

Tallinn University of Technology  
Dept. of Computer Engineering  
Chair of Digital Systems Design

# Design of a Video Game

Project in „IAY0070 HW/SW Co-design”

Anti Sullin  
020633 IASM-21  
04.2006

Project supervisor: Kalle Tammemäe

Tallinn 2006

## 1. Assignment

The task of the lab is to implement a classic video game using Spartan3 FPGA on a Xess XSA-3S1000 board. Input device for the game is a PS/2 keyboard and output device is a VGA monitor.

## 2. Solution planning – the gaming platform

### *Memory subsystem*

XSA-3S1000 board has SDRAM memory. It is clocked as the FPGA on 100MHz. For the project, a memory controller from Xess inc is used. The controller has a 4-level pipeline. The memory is read by VGA generator, inbetween there is a 8-word FIFO buffer. Under these constraints, the memory has to be read at least one word per pixel drawn. When not taking into account the SDRAM refreshes and row, bank changing overhead, then with 640x480, 60Hz VGA (25MHz dot clock) the memory usage is about 25% and there is enough time for writing next frame as well. As the memory row is changed, the pipeline has to be flushed and so we lose 4 clock cycles. Thus it is important to write the memory only when the read FIFO is full.

The video memory is composed of two layers. The data of both layers has to be interleaved in video memory, otherwise we have to change memory row twice per pixel and we would not meet the timing restrictions.

A two-port unit is used to interface the memory to VGA generator and microcontroller code at once. This multiplexes the access from both channels. The microcontroller has always lower priority to avoid starvation in VGA generator.

### *Interfacing memory and PicoBlaze microcontroller*

To interface memory subsystem and PicoBlaze, a memory interface is implemented. The module is on the microcontroller bus and uses 6 output ports and 3 input ports for communication.

address	name	7	6	5	4	3	2	1	0
000	ADDR0	A7	A6	A5	A4	A3	A2	A1	A0
001	ADDR1	A15	A14	A13	A12	A11	A10	A9	A8
010	ADDR2	A23	A22	A21	A20	A19	A18	A17	A16
100	DATA0	D7	D6	D5	D4	D3	D2	D1	D0
101	DATA1	D15	D14	D13	D12	D11	D10	D9	D8
111	CMD							WR	RD

Table 1 – Output port map

address	name	7	6	5	4	3	2	1	0
100	DATA0	D7	D6	D5	D4	D3	D2	D1	D0
101	DATA1	D15	D14	D13	D12	D11	D10	D9	D8

111	Status						WRX	RDX	RDD
-----	--------	--	--	--	--	--	-----	-----	-----

Table 2 – Input port map

Bit definitions:

**A0-A23:** Memory word address

**D0-D15:** Memory word

**WR:** Write command

**RD:** Read command

**WRX:** Write operation pending flag (set with write operation command, cleared with earlyOpBegun signal from SDRAM)

**RDX:** Read operation pending flag

**RDD:** Read operation done flag, data in register ready (reset when ADDR0, ADDR1 or ADDR2 is written).

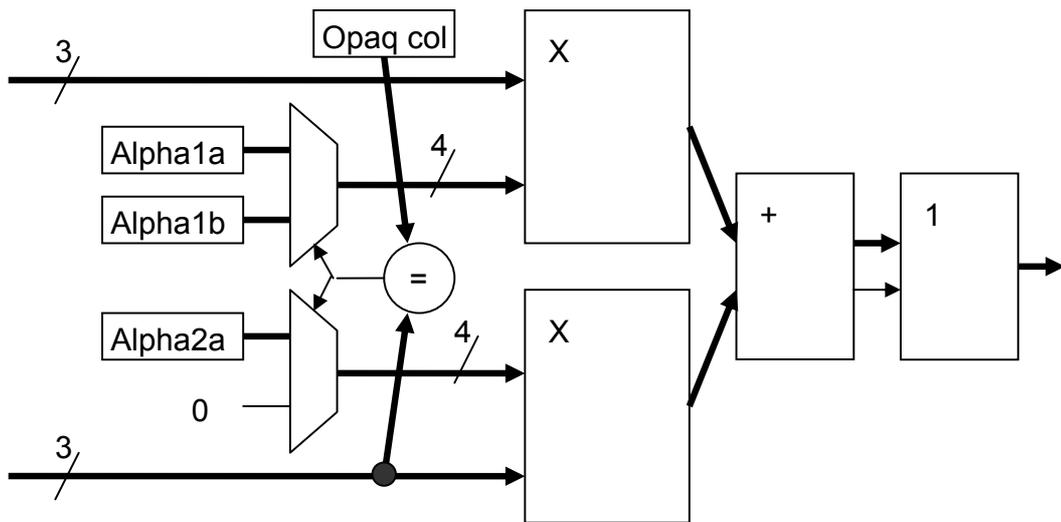
### ***Graphics subsystem***

The graphics subsystem is implemented with two layer support. This way, a background image can be loaded on one layer and the game itself can be drawn on it without any difficult and computing-intensive operations.

The graphics subsystem is based on the VGA generator demo by Xess that uses LPT for image uploading. The VGA generator is running by default at 800x600, 60Hz. This is lowered to 640x480, 60Hz to lower the dot clock to 25Hz. This is important as twice the memory bandwidth is needed for two layers.

The graphics subsystem design is shown on drawing 1. The brightness of every pixel component (R, G, B) is multiplied by a coefficient called alpha that describes the opacity of that layer. The results of both layers are added. The result is or'ed with the carry bit to allow saturation of the color component if sum of both parts is higher than the maximum value.

Additionally, a opaque color can be defined on the top layer to allow transparent areas on the top image. For that, the alpha coefficients are switched by multiplexers. The multiplexers are controller by a comparator, that verifies if the top layer color is equal to the opaque color.



**Figure 1 – The graphics subsystem**

As the BMP image format is very close to raw image data and it is relatively easy to create with any graphics editing software, the image data is read from the BMP format. For that, the start address has to be increased to skip the BMP header. The BMP image has to be crafted with the right color map so that the eight bits of the color table index correspond to the eight output bits of the VGA generator.

To create a image in the required format, photo editing software Jasc Paint Shop Pro is used. It is capable of converting an image to a given color map. This is done with dithering to compensate the low number of actual colors available. The image is more dotted but the colors are much better and there are no large areas of the same color with very disturbing edges. After the conversion, the image has to be rotated upside down as BMP format is written backwards. The color table is stored in the BMP format as well but is not read in this given application.

The two-layer system works in one of two modes. If the color on the second layer is not equal to the pre-defined opaque color, the first layer is displayed with opacity alpha1a and the second layer with opacity alpha2a. If the color on the second layer is equal to the opaque color, the first layer is displayed with opacity alpha1b and the second layer is not displayed.

Alpha1a, 1b, 2a are 4-bit variables, which value 0b1000 means opacity 100% and 0b0000 is 0%.

### **Test images**

Settings used on test images:

OpaqueColor="11111111" (white)

Alpha1a="0101" (on non-opaque spot, the opacity of first layer is 62,5%)

Alpha1b="1000" (On opaque spot, the first layer is totally visible)  
Alpha2a="0011" (on non-opaque spot, the opacity of the second layer is 37,5%)



Figure 2 – First test image



Figure 3 – Second test image



Figure 4 – Result of both images stacked

### **Keyboard PS/2 interface**

An PS/2 keyboard interface written for "Advanced Digital Design I" course is used. It keeps track of the state of four keys:

- Up arrow (E0 72)
- Down arrow (E0 75)
- Left arrow (E0 6B)
- Right arrow (E0 74)

### **PicoBlaze Interface**

Only the four LSB address bits are used on the PicoBlaze port.

Port\_ID(3) = 0: memory interface

Port\_ID(3) = 1: VGA and PS/2 interface

Address	R	W
<b>0000</b>	-	ADDR0
<b>0001</b>	-	ADDR1
<b>0010</b>	-	ADDR2
<b>0100</b>	DATA0	DATA0
<b>0101</b>	DATA1	DATA1
<b>0111</b>	STATUS	COMMAND
<b>1000</b>	-	OPAQUECOLOR
<b>1001</b>	-	ALPHA1A
<b>1010</b>		ALPHA1B
<b>1011</b>		ALPHA2A
<b>1100</b>	KEYBOARD	-

**Table 3 - PicoBlaze I/O ports**

In addition to the I/O ports, an interrupt is used. The interrupt synchronizes the game with VGA vertical synchronization. At the end of every frame, an interrupt is given; later an interrupt acknowledge signal is returned.

### 3. Implementation

#### **Hardware**

A board with 3S1000 FPGA is used (XSA-3S Board, V1.0, ©2004). The custom hardware is written in VHDL according to the given specification. For implementation, testing and synthesizing, Xilinx ISE WebPACK'i, V8.1i is used.

#### **Software – the snake game**

A classic snake game demo is implemented on this platform. The two-layer graphics subsystem first layer holds the background image and the second layer the game itself. 0xFF is selected as the opaque color, the snake and the game field is drawn with the color 0x00 (black).

At the start, the opacity of both layers is set to zero, thus clearing the screen.

After that, the second layer is overwritten with 0xFF and the borders of the game field are drawn. Now the opacity of the first layer is increased step-by-step to its maximal value, creating a fade effect. Then the game field is faded to 37.5%.

The game is synchronized with 60Hz VGA vertical synchronization signal. To set the speed of the game, a number of interrupts are ignored before one is serviced. Keyboard is read on every interrupt to find a keypress as soon as possible.

The snake head is positioned with its x and coordinates. The screen is divided into 4x4 pixel blocks, where a 3x3 pixel cube forms one part of the snakes body.

The position of the block in video memory is calculated as:

$Addr = base + (640 * 4 * y) + (4 * x)$ , which is transformed to:

$Addr = base + y \ll 9 + y \ll 11 + x \ll 2$ .

A circular buffer is used to keep track of the snakes tail. For this, 65536 memory words are used in SDRAM (addresses 0x100000 – 0x10FFFF). Every 16-bit memory word holds an 8-bit x and an 8-bit y coordinate. Two 8-bit registers are used for pointers of the circular buffer.



**Figure 5 - Game screen**

One more register is used to hold number of pieces the snake has to grow. This is set to a given number at the beginning and decremented if it is non-zero. If this register is zero, the tail of the snake is removed. Later, this register could be used to easily add some kind of game functionality.

## 4. Difficulties

A number of difficulties had to be overcome for this project:

- The SDRAM memory pipeline needs to be flushed to change the memory row. The memory row has to be changed quite actively on normal operation. The first experiment did not have the memory areas of the both layers interleaved and so it did not meet the timing requirements. Still, the non-displayable periods of the VGA interface sufficed to show a quarter of the image correctly and this was confusing.
- The SDRAM and PicoBlaze interface was quite timing-critical and it is easy to get signal race on so high frequencies. Some designs, for example, worked every other VHDL synthesis, even there were no changes in these units at all.
- For some reason, the XSLOAD is not capable of loading a regular binary file into SDRAM. Thus a converted had to be implemented to convert binary files into XES HEX files.

## **5. Conclusion**

The design of a video game was quite a challenging process. It demanded knowledge from many different fields – discrete mathematics, Boolean logic, arithmetics, digital hardware, different programming languages etc. The work was quite time-consuming, it was done on a couple of weeks on evenings and some weekends as well. Later, after getting everything to work, I value the experience gained and I feel that this kind of creative work and the independent problem solving skills are very important.