

# Embedded Systems Lab 2:

## DE2 Graphics and Hardware

---

### Part 1: LCD Display

This section is designed to get the student used to reading and interpreting pre-written code. It also introduces the student to extracting information from data sheets and information tables and a first peek at a commonly missed difference between simulation, desktop implementation and deployment – start up timing.

### Set up

Download the Lab 2 Part 1 package from the course website and extract to your working directory.

### Understanding the code

Driving the LCD is split in to two parts. `LCD.v` contains the instructions and data to move to the LCD while `LCD_Controller.v` is responsible for actually driving the LCD pins.

An extract of the pin functions and instruction set for the LCD are shown below. The data to be transferred to the LCD is encoded in the look-up table in `LCD.v`

Table 1 LCD Pin Assignments

#### Pin Functions

Signal	No. of Lines	I/O	Device Interfaced with	Function
RS	1	I	MPU	Selects registers. 0: Instruction register (for write) Busy flag: address counter (for read) 1: Data register (for write and read)
$\overline{R/W}$	1	I	MPU	Selects read or write. 0: Write 1: Read
E	1	I	MPU	Starts data read/write.
DB4 to DB7	4	I/O	MPU	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. DB7 can be used as a busy flag.
DB0 to DB3	4	I/O	MPU	Four low order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. These pins are not used during 4-bit operation.

Table 2 LCD Instructions

Table 6 Instructions

Instruction	Code										Description	Execution Time (max) (when $f_{op}$ or $f_{osc}$ is 270 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	0	1	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 $\mu$ s
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 $\mu$ s
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 $\mu$ s
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 $\mu$ s
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 $\mu$ s
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 $\mu$ s
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 $\mu$ s

Table 3 The ASCII character set

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	:	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

 Source: [www.LookupTables.com](http://www.LookupTables.com)

## Activities

The data and instructions to the LCD are 8-bits wide. Why are the entries in LCD.v's LUT\_DATA 9 bits wide? What role does the most significant bit play (hint: the highest bit of LUT\_DATA is only read once, where and why)?

The instructions to initialise the LCD are given at the beginning of the LUT. Convert each hexadecimal instruction to its binary equivalent and look them up in the table of LCD instructions (Table 2 above). Comment each of the first 5 LUT entries describing what each entry does. Where full descriptions of the bit purposes aren't given above (e.g. the Entry Mode Set I/D bit), just write the bit name and value in the comment.

Display characters are given by their ASCII representation. Using the table of ASCII codes (Table 3 above), change the message displayed on the LCD to whatever you wish.

The Reset\_Delay module makes sure the LCD doesn't start receiving commands for some time after the FPGA starts up. Noting that the clock is 50MHz, how long does that module wait before it sets the reset line high?

Try commenting out the Reset\_Delay module instantiation and replacing it with

```
assign DLY_RST = 1;
```

The program should still work. Given that, what's the point of having this delayed reset? Hint: Refer to the Reset Function extract of the LCD data sheet shown below and think: What's different

between how you're using the FPGA in labs compared to how it might be used in an Embedded System?

## Reset Function

### Initializing by Internal Reset Circuit

An internal reset circuit automatically initializes the HD44780U when the power is turned on. The following instructions are executed during the initialization. The busy flag (BF) is kept in the busy state until the initialization ends (BF = 1). The busy state lasts for 10 ms after  $V_{CC}$  rises to 4.5 V.

1. Display clear
2. Function set:
  - DL = 1; 8-bit interface data
  - N = 0; 1-line display
  - F = 0;  $5 \times 8$  dot character font
3. Display on/off control:
  - D = 0; Display off
  - C = 0; Cursor off
  - B = 0; Blinking off
4. Entry mode set:
  - I/D = 1; Increment by 1
  - S = 0; No shift

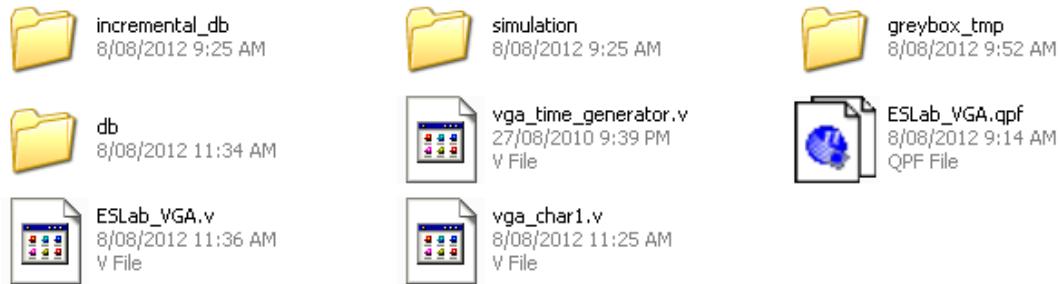
**Note:** If the electrical characteristics conditions listed under the table Power Supply Conditions Using Internal Reset Circuit are not met, the internal reset circuit will not operate normally and will fail to initialize the HD44780U. For such a case, initialization must be performed by the MPU as explained in the section, Initializing by Instruction.

## Part 2: Fun with Clocks

In this section, we will learn how to drive the VGA display, use the Quartus MegaFunction Wizard to generate pre-written parameterised modules and learn some of the subtler aspects of clock generation.

### Set up

Download the Lab 2 Part 1 package from the course website and extract to your working directory. You should have the following files:



Connect a VGA cable from the output of the DE2 board to the input on the back of your main monitor. When testing the VGA function, you will use the control keys on your monitor to switch backwards and forwards between the PC output and the board output.

### Understanding the Code

The code is split in to two parts; `ESLab_VGA.v` provides all the top level connections, while `vga_time_generator.v` takes care of generating synchronization and timing signals to the display.

The time generator produces two three signals of interest, `CounterX` and `CounterY` contain the current row and column being drawn to the screen and `VGA_BLANK_N` is high when these two are valid. In order to draw to the screen at a particular location, all one has to do is wait for the counters to be valid and at the correct location, then set the RGB signals to the desired colour. The example code draws a vertical red line and a horizontal green line in exactly this way.

### Activities

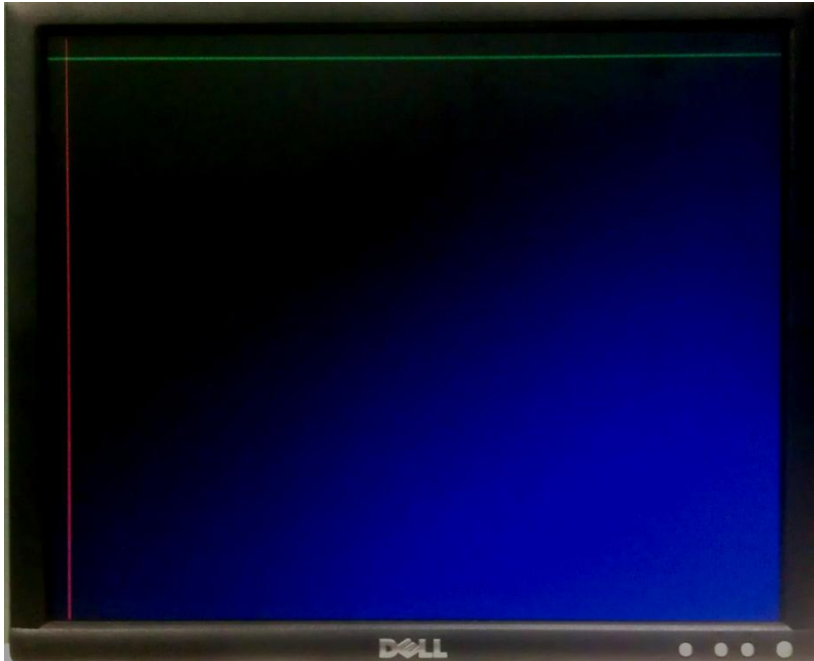
#### Part 1: Basic VGA

How many bits are used for each colour channel and therefore how many different colours can be displayed on the screen with this code?

At what resolution is the screen being driven?

The VGA timing generator has 'Porch' timings. These are extra pixels added off the screen at the top, bottom, left and right to provide padding around the main image. When the VGA code is writing to this porch area, the `VGA_BLANK_N` is set to 0 but does `CounterX` and `CounterY` start inside the porch area or inside the main screen? That is, is the first visible pixel in the X direction `CounterX == 0` or `CounterX == hfporch (=16)`? Answer this question by looking at the code in `vga_timing_generator.v` then confirm experimentally.

The blue channel currently isn't utilised for anything. Write code to generate a blue diagonal gradient as shown below. It might be helpful to start with a vertical gradient first.



### Part 2: Clock Generation and the Megafunction Wizard

The VGA Timing Generator requires a 25MHz clock input. This is generated from the 50MHz signal with the following code:

```
always @( posedge CLOCK_50 )  
    VGA_CLK <= VGA_CLK + 1;
```

In lectures, it was explained that generating your own clocks by dividing external ones in logic is generally a bad idea. Logic propagation times through the chip are much greater than clock propagation times, so clock skew issues can arise. In this particular case though, it's quite safe. To see why, open the "Compilation Report" tab, expand the "Fitter" item and click "Messages". A screen shot of this is shown below.

The screenshot shows the Quartus II Fitter Messages window. The left pane displays a tree view of the project files, with the 'Messages' folder under the 'Fitter' section highlighted. The right pane shows a list of messages, including 'Info (170190): Fitter placement preparat' which is highlighted. The bottom status bar indicates 'Message: 1055 of 1075'.

Type	Message
Info	Info (171003): Fitter is performing an A
Warning	Warning (292013): Feature LogicLock is c
Info	Info (176444): Device migration not sele
Info	Info (169124): Fitter converted 5 user p
Warning	Warning (15714): Some pins have incomple
Critical Warning	Critical Warning (332012): Synopsys Desi
Info	Info (332144): No user constrained base
Info	Info (332143): No user constrained clock
Info	Info (332123): Deriving Clock Uncertain
Info	Info (332130): Timing requirements not s
Info	Info (176353): Automatically promoted nc
Info	Info (176353): Automatically promoted nc
Info	Info (176233): Starting register packing
Extra Info	Extra Info (176273): Performing register
Extra Info	Extra Info (176274): Completed register
Extra Info	Extra Info (176236): Started Fast Input/
Extra Info	Extra Info (176237): Finished Fast Input
Extra Info	Extra Info (176248): Moving registers ir
Extra Info	Extra Info (176249): Finished moving reg
Info	Info (176235): Finished register packing
Warning	Warning (15709): Ignored I/O standard as
Warning	Warning (15705): Ignored locations or re
Info	Info (171121): Fitter preparation operat
Info	Info (170189): Fitter placement preparat
Info	Info (170190): Fitter placement preparat
Info	Info (170191): Fitter placement operatic
Info	Info (170137): Fitter placement was succ
Info	Info (170192): Fitter placement operatic
Info	Info (170193): Fitter routing operations
Info	Info (170195): Router estimated average
Info	Info (170194): Fitter routing operations
Info	Info (170199): The Fitter performed an A
Info	Info (334003): Started post-fitting dele
Info	Info (334004): Delay annotation complete
Warning	Warning (171167): Found invalid Fitter s
Warning	Warning (169177): 1 pins must meet Alter
Info	Info: Quartus II 32-bit Fitter was succe

The Quartus fitter is smart enough to recognize some logic nets as actually being clocks and “Promotes” them to clock nets, for which clock skew doesn’t matter as much. Find the message in the window that confirms that the `VGA_CLK` net has been recognized as a clock signal and is therefore perfectly safe to use in this fashion. Which other net has been promoted to a clock? Look at the code for that second net, does its behaviour look approximately “clock-like”?

Expand the “TimeQuest Timing Analyzer” section of the Compilation Messages tab, then the “Slow 1200mV 85C Model” section and click “Fmax Summary”. The `VGA_CLK` must run at 25MHz, has the promotion to clock net meant that that frequency can be achieved?

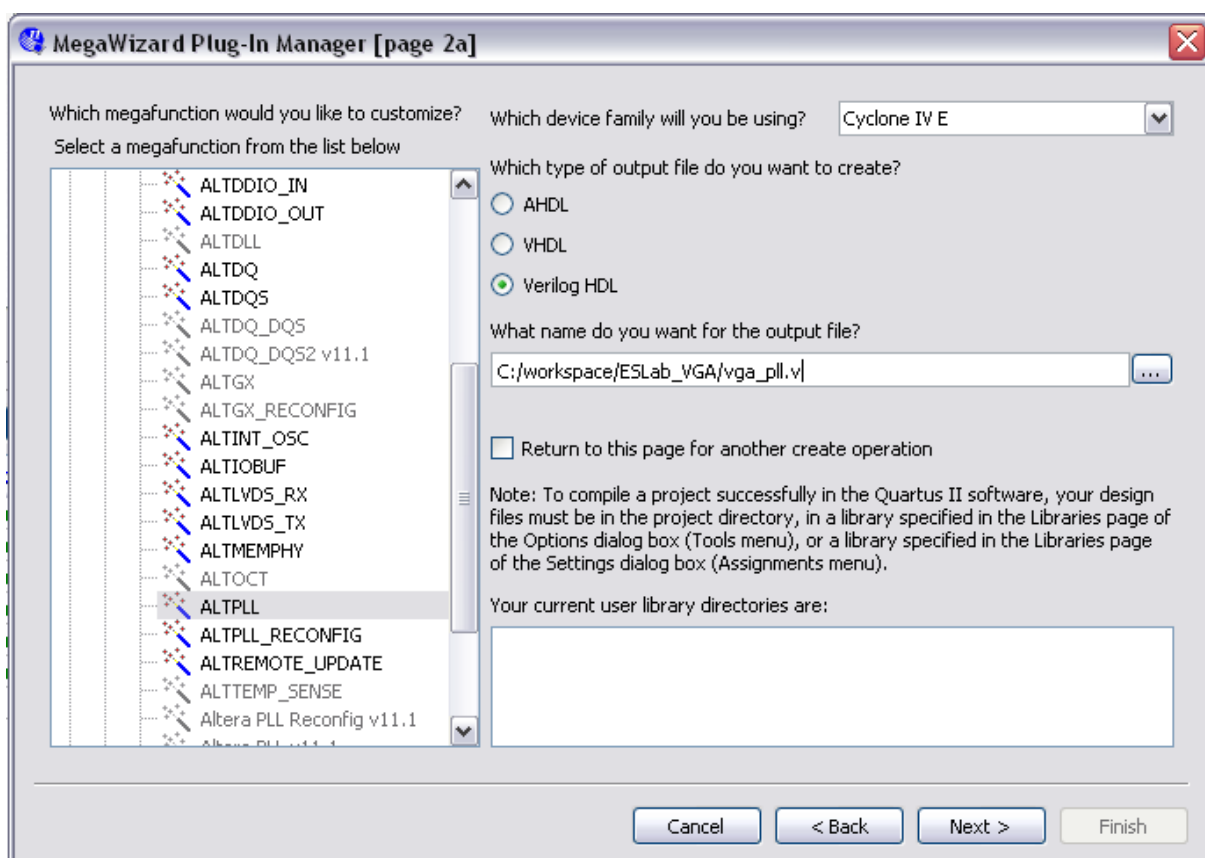


There are a limited number of clock nets for the Fitter to work with and it might not always get this promotion correct. To make the clock generation explicit, one might use a PLL.

From the Tools menu, select Megawizard Plug-in Manager. Click Next to create a new custom megafunction variation.

Look through the list of available logic blocks, there are options to create floating- or fixed-point arithmetic blocks, communications interfaces and also the Nios II soft-core processor that will be examined in a future lab.

Select the ALTPLL block under the I/O tab and enter 'vga\_pll.v' on the end of the output file path. Ensure that the Verilog HDL language and Cyclone IV E device family are selected. Click Next.





Set the Input Clock frequency to 50MHz and click Next.

MegaWizard Plug-In Manager [page 3 of 14]

**ALTPLL** About Documentation

1 Parameter Settings 2 PLL Reconfiguration 3 Output Clocks 4 EDA 5 Summary

General/Modes > Inputs/Lock > Bandwidth/SS > Clock switchover >

vga\_pll

inclk0 frequency: 50.000 MHz  
Operation Mode: Normal

Clk	Ratio	Ph (deg)	DC (%)
c0	1/1	0.00	60.00

Cyclone IV E

Currently selected device family: Cyclone IV E

☒ Match project/default

Able to implement the requested PLL

**General**

Which device speed grade will you be using? 7

☐ Use military temperature range devices only

What is the frequency of the inclk0 input? 50 MHz

☐ Set up PLL in LVDS mode Data rate: Not Available Mbps

**PLL Type**

Which PLL type will you be using?

☐ Fast PLL ☐ Enhanced PLL ☒ Select the PLL type automatically

**Operation Mode**

How will the PLL outputs be generated?

☒ Use the feedback path inside the PLL

☒ In normal mode

☐ In source-synchronous compensation Mode

☐ In zero delay buffer mode

☐ Connect the fbimic port (bidirectional)

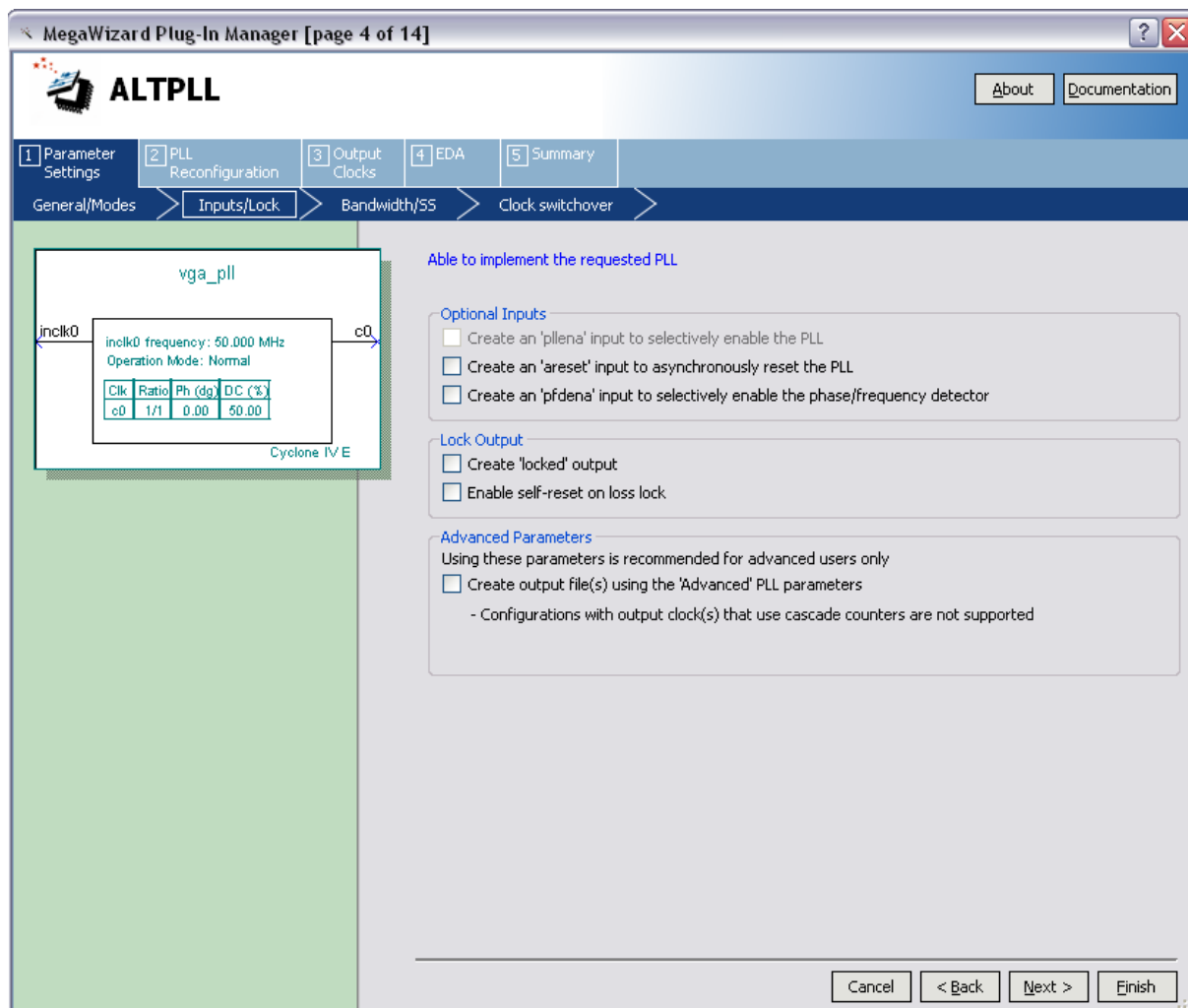
☐ With no compensation

☐ Create an 'fbim' input for an external feedback (External Feedback Mode)

Which output clock will be compensated for? c0

Cancel < Back Next > Finish

Uncheck all the options for extra signals as shown below.



Click Next several times leaving the tabs at their default values until you reach the configuration for clk c0 shown below. Check "Enter output clock frequency" and set it to 25MHz.

**ALTPLL**

1 Parameter Settings 2 PLL Reconfiguration 3 **Output Clocks** 4 EDA 5 Summary

clk c0 > clk c1 > clk c2 > clk c3 > clk c4

**c0 - Core/External Output Clock**  
Able to implement the requested PLL

☒ Use this clock

**Clock Tap Settings**

☒ Enter output clock frequency:  
☐ Enter output clock parameters:  
Clock multiplication factor  
Clock division factor  
Clock phase shift  
Clock duty cycle (%)

Requested Settings: 25 MHz  
Actual Settings: 25.000000

1 1 1 0.00 deg 50.00

Note: The displayed internal settings of the PLL is recommended for use by advanced users only

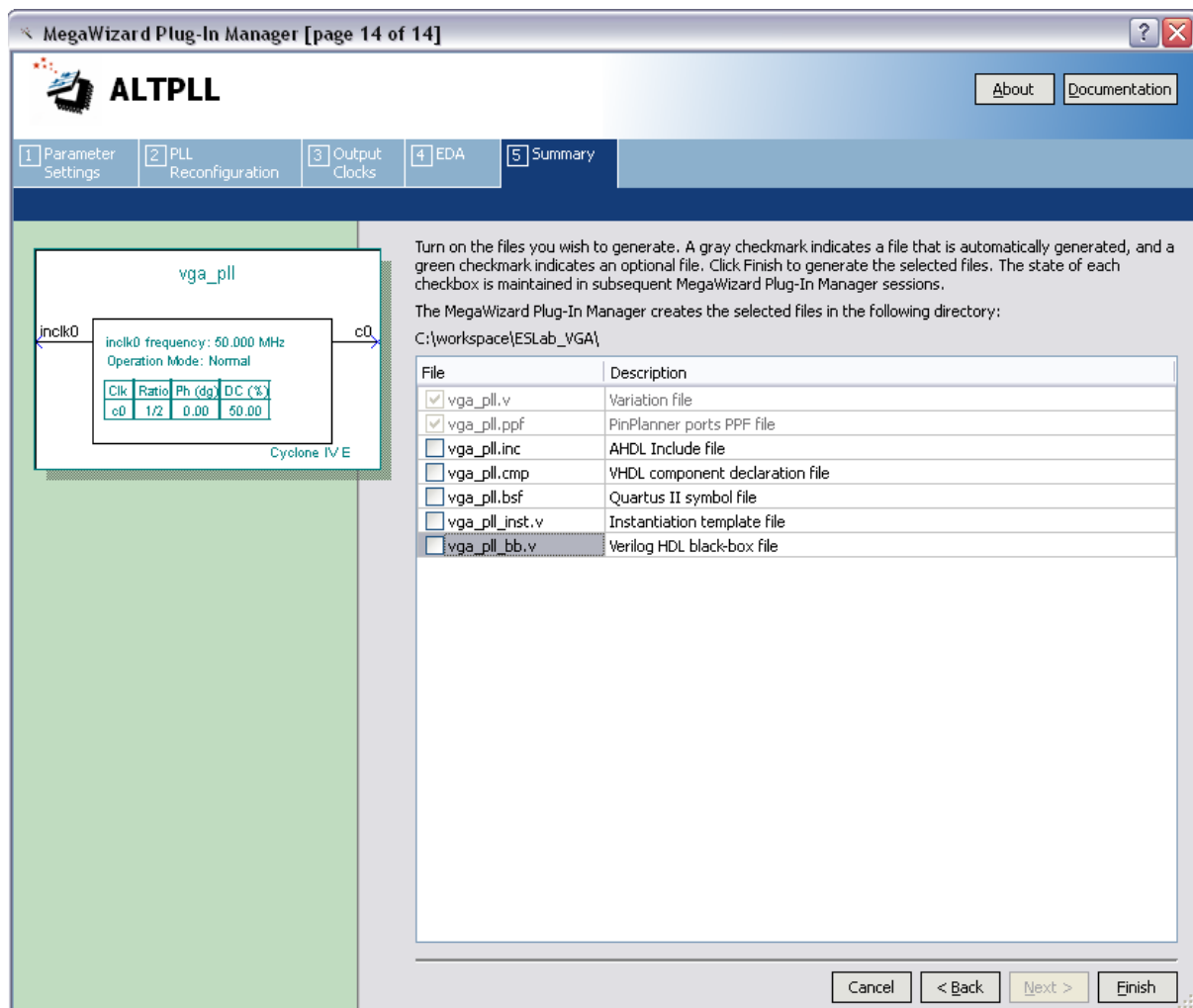
Description	Value
Primary clock VCO frequency (MHz)	6...
Modulus for M counter	12

Per Clock Feasibility Indicators: c0 c1 c2 c3 c4

Cancel < Back Next > Finish

Each PLL can generate up to five clocks (c0 to c4), each with their own frequency, phase shift and duty cycle. In this case we only need a single clock, so once c0 is configured, click Finish.

Uncheck the `vga_pll_bb.v` field and click Finish again.



On the "Files" tab of the Project Navigator, you will now see `vga_pll.qip`. Expand that, you will see the source file `vga_pll.v`. Have a look through that file, nothing should be edited by hand however you need to note the module port list so you know how to instantiate the device.

Back in `ESLab_VGA.v`, comment out the `always` block that used to generate the `VGA_CLK` code and write the following instead:

```
vga_pll vp11 (.inclk0(CLOCK_50), .c0(VGA_CLK));
```

Remove the 'reg' specifier from the `VGA_CLK` input (PLLs drive wires) then recompile the design and verify that it still works.

### Part 3: VGA Characters

When correctly instantiated, the following module will draw a 10 x 200 pixel white box to the screen with the top-left corner given as an input to the module.

```

module vga_char1 (
    counterX,
    counterY,
    baseX,
    baseY,
    vga_r,
    vga_g,
    vga_b);

input [11:0] counterX;
input [11:0] counterY;
input [11:0] baseX;
input [11:0] baseY;

output [9:0] vga_r;
output [9:0] vga_g;
output [9:0] vga_b;

wire inBox;

assign inBox = counterX > baseX &&
               counterX < baseX + 10 &&
               counterY > baseY &&
               counterY < baseY + 200;

assign vga_r = inBox ? 10'h3ff : 10'b0;
assign vga_g = inBox ? 10'h3ff : 10'b0;
assign vga_b = inBox ? 10'h3ff : 10'b0;

endmodule

```

This box is a very simple way to represent a '1'. Write modules for each of the numbers 0-9, either built out of blocks or using more complicated and pretty tests if you wish. Write code at the top level that uses the switches `SW[2:0]` to select which character is shown on the screen.

### Extra Time

If you have spare time, see whether you can get two-digit numbers to display correctly.