' [using existing xil_index.xml]
The IP Catalog has been reloaded.
Wrote CGP file for project 'coregen'.

Search IP Catalog: Clear

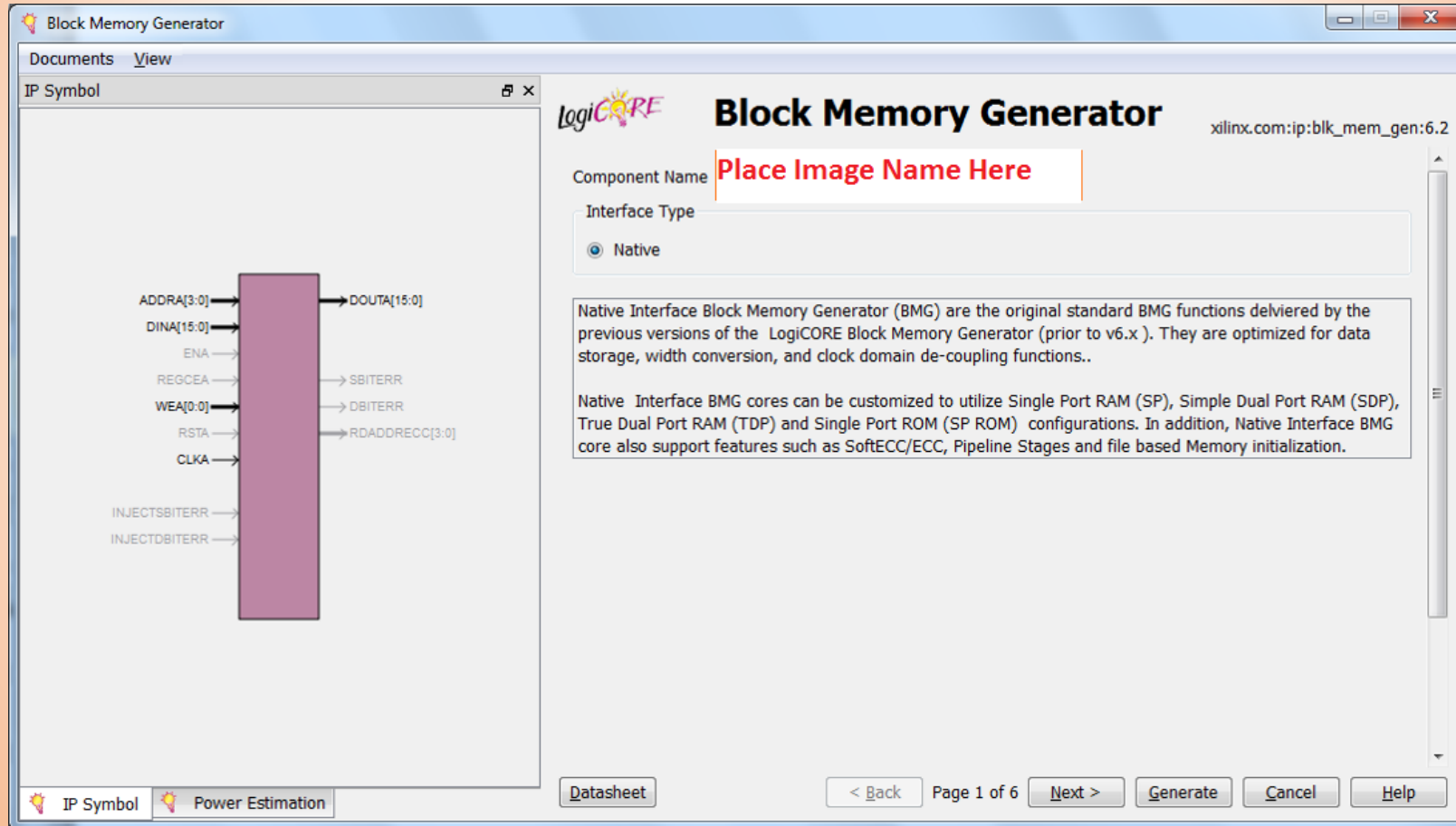
☐ All IP versions ☐ Only IP compatible with chosen part

Search Console: Find Save Clear

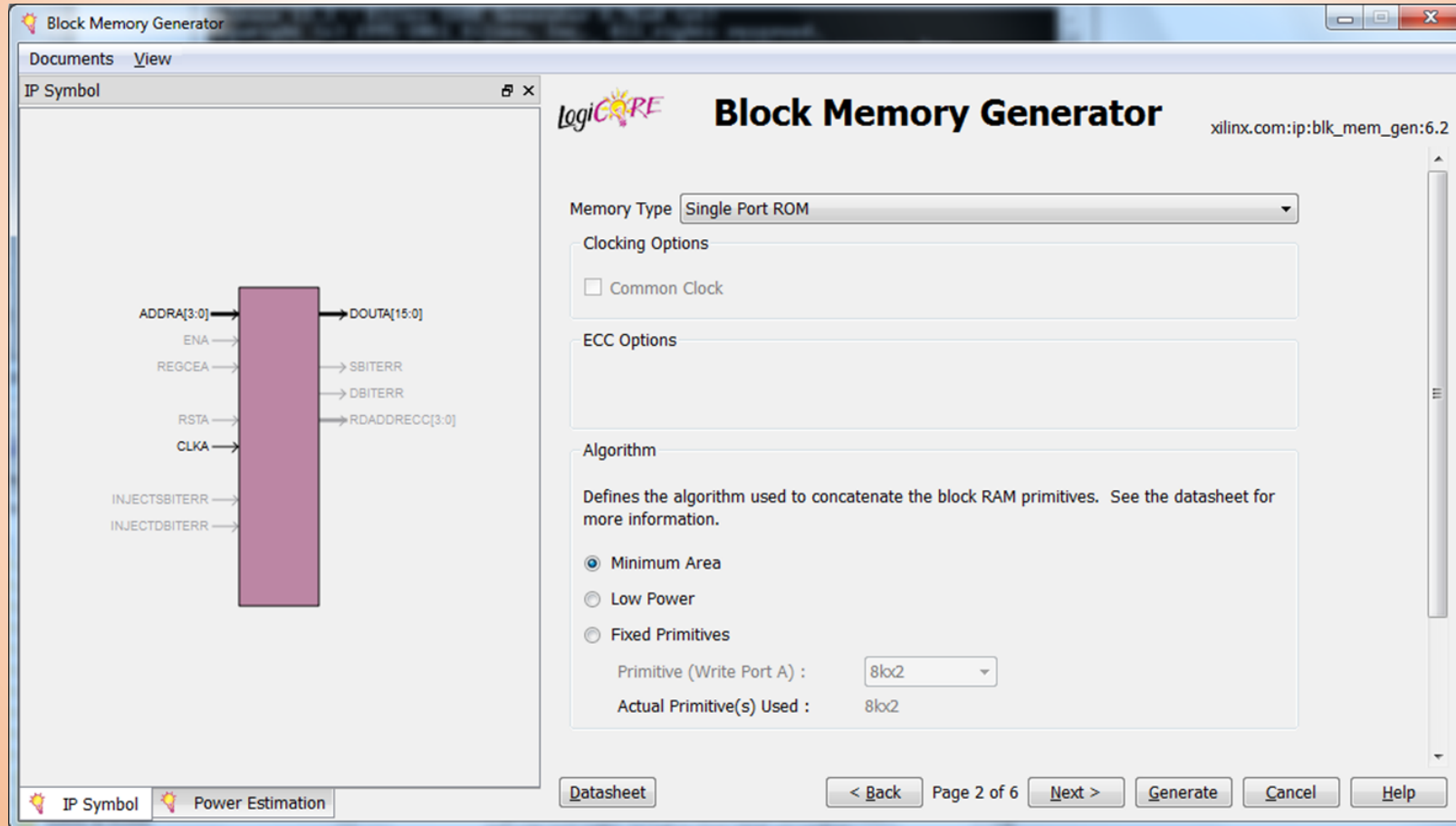
Information Warnings Errors

Part: xc3s500e-5fg320 Design Entry: VHDL

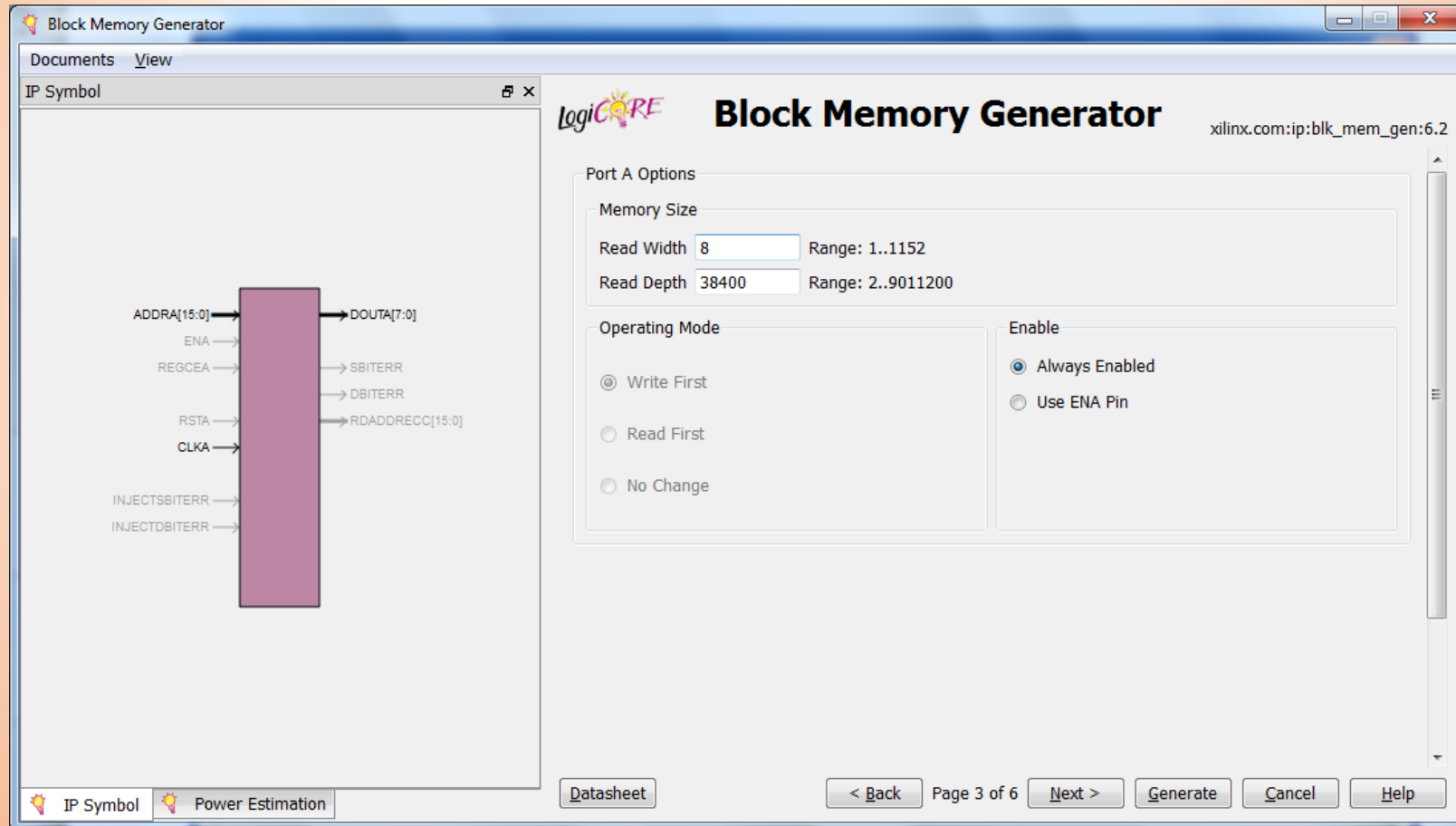
Name The image here



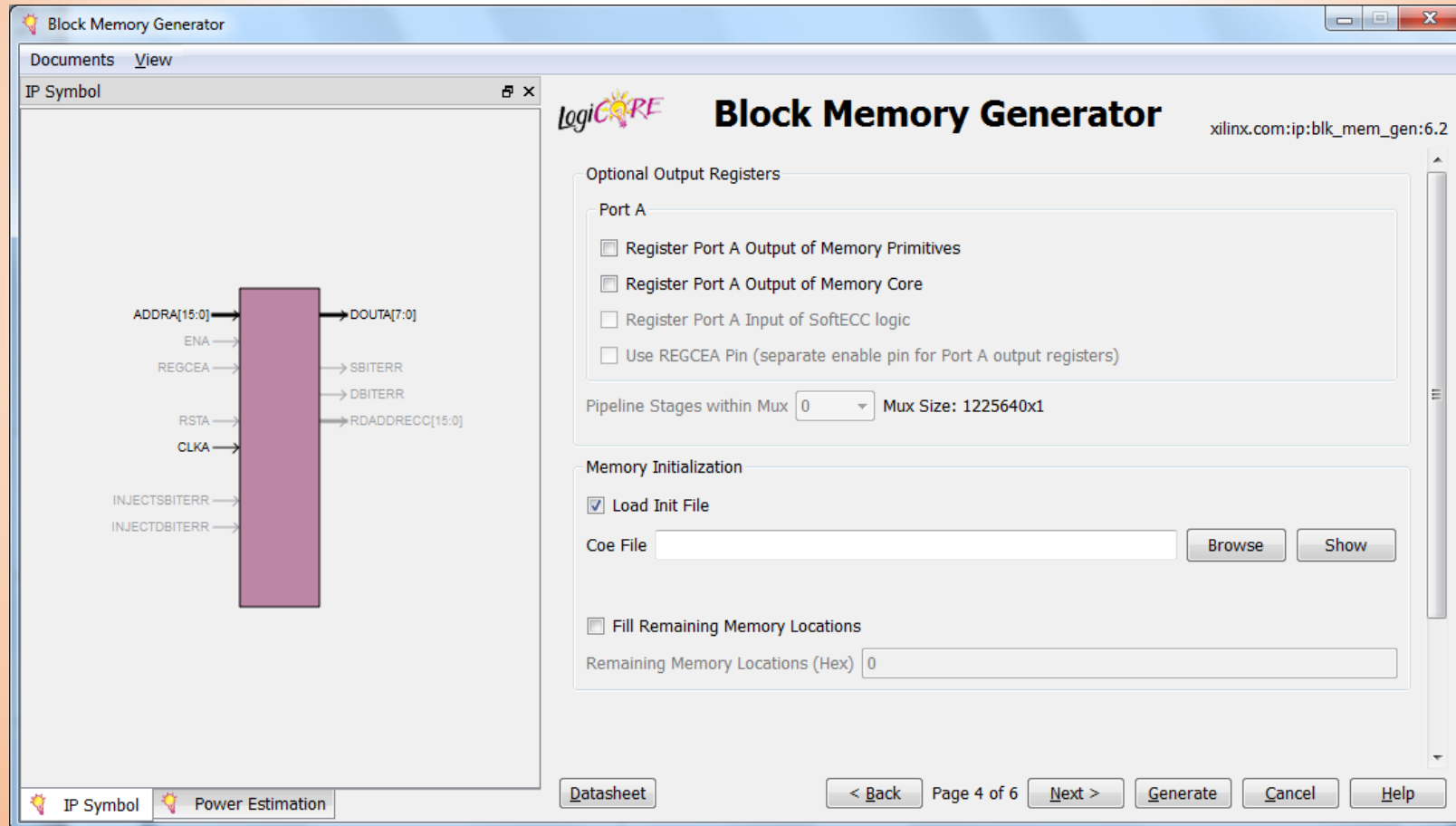
We Use Single Port Rom because it is easy and nice



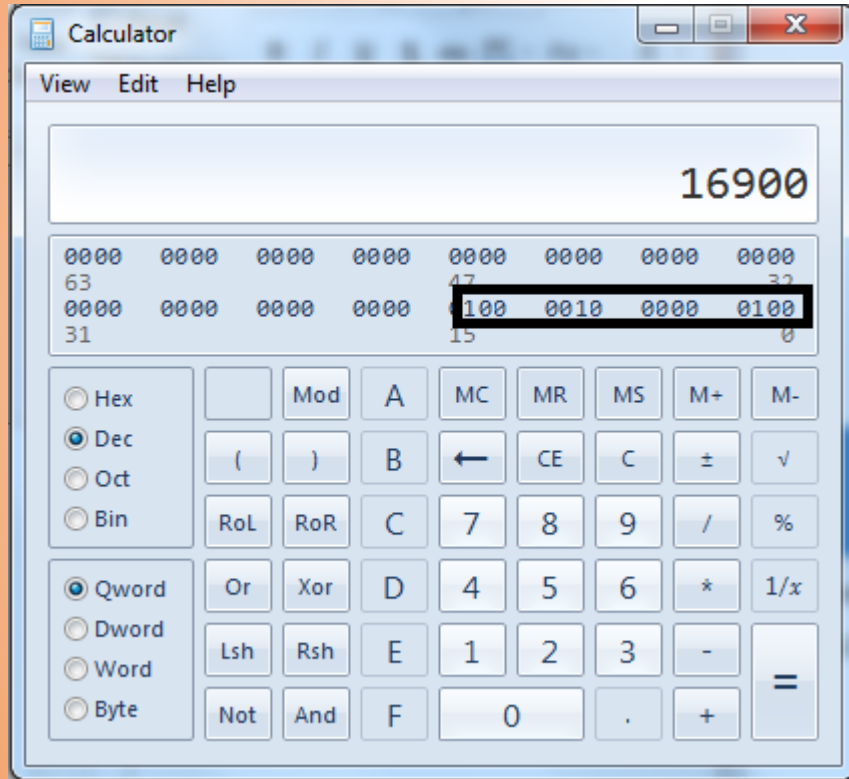
- Read Width is how many color bits for the image (we used 12 bits)
- Read Depth is the multiplication of the dimensions ex for a 130x130 pic the read depth is $130 \times 130 = 16900$



We Browse looking for that coe file then we generate it as a block memory



Using the windows calculator (programmer mode)



We are using 15 bits for the 130x130 image

So address <=(14 downto 0);

Demo
Time