Appendix D – VGA Interface

A. Bubble Sort Implementation

To display the performance of MIPS processor we used two tools i.e. VGA interface and seven segments. Figure 1 to Figure 3 shows the schematic of these two interfaces which connects to MIPS processor.

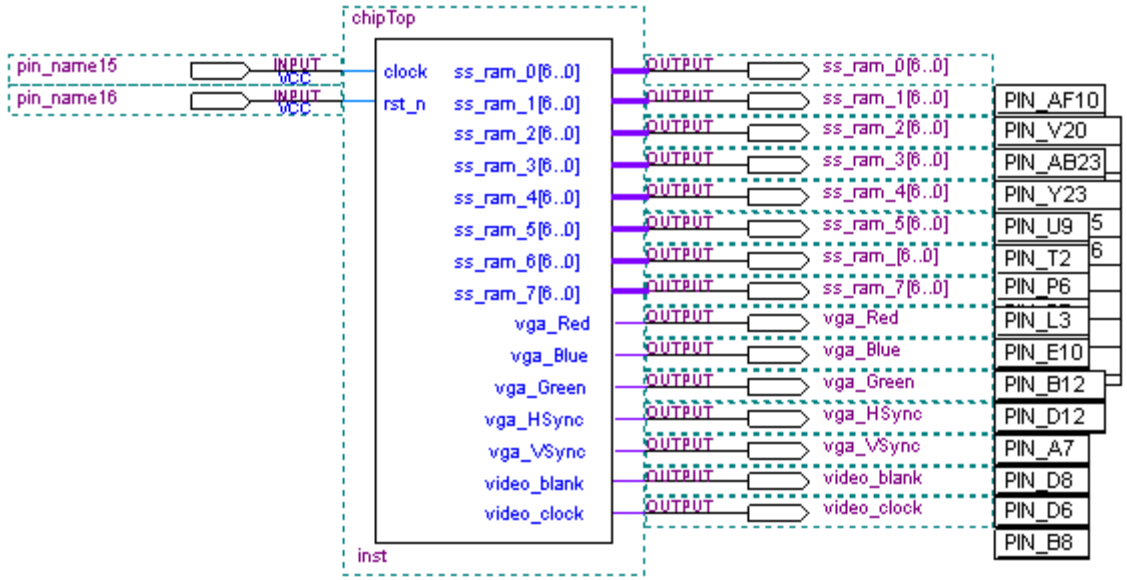

Figure 1 Top level of VGA and Seven Segment Interface

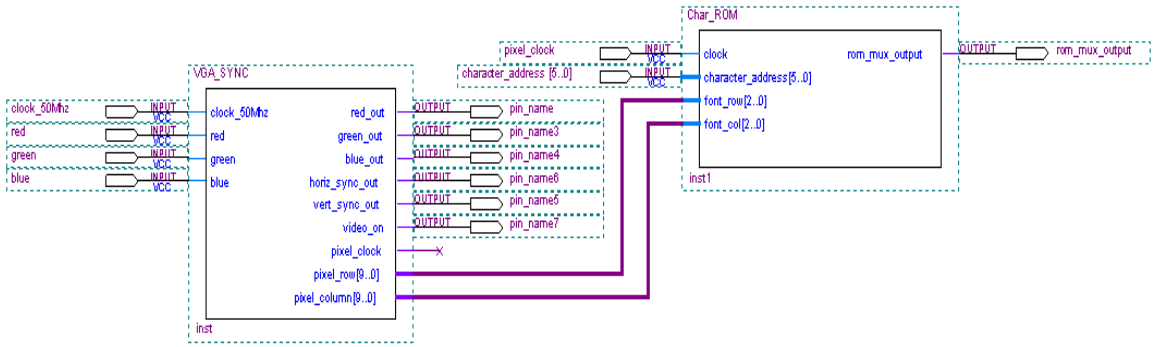

Figure 2 Internal modules of vgaCharacter.v

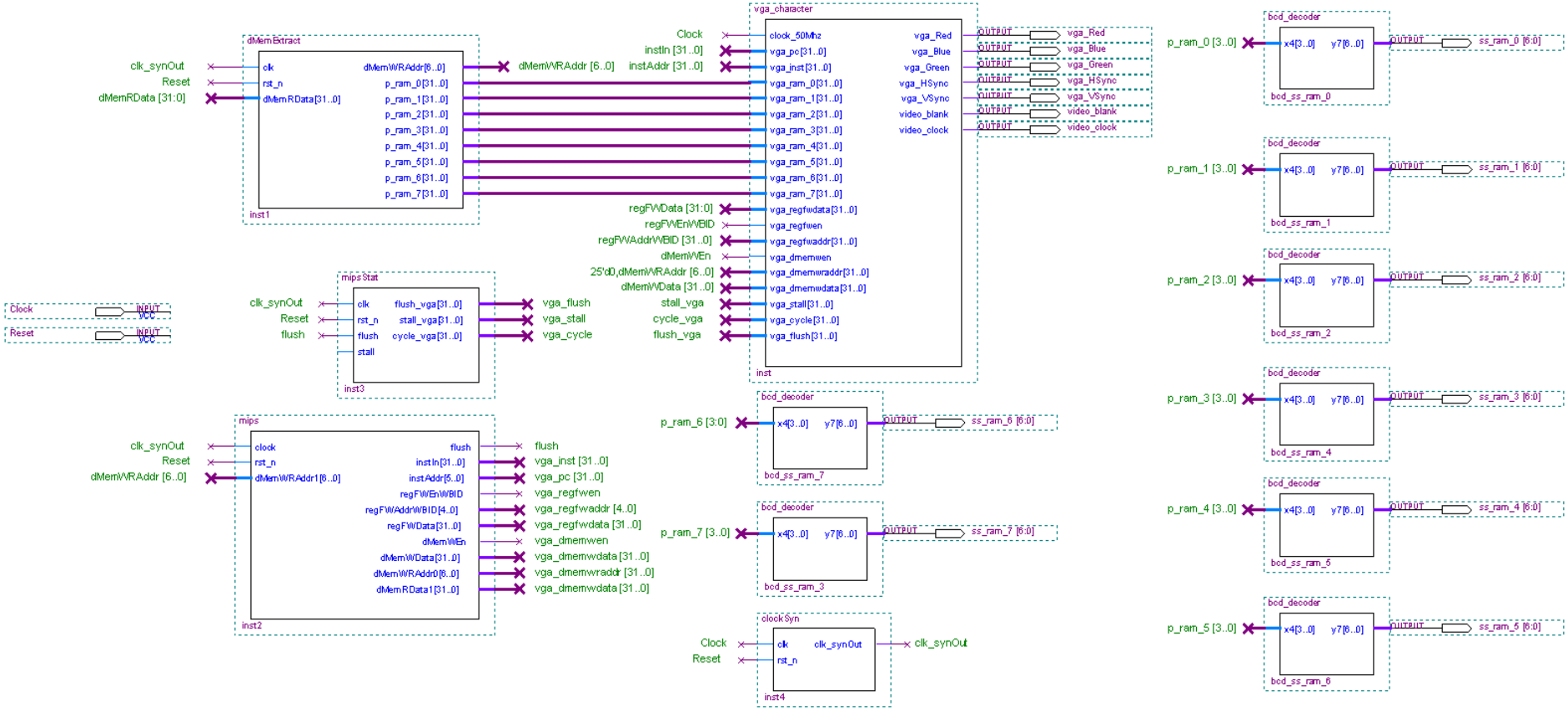

Figure 3 Internal Module of chipTop.v for VGA and Seven Segments Interface

The data from several addresses in data memory which relates to the bubble sort program is shown by seven segments on FPGA board, and VGA Monitor. The data is extracted from data memory by *dMemExtract* module. For sevens segments interface, the first four bits of data from *dMemExtract* are decoded by *bcd_decoder* module. Then from *bcd_decoder*, the data are connected to seven segment displays on board. For VGA Monitor, the data that have been extracted by *dMemExtract* is sent to *vga_character*. Besides of the data from data memory, the VGA Monitor displays other important data such as total stall event, total flush event, data to be written to register file, data memory write enable, data to be written to data memory, and total clock cycle.

Vga_character module consists of two main modules i.e. *VGA_SYNC* and *CHAR_ROM*. *VGA_SYNC* is used to generate the timing signals needed for VGA video display. The *VGA_SYNC* generates the horizontal and vertical sync signals, by using 10-bit counters, *H_count* for the horizontal count and *V_count* for the vertical count. *H_count* and *V_count* generate a pixel row and column addresses that are used by other modules. The VGA video display uses clock with frequency 25 MHz, while the system uses 50 MHz. The 50 MHz clock is divided by *Video_PLL* module. Counters are used to produce video synchronization timing signals like those seen in Figure 4 and Figure 5.

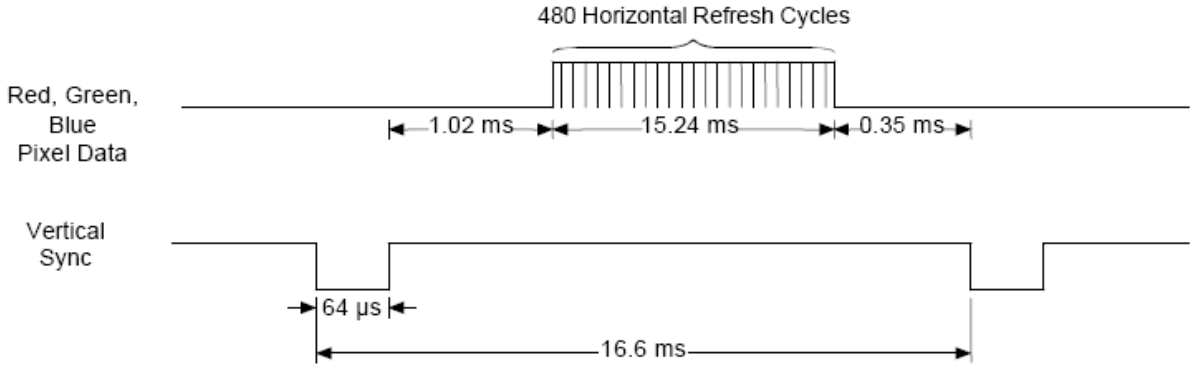

Figure 4 Vertical Synchronization Signal Timing for 640 by 480 at 60Hz

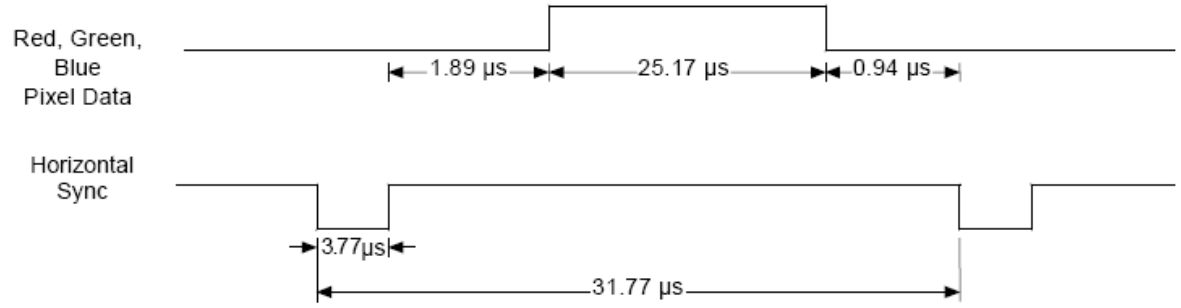

Figure 5 Horizontal Synchronization Signal Timing for 640 by 480 at 60Hz

The character displayed on VGA Monitor is initialized by ROM called *CHAR_ROM*. Each letter, number, or symbol is a pixel image from the 8 by 8 character font. To make the character larger, each dot in font maps to a 2 by 2 pixel block the row and column counter by 2. The row and column counters provide inputs to circuits that address the character font ROM and determine the color of each pixel.

Here is an example implementation of character font used in *CHAR_ROM*. To display an "A" the character ROM would contain only the starting address 000001 for the font table for "A". The 8 by 8 font in the character generation ROM would generate the letter "A" using the following eight memory words: [2]

Address	Font Data
000001000 :	*00011000* ;
000001001 :	*00111100* ;
000001010 :	*01100110* ;
000001011 :	*01111110* ;
000001100 :	*01100110* ;
000001101 :	*01100110* ;
000001110 :	*01100110* ;
000001111 :	*00000000* ;

Figure 6 Font Memory Data for the Character "A"

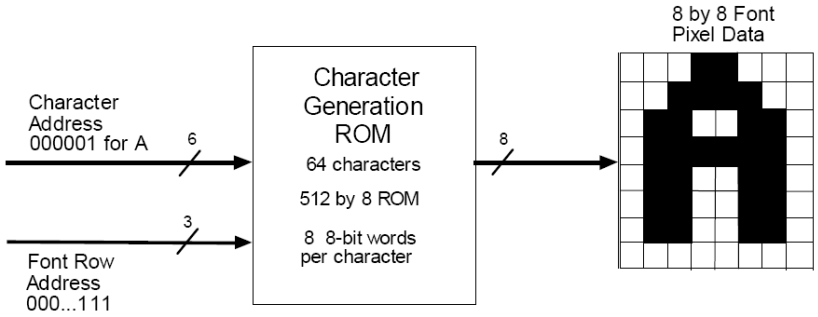

Figure 7 Accessing a Character Font Using a ROM

The column counters are used to select each font bit from left to right in each word of font memory as the video signal moves across a row. This value is used to drive the logic for the RGB signal so that a "0" front bit has a different color from a "1". Using the low three character font row address bits, the row address bits, the row counter would select the next memory location from the character font ROM when the display moves to the next row.

A 3-bit font column address can be used with a multiplexer to select the appropriate bit from the ROM output word to drive the RGB pixel color data. Both the character font ROM and the multiplexer are contained in *vga_character* module. Here is the base address of each character at *CHAR_ROM* module [2].

CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS
@	*00*	P	*20*	Space	*40*	0	*60*
A	*01*	Q	*21*	!	*41*	1	*61*
B	*02*	R	*22*	"	*42*	2	*62*
C	*03*	S	*23*	#	*43*	3	*63*
D	*04*	T	*24*	$	*44*	4	*64*
E	*05*	U	*25*	%	*45*	5	*65*
F	*06*	V	*26*	&	*46*	6	*66*
G	*07*	W	*27*	'	*47*	7	*67*
H	*10*	X	*30*	(	*50*	8	*70*
I	*11*	Y	*31*	)	*51*	9	*71*
J	*12*	Z	*32*	*	*52*	A	*72*
K	*13*	[	*33*	+	*53*	B	*73*
L	*14*	Dn Arrow	*34*	,	*54*	C	*74*
M	*15*	]	*35*	-	*55*	D	*75*
N	*16*	Up Arrow	*36*	.	*56*	E	*76*
O	*17*	Lft Arrow	*37*	/	*57*	F	*77*

Figure 8 Character Address Map for 8 by 8 Font ROM

B. Sokoban Implementation

The VGA Game interface main function is to display the state of the game environment, which is located in the data memory. The address of data memory to be read is generated by *VGA_Logic* module. *VGA_Logic* determine the data memory address according to the location of the pixel. This pixel location is given from *VGA_Sync* module. From this state data, *VGA_Logic* generates the *character_address* input of character ROM. The ROM then sends 4-bit RGB data to *VGA_Sync*, which renders these video signals.

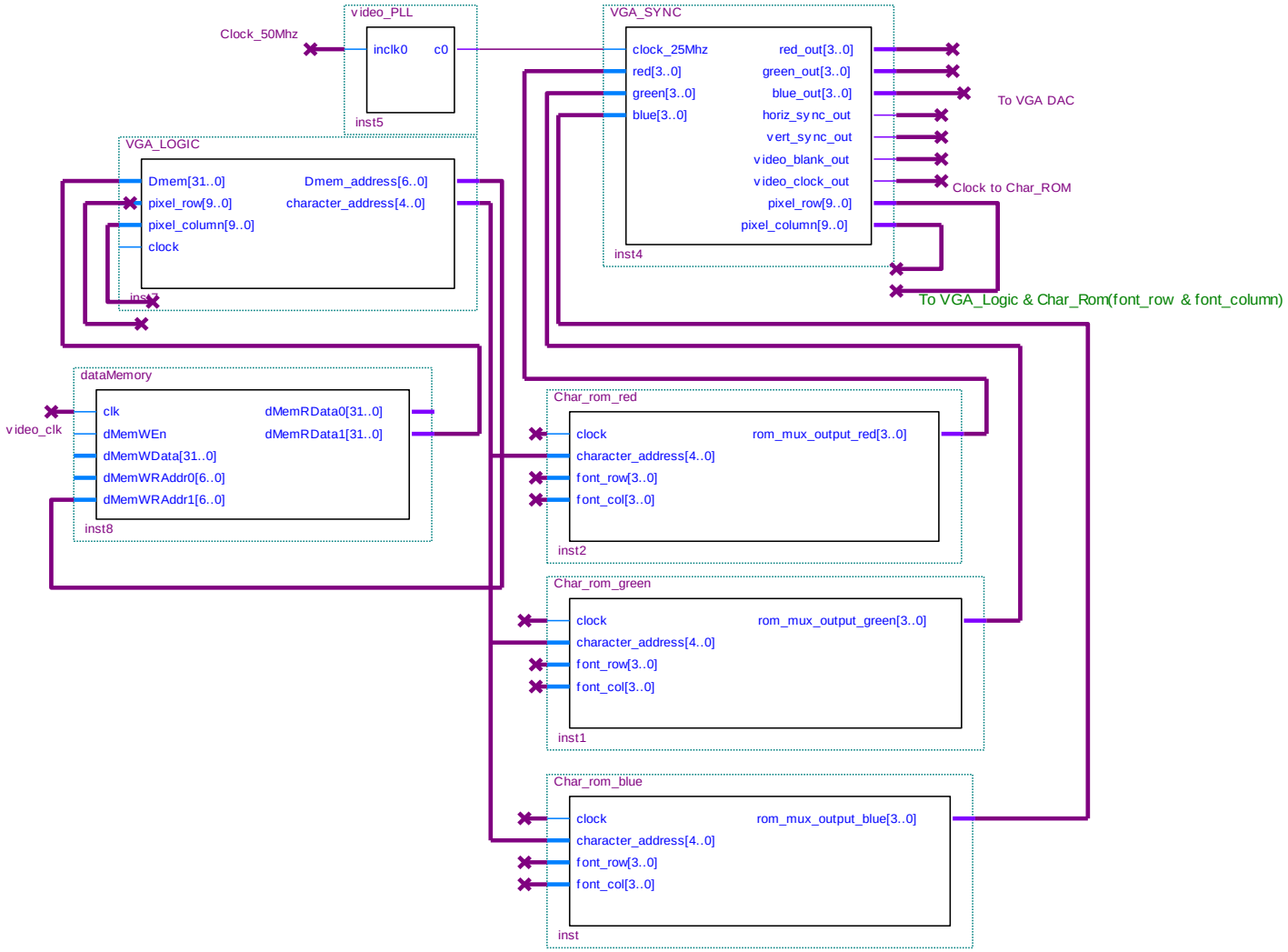

Figure 9 VGA Interface for Sokoban Implementation

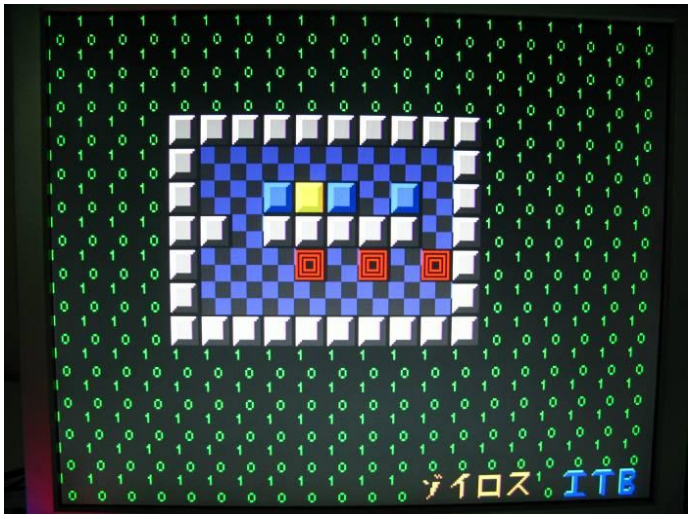

Figure 10 Sokoban Initial Game Display