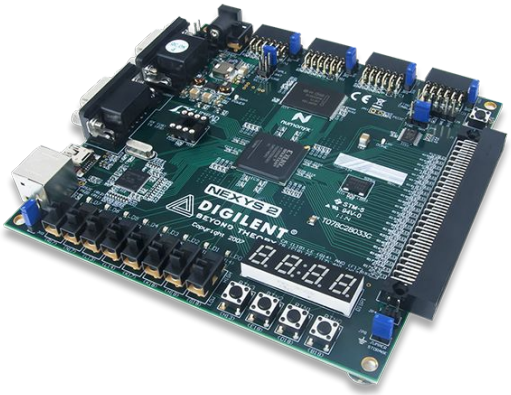
MIPS COMPUTER ON FPGA

Supervisor: Professor Layla Abou Hadid
Team: Mostafa Abd El-Aziz

Overview

- The target of the project is to build a general-purpose computer system on FPGA.
- General-purpose computer?
- FPGA: Spartan-3E chip on Nexys II development board.

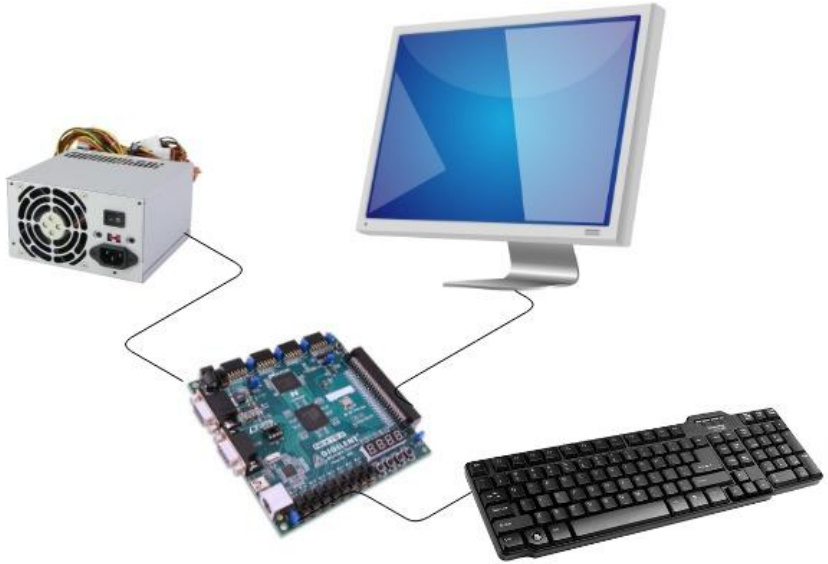


WHY?

- Argument: There are tons of MIPS processors implemented in VHDL and Verilog!
- Fallacy: *MIPS processors*... not *MIPS computers*.
- Research the capabilities of FPGAs vs. capabilities of modern micro-computers.
- Computer architecture education in CSED (That's why we chose MIPS).
- Processor-oriented education vs. system-oriented education (Clements vs. Peterson).
- Fun.

Goal

- General-purpose computer:
  - Capabilities?
  - Limitations?
  - Users: Gamers - Students - Hobbyists
- Processor: MIPS-I ISA (pipeline, cache, and paging).
- Interfaces: VGA (output) and PS/2 (input).
- Hardware: MIPS processor + I/O interfaces in VHDL.
- Software: Firmware and OS [Quafios].



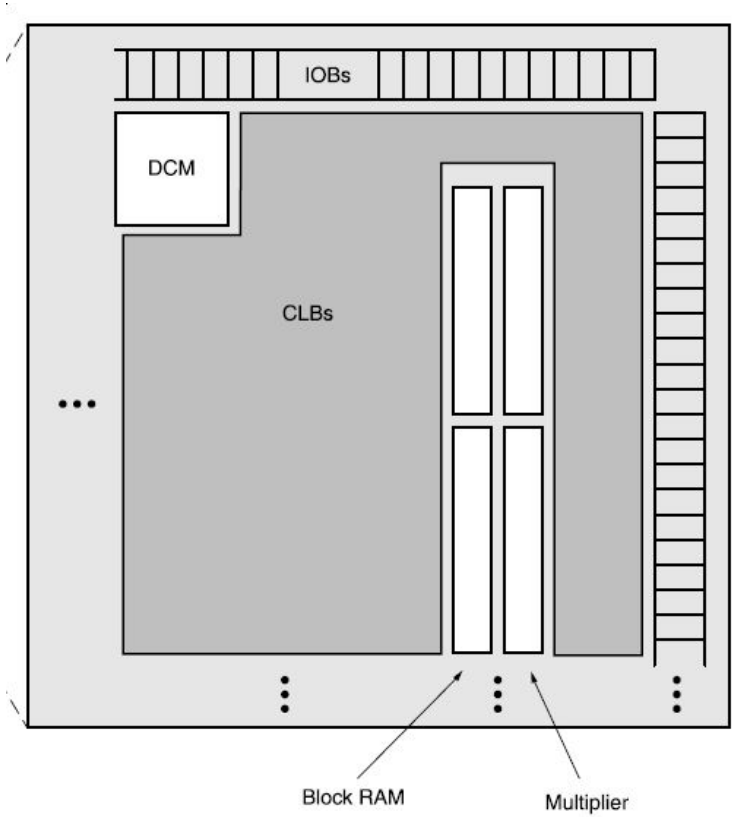
Outline

- High-Level Architecture
- System components:
  - Processor
  - Memory
  - VGA
  - PS/2 Keyboard
  - Interrupt Subsystem
  - Timer
- Emulation
- Software
  - Firmware
  - OS
- Conclusion

Outline

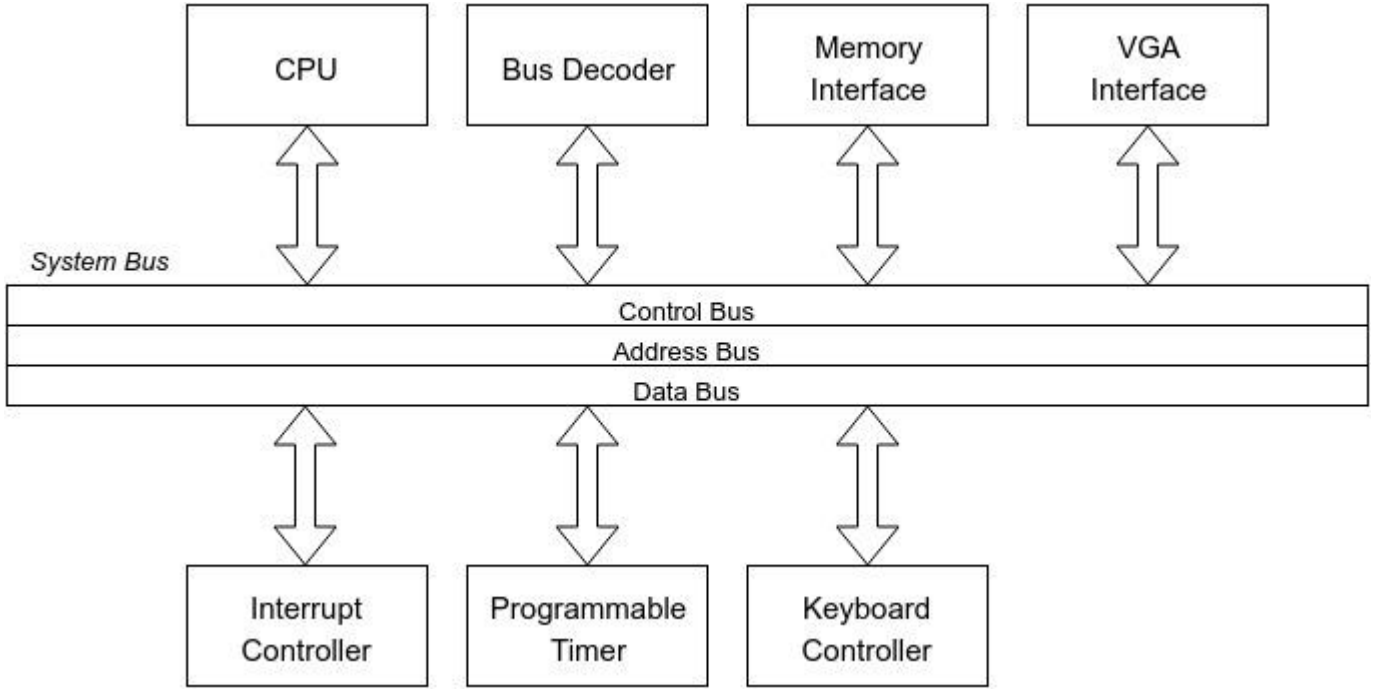
- High-Level Architecture
- System components:
  - Processor
  - Memory
  - VGA
  - PS/2 Keyboard
  - Interrupt Subsystem
  - Timer
- Emulation
- Software
  - Firmware
  - OS
- Conclusion

System Architecture

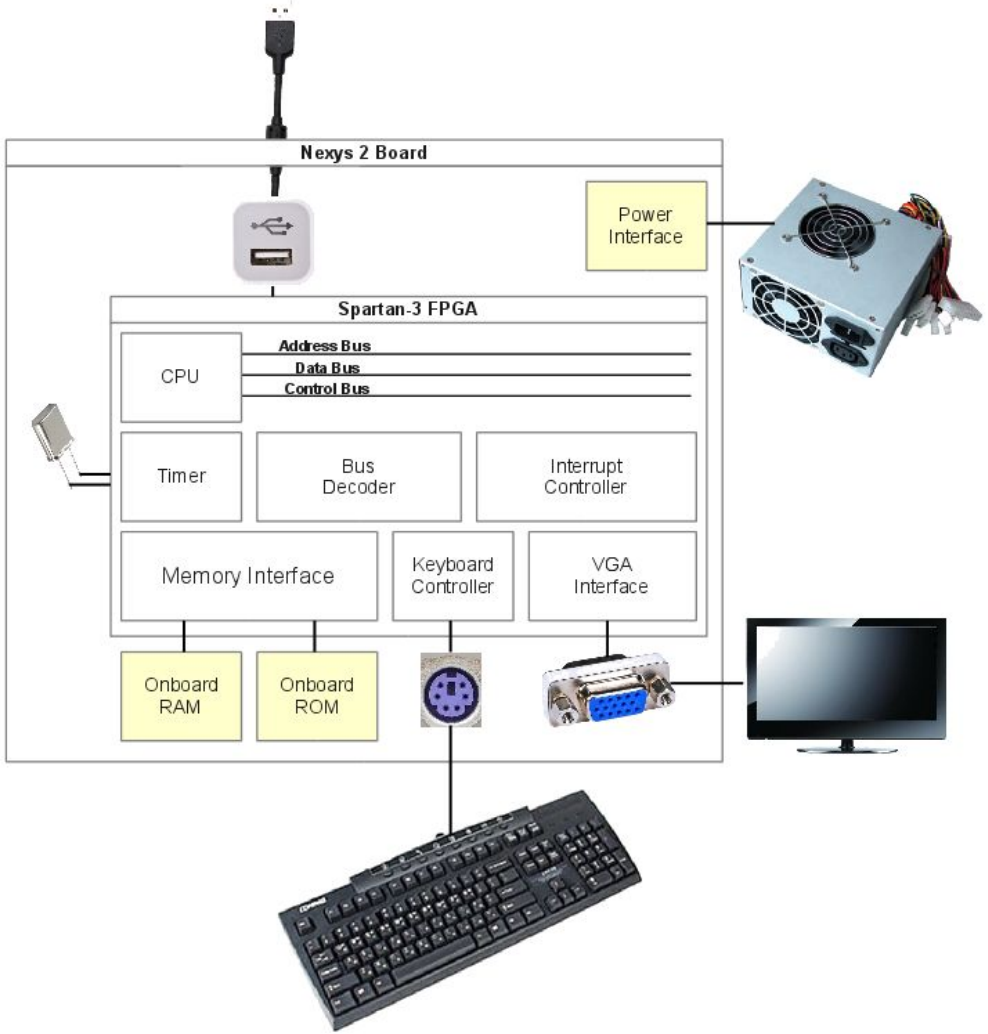


- Challenges:
  - Low clock rate.
  - LUTs, multiplexers, adders, etc...
  - Block RAMs: only 20 slices.
  - Limited distributed RAM.
  - Timing constraints!
- These limited resources led to various design decisions:
  - **The choice of MIPS architecture (RISC merits).**
  - Direct-mapped TLB and caches.
  - Text-mode VGA.
  - Precise exceptions.

System Architecture: High Level



System Architecture: High Level



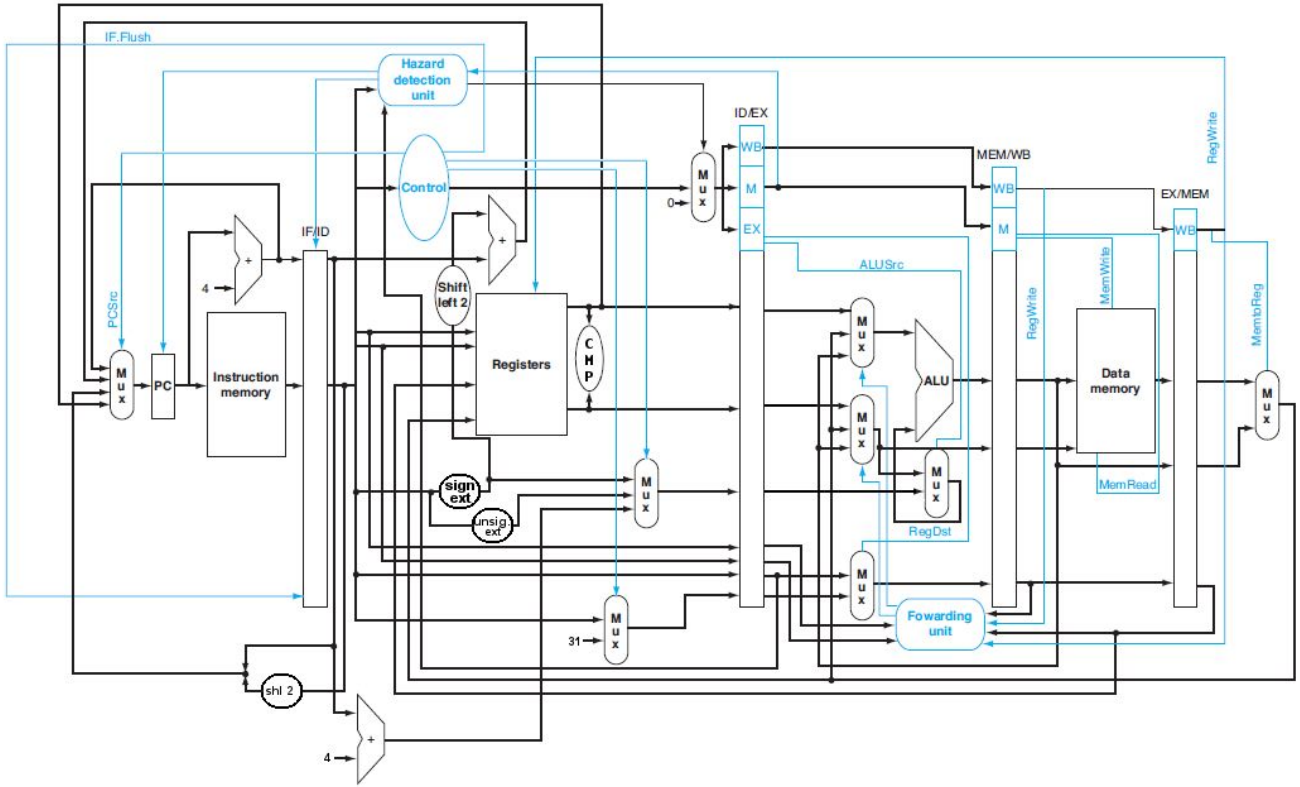
Outline

- High-Level Architecture
- System components:
  - Processor
  - Memory
  - VGA
  - PS/2 Keyboard
  - Timer
  - Interrupt Controller
- Emulation
- Software
  - Firmware
  - OS
- Conclusion

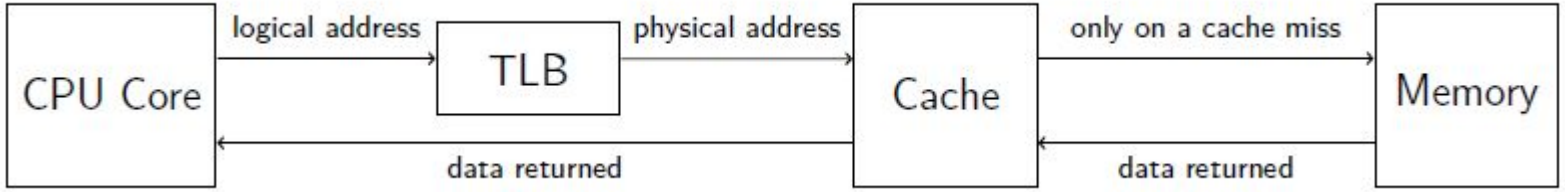
Central Processing Unit (1/6)

- The CPU consists of three components:
  - The core (pipeline).
  - Caching sub-system: instruction cache and data cache.
  - Software-controlled translation-lookaside buffer (TLB).
- The CPU is actually an implementation of MIPS-I instruction set architecture:
  - 58 instructions: memory access, ALU, branch/jump, and misc instructions.
  - 32 general purpose registers + HI and LO registers.
- Additions (compatible with IDT's R3000):
  - Control instructions: MFC0, MTC0, TLBR, TLBWI, and RFE.
  - Control registers: Index, EntryLo, BadVaddr, EntryHi, SR, Cause, and EPC.
  - Transparent cache.
  - Software-controlled TLB.

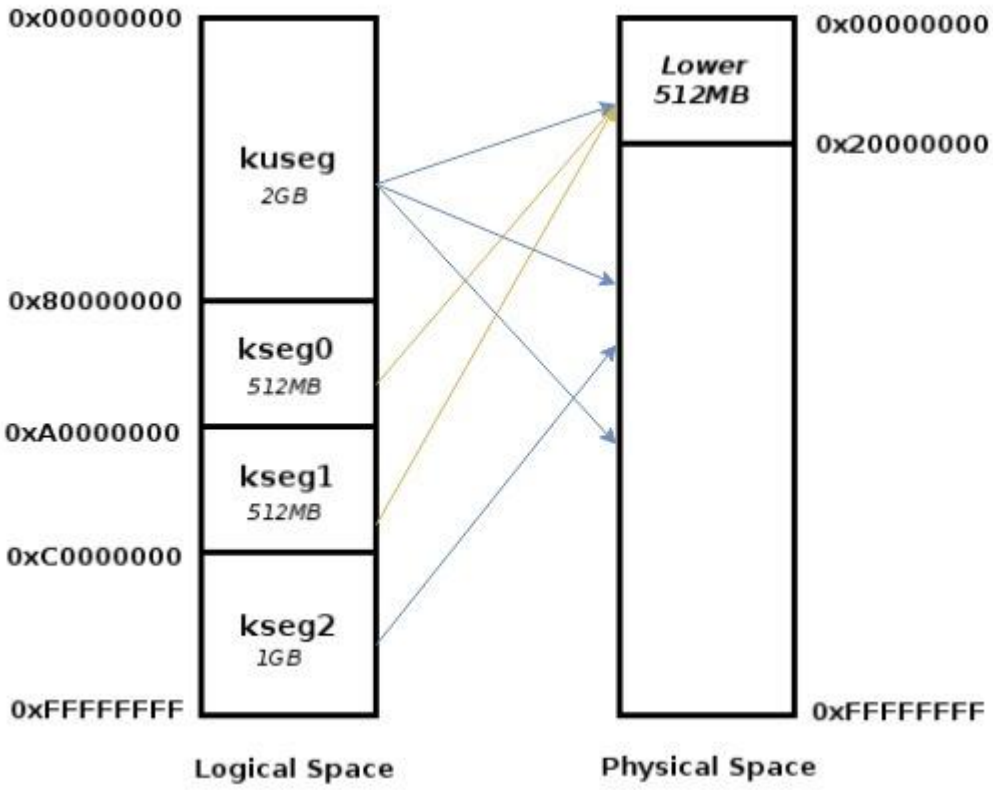
Pipeline


IF.Flush
Hazard detection unit
ID/EX
WB
M
EX
MEM/WB
WB
M
EX/MEM
WB
RegWrite
Control
0
M u x
IF/ID
PCSrc
4
+
Shift left 2
Registers
C H P
ALUSrc
M u x
ALU
M u x
M u x
M u x
RegWrite
MemWrite
Data memory
MemRead
MemtoReg
M u x
PC
Instruction memory
sign ext
unsig ext
M u x
RegDst
M u x
Fowarding unit
31
M u x
shl 2
4
+

Address Translation



Address Translation



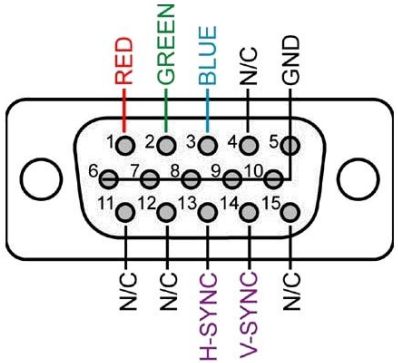
Memory (2/6)

- MIPS-I physical address space is 4GB.
- Nexys II board contains two memory chips:
  - 16MB Pseudo-static RAM chip.
  - 16MB ROM chip.
- VHDL interface between the chips and system bus.
- Memory-mapped I/O

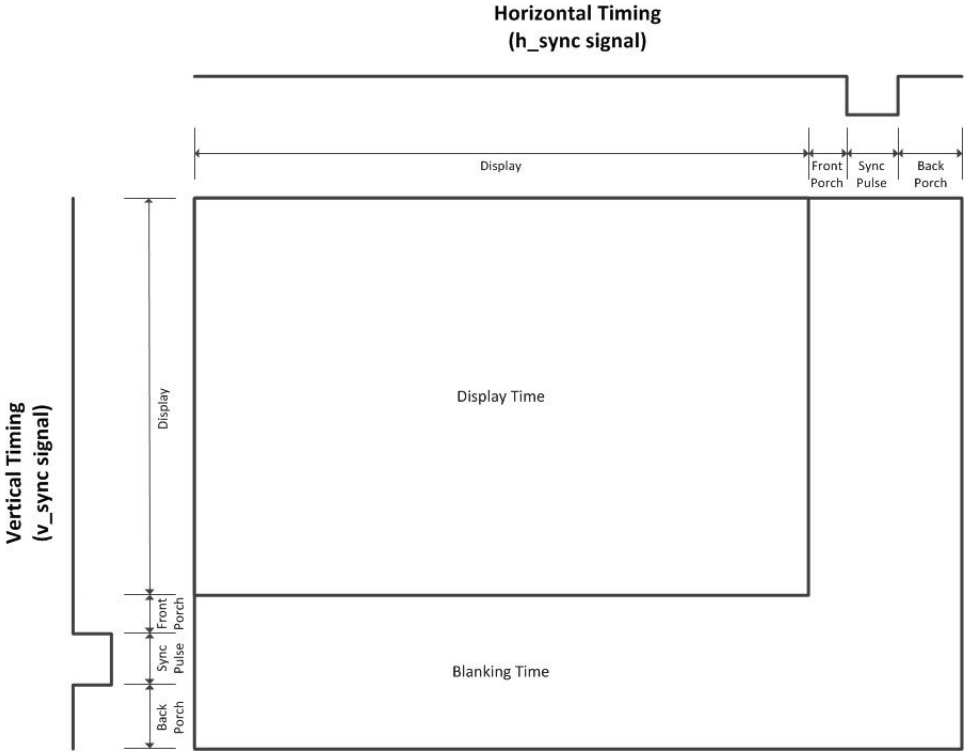
Start Address	End Address	Size	Device
0x00000000	0x00FFFFFF	16MB	RAM
0x1E000000	0x1E003FFF	16KB	VGA
0x1E800000	0x1E800FFF	4KB	KBD (Keyboard Controller)
0x1E801000	0x1E801FFF	4KB	PIT (Programmable Interval Timer)
0x1E802000	0x1E802FFF	4KB	PIC (Programmable Interrupt Controller)
0x1F000000	0x1FFFFFFF	16MB	ROM

VGA Interface (3/6)

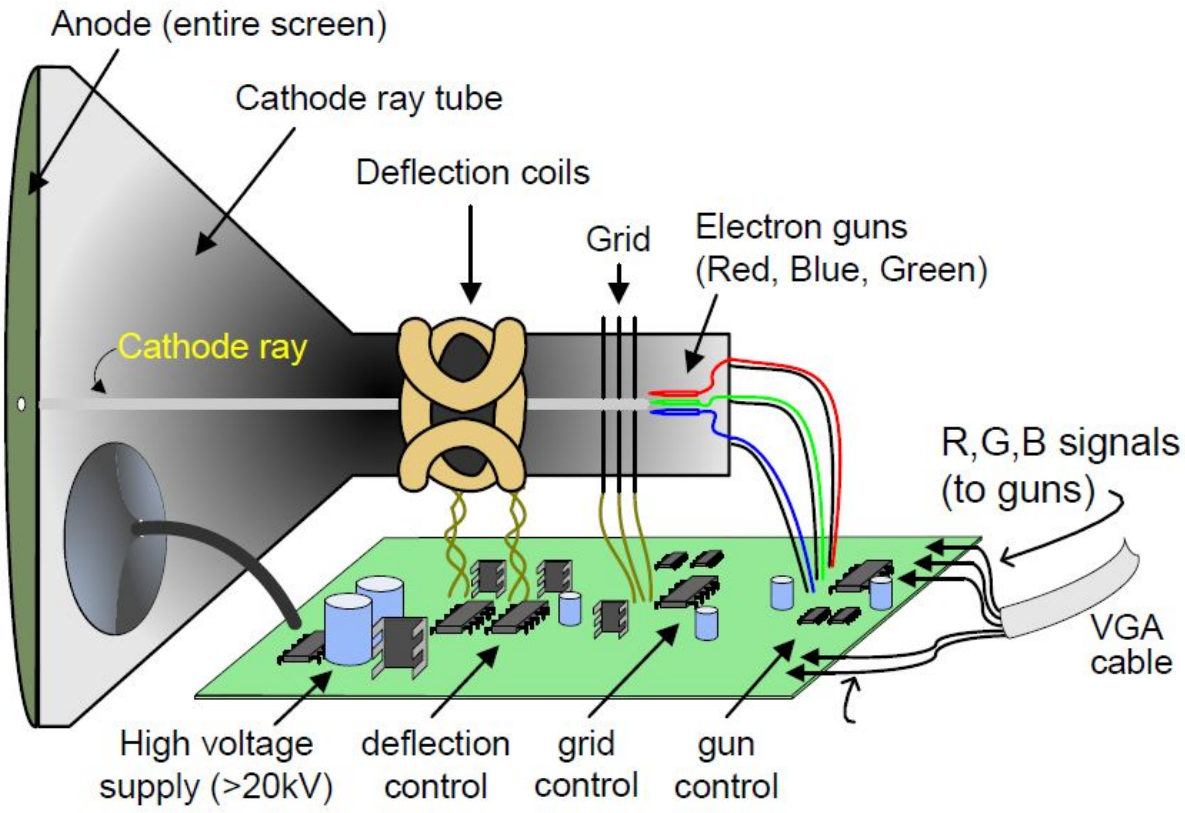
- The image on screen is constructed in rectangular pattern (raster scan).
- Five signals control the display: HS, VS, R, G, and B.



A masculine DE15 VGA socket

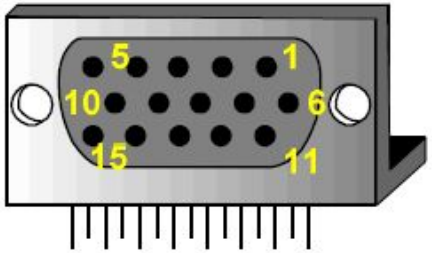


VGA Interface: CRT Screen



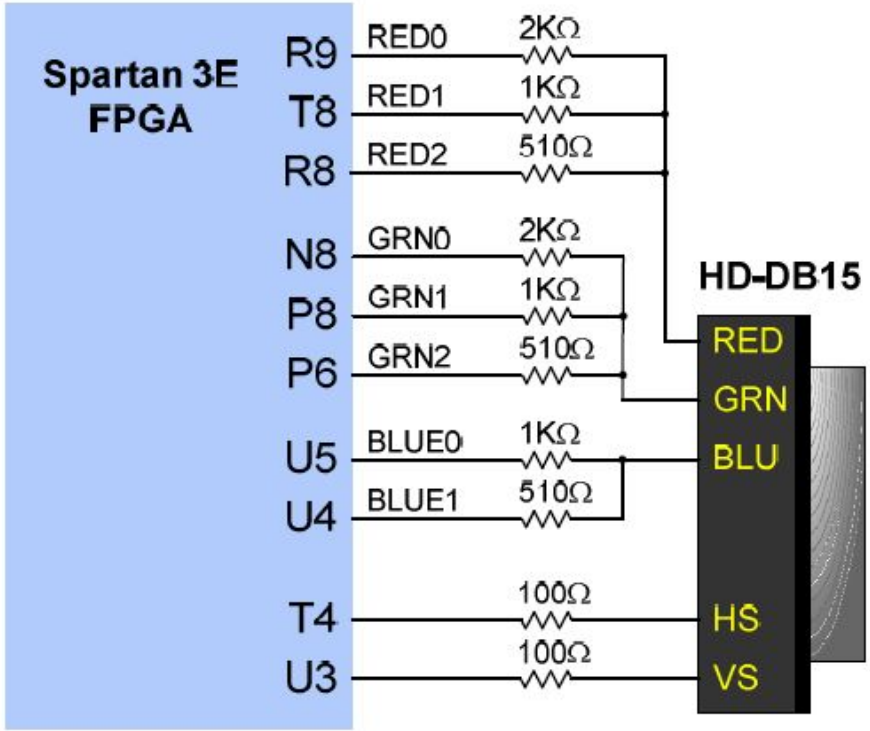
VGA Interface

- Nexys II board provides a DE15 VGA port. HS and VS signals are directly connected to the Spartan 3E FPGA chip.
- Since R, G, and B are analog singnals, a DAC circuit is put between FPGA chip and VGA port.
- FPGA could produce 8 levels of red color, 8 levels of green, and 4 levels of blue (total 256 colors).



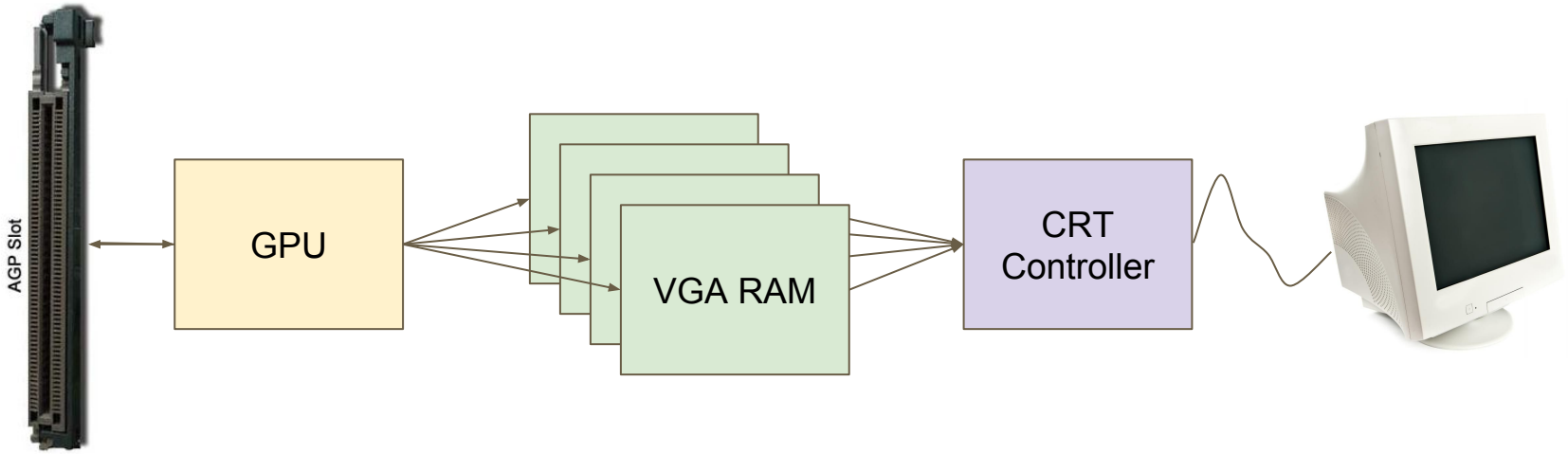
Pin 1: Red
Pin 2: Grn
Pin 3: Blue
Pin 13: HS
Pin 14: VS

Pin 5: GND
Pin 6: Red GND
Pin 7: Grn GND
Pin 8: Blu GND
Pin 10: Sync GND

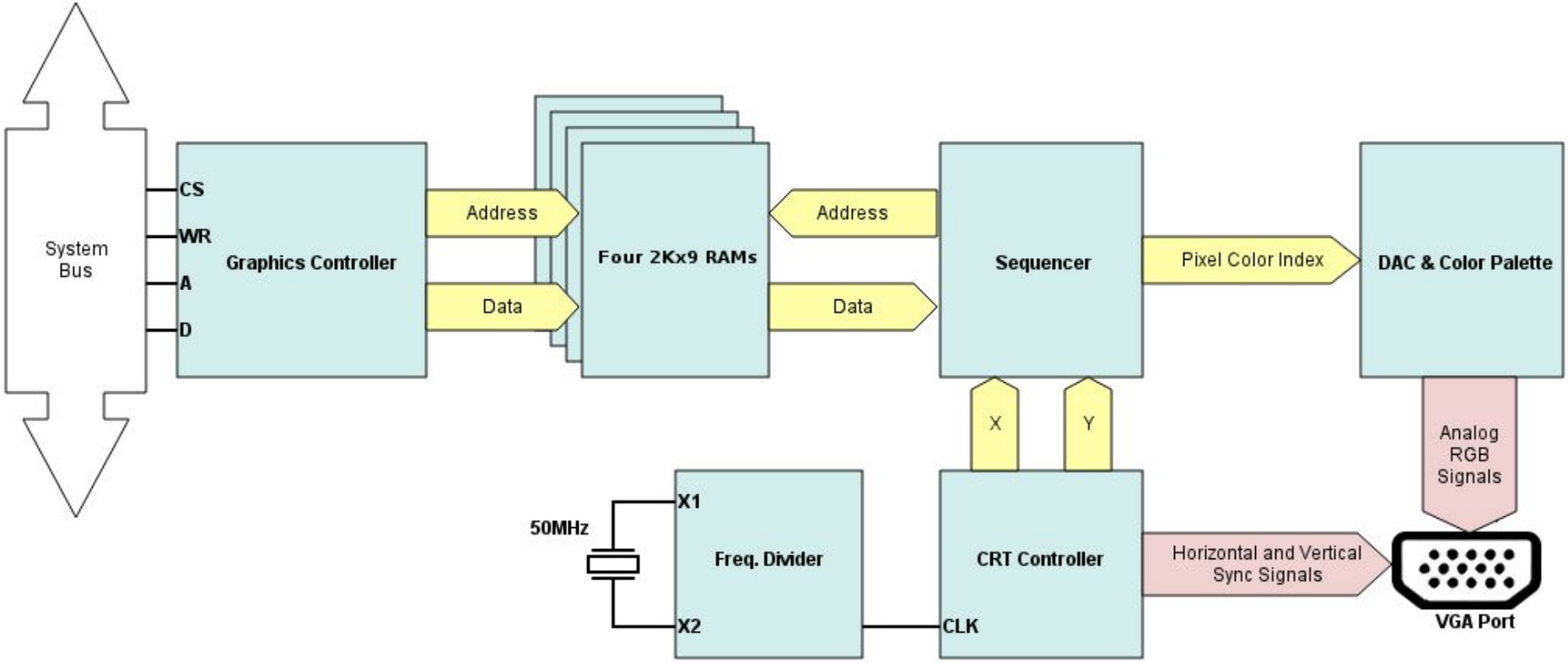


VGA Interface

- Our task is to create an interface between VGA pins (the IOBs on FPGA board) and system bus.
- Analogous to the PC:



VGA Interface



VGA Interface

Row/Col	0	1	2	3	4	5	6	7	8	9	10	11	12	...	79
0	H	e	l	l	o		W	o	r	l	d	!			
1															
2															
3															
4															
...															
24															

- Resolution: 720x400.
- Framebuffer size → VGA supports text mode only.
- 80 columns, 25 rows. Cell size: 9x16.
- Framebuffer consists of four 2Kx9 memories:
  - Character memory (bank 0)
  - Attribute memory (bank 1)
  - Font memory (bank 2 and 3).

GTKW

File Edit Search Time Markers View Help

From: 0 sec To: 2613700 ns

SST

tlc_sim

u : tlc(structural)

Signals

Time

addr[23:0] =

data[15:0] =

rom_cs =

Waves

100 ns 200 ns 300 ns 400

uuuuuu C00000 C00002

0000 0000

build : make - Konsole

File Edit View Bookmarks Settings Help

```
** Note: VHPI printf at 23840ns: ROM RD ADDR: 0x00C00088, DATA: 0x00000000
** Note: VHPI printf at 23920ns: ROM RD ADDR: 0x00C0008A, DATA: 0x00000000
** Note: VHPI printf at 23960ns: ROM RD ADDR: 0x00C0008A, DATA: 0x00000000
** Note: VHPI printf at 24000ns: ROM RD ADDR: 0x00C0008A, DATA: 0x00000000
** Note: VHPI printf at 24760ns: ROM RD ADDR: 0x00C0008C, DATA: 0x00000000
** Note: VHPI printf at 24800ns: ROM RD ADDR: 0x00C0008C, DATA: 0x00000000
** Note: VHPI printf at 24840ns: ROM RD ADDR: 0x00C0008C, DATA: 0x00000000
** Note: VHPI printf at 24920ns: ROM RD ADDR: 0x00C0008E, DATA: 0x00000000
** Note: VHPI printf at 24960ns: ROM RD ADDR: 0x00C0008E, DATA: 0x00000000
** Note: VHPI printf at 25000ns: ROM RD ADDR: 0x00C0008E, DATA: 0x00000000
** Note: VHPI printf at 25240ns: ROM RD ADDR: 0x00C00090, DATA: 0x00000000
** Note: VHPI printf at 25280ns: ROM RD ADDR: 0x00C00090, DATA: 0x00000000
** Note: VHPI printf at 25320ns: ROM RD ADDR: 0x00C00090, DATA: 0x00000000
** Note: VHPI printf at 25400ns: ROM RD ADDR: 0x00C00092, DATA: 0x00000000
** Note: VHPI printf at 25440ns: ROM RD ADDR: 0x00C00092, DATA: 0x00000000
** Note: VHPI printf at 25480ns: ROM RD ADDR: 0x00C00092, DATA: 0x00000000
** Note: VHPI printf at 26240ns: ROM RD ADDR: 0x00C00094, DATA: 0x00000025
** Note: VHPI printf at 26280ns: ROM RD ADDR: 0x00C00094, DATA: 0x00000025
** Note: VHPI printf at 26320ns: ROM RD ADDR: 0x00C00094, DATA: 0x00000025
** Note: VHPI printf at 26400ns: ROM RD ADDR: 0x00C00096, DATA: 0x00000800
** Note: VHPI printf at 26440ns: ROM RD ADDR: 0x00C00096, DATA: 0x00000800
** Note: VHPI printf at 26480ns: ROM RD ADDR: 0x00C00096, DATA: 0x00000800
** Note: VHPI printf at 26720ns: ROM RD ADDR: 0x00C00098, DATA: 0x00000000
** Note: VHPI printf at 26760ns: ROM RD ADDR: 0x00C00098, DATA: 0x00000000
** Note: VHPI printf at 26800ns: ROM RD ADDR: 0x00C00098, DATA: 0x00000000
** Note: VHPI printf at 26880ns: ROM RD ADDR: 0x00C0009A, DATA: 0x00000000
** Note: VHPI printf at 26920ns: ROM RD ADDR: 0x00C0009A, DATA: 0x00000000
** Note: VHPI printf at 26960ns: ROM RD ADDR: 0x00C0009A, DATA: 0x00000000
```

build : make | build : bash | hardware : gtkwave

MIPS FPGA Computer VHDL Simulator

H e l l o f r o m M I P S !

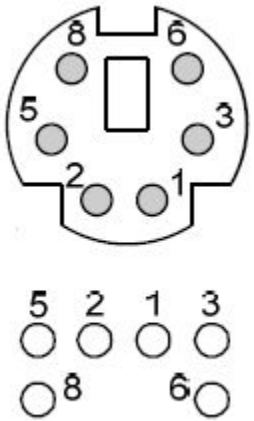
Next Document Save Save As Close Undo

```
ENTRY POINT FOR THE FIRMWARE

ection .entry, "ax"
lobal start
art:

 /*j .*/

lui   $v0, 0xBE00        # IF ID EX MM WB
ori   $v0, $v0, 0x0000   #    IF ID EX MM WB
ori   $v1, $0, 'H'       #       IF ID EX MM WB
sw    $v1, 0x00($v0)     #          IF ID EX MM WB
ori   $v1, $0, 'e'       #             IF ID EX MM WB
sw    $v1, 0x08($v0)
ori   $v1, $0, 'l'
sw    $v1, 0x10($v0)
ori   $v1, $0, 'l'
sw    $v1, 0x18($v0)
ori   $v1, $0, 'o'
sw    $v1, 0x20($v0)
```

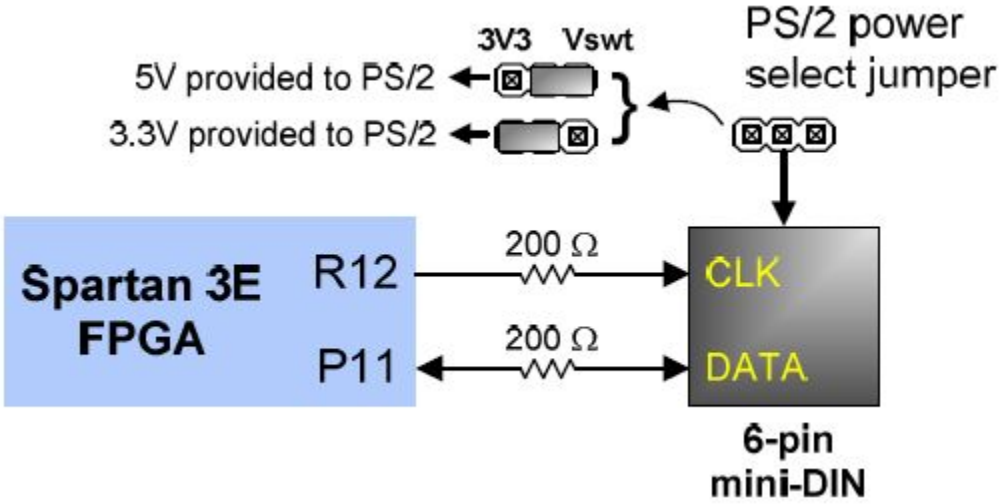
R3000.pdf (SE | hardware – Dr | GTKWave - wa | start.s – Kate | build : make – | MIPS FPGA Co | Pink Floyd - Ec | us | 03:40 PM Saturday 07 November 2015

PS/2 Interface (4/6)

- A PS/2 port is mounted on Nexys II board.

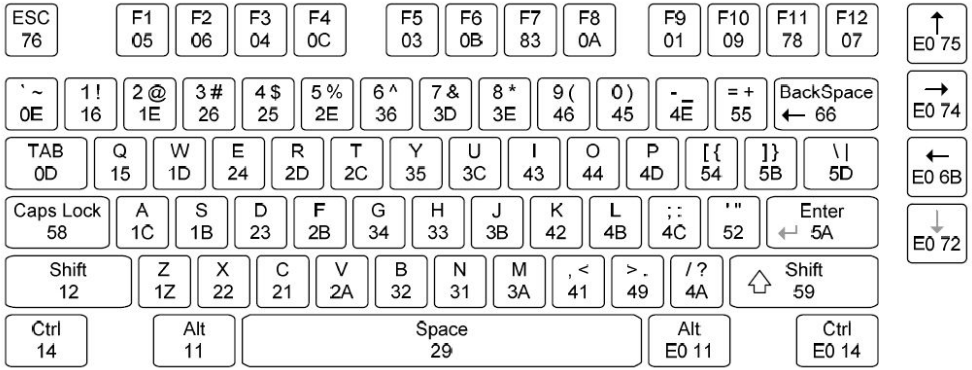


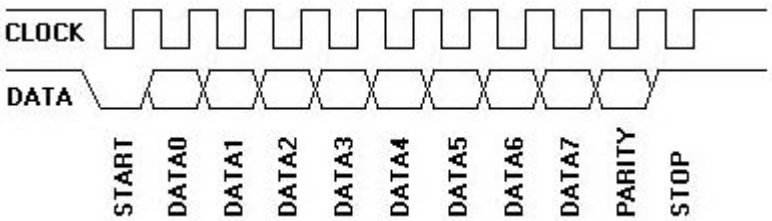
Pin1: Data
Pin2: Data
Pin3: GND
Pin5: Vdd
Pin6: Clock
Pin8: Clock



PS/2 Interface

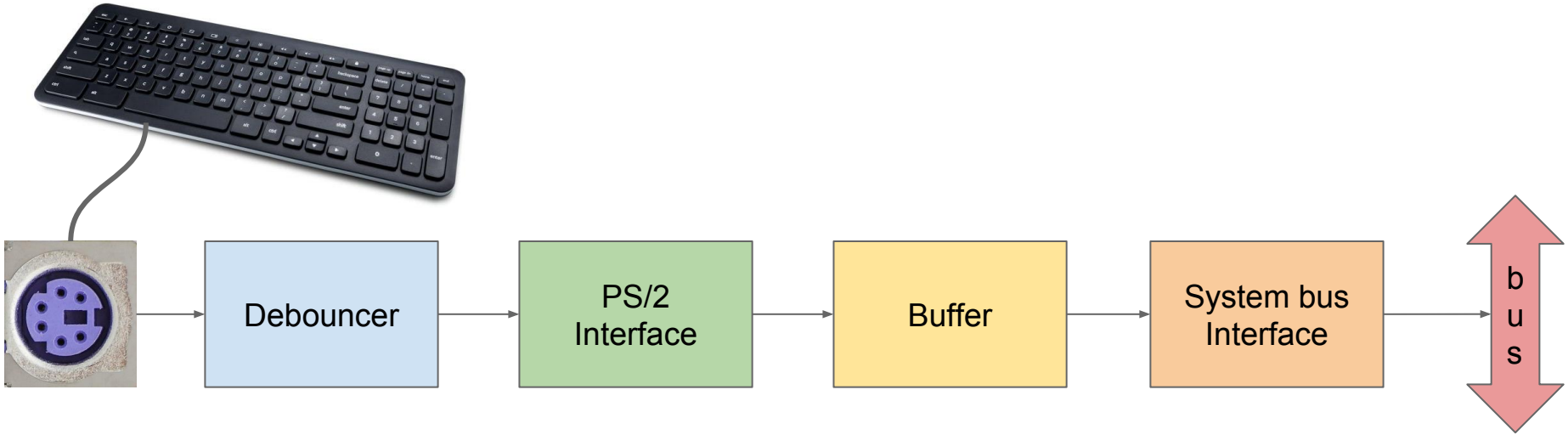
- When the user presses down (or up) one of the keys on the keyboard, a specific scan code is transmitted by the PS/2 keyboard (through CLK and DATA signals).





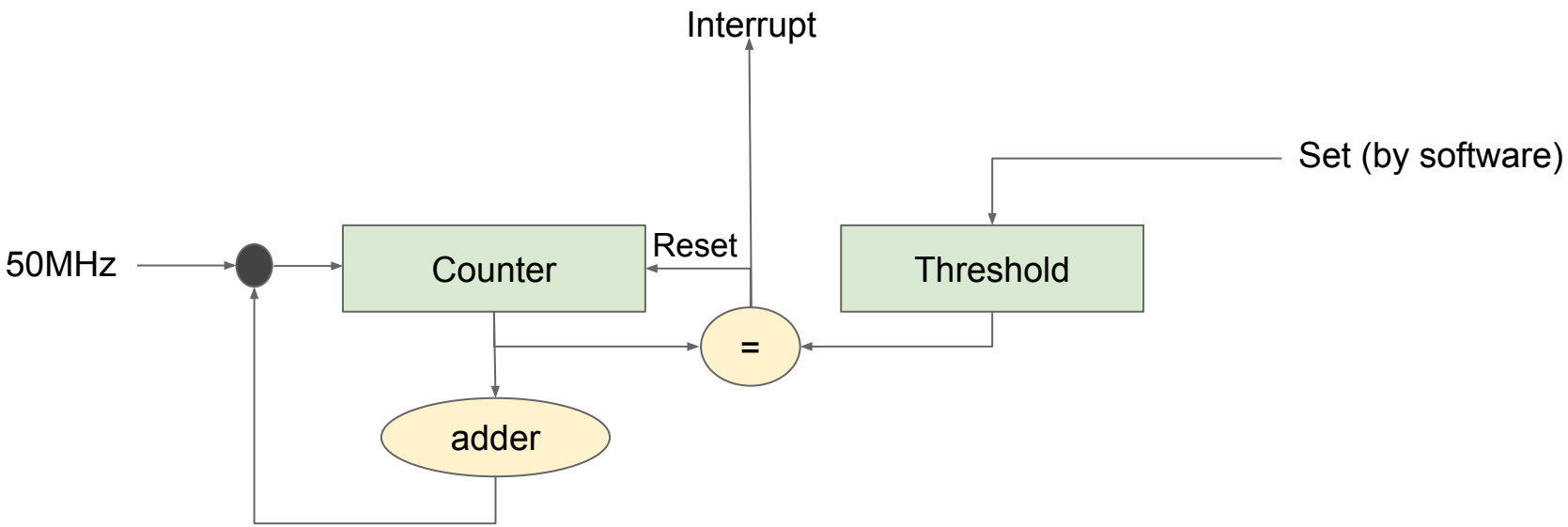
PS/2 Interface

- When a scan code is sent over PS/2 cable, our PS/2 interface stores it in a buffer and interrupts the CPU. System software accesses a special register to read the contents of the buffer.
- Translation from AT scan codes to ASCII is done by software.



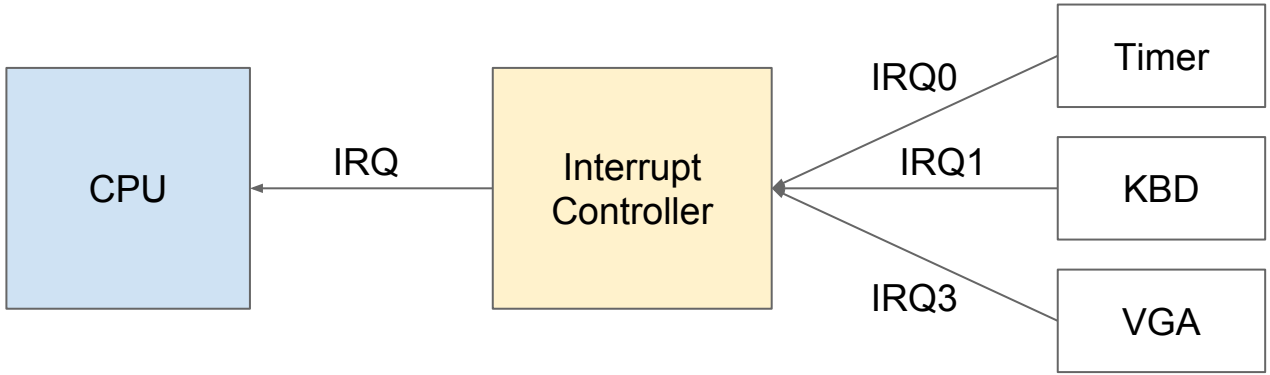
Programmable Timer (5/6)

- Similar to programmable interval timer (PIT) chip in the PC world.
- Works as a programmable frequency divider for the input 50MHz clock.
- Could be used by the OS to trigger scheduler when a process is timed out.



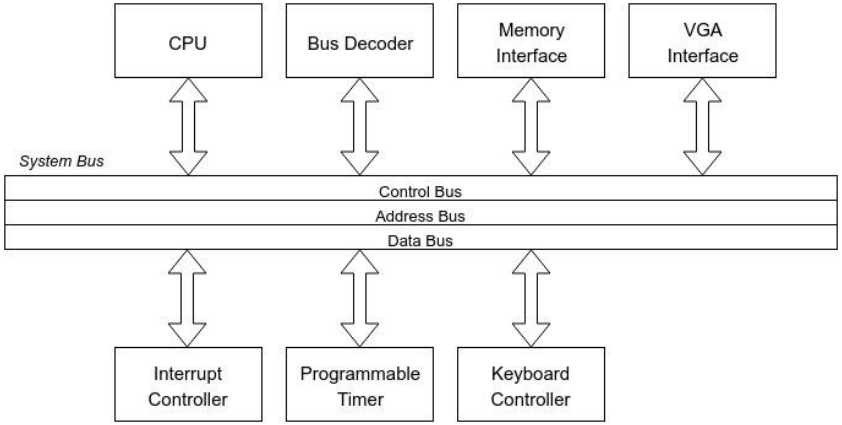
Interrupt Controller (6/6)

- Various interrupt sources: timer, keyboard, and VGA.
- Interrupt signals generated by these sources are fed into interrupt controller component, which manages priority and allows software to enable/disable the whole interrupt system.
- One interrupt signal between interrupt controller and CPU (removes the overhead of interrupt management from CPU).



Architecture Summary

- System components:
  - CPU
  - Memory (16MB RAM + 16MB ROM)
  - I/O devices (VGA, keyboard, timer, interrupt controller)
- All components are connected to a shared bus (system bus).
- Components are implemented in VHDL and uploaded to FPGA chip.

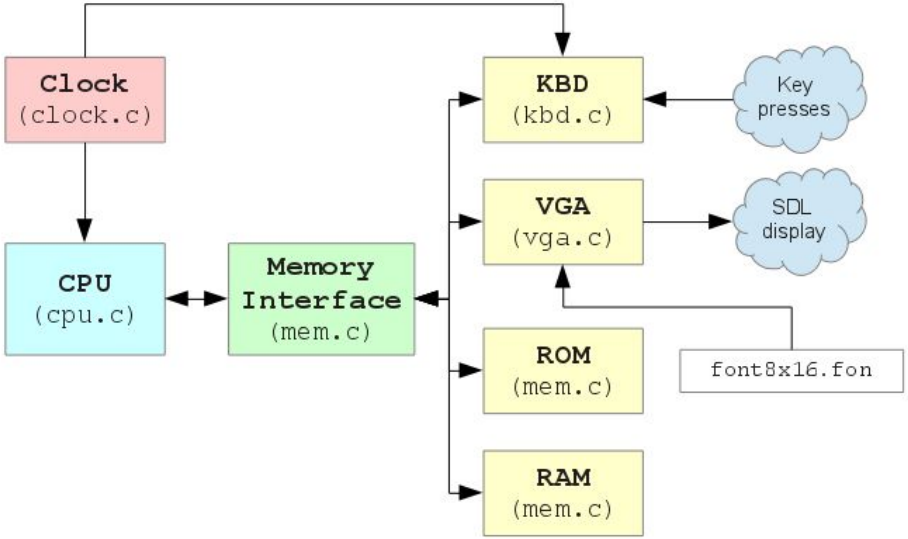


Outline

- High-Level Architecture
- System components:
  - Processor
  - Memory
  - VGA
  - PS/2 Keyboard
  - Timer
  - Interrupt Controller
- Emulation
- Software
  - Firmware
  - OS
- Conclusion

Emulation

- One architecture; several implementations:
  - FPGA implementation.
  - VHDL simulation (very slow).
  - Emulator in C over GNU/Linux.
- The emulator *replicates* the architecture we explained earlier.
- Work on the emulator started with the start of the project. We used it to validate the design before testing on FPGA.



Outline

- High-Level Architecture
- System components:
  - Processor
  - Memory
  - VGA
  - PS/2 Keyboard
  - Timer
  - Interrupt Controller
- Emulation
- Software
  - Firmware
  - OS
- Conclusion

Firmware

- Stored on the 16MB ROM chip.
- Aligned such that the physical address of the first instruction of the firmware program = MIPS Reset Vector.
- Tasks:
  - Boots up the system.
  - Initializes hardware.
  - Performs PoST.
  - Loads the OS.
  - Provides an interface between the OS and hardware (before the OS loads its own drivers).

```
Alexandria University,
Faculty of Engineering,
Computer and Systems Eng. Dept.

      ****************************************
      * MIPS Microcomputer System on FPGA *
      ****************************************

MIPS-I 32-Bit CPU at 50MHz
Memory Test: 0x00004000KB OK

No valid bootable medium found! drop to BIOS shell...

Welcome to MIPS computer shell. Type `help' for command listing.
> _
```

```
Alexandria University,
Faculty of Engineering,
Computer and Systems Eng. Dept.

    ***************************************
    * MIPS Microcomputer System on FPGA *
    ***************************************

MIPS-I 32-Bit CPU at 50MHz
Memory Test: 16384KB OK

Loading boot/loader.bin (inode 4) to 0x00000000... done
```

MIPS FPGA Computer Emulator

```
Alexandria University,
Faculty of Engineering,
Computer and Systems Eng. Dept.

     ************************************
     * MIPS Microcomputer System on FPGA *
     ************************************

MIPS-I 32-Bit CPU at 50MHz
Memory Test: 14464KB
```

Operating System

- We chose to port Quafios OS to our system.
- Quafios was written for IBM PC, but it was designed to be portable. So we only modified the architecture-dependent part of the kernel and added device drivers for our machine.
- Quafios kernel: monolithic
  - Architecture-dependant code.
  - Memory manager (pmem, kmem, umem).
  - Filesystem (VFS, diskfs, tmpfs, devfs, and sysfs).
  - Device drivers.
  - Process manager and scheduler.

Boot Quafios 2.0.1 (MIPS) in text mode

Boot from first storage medium
Reboot

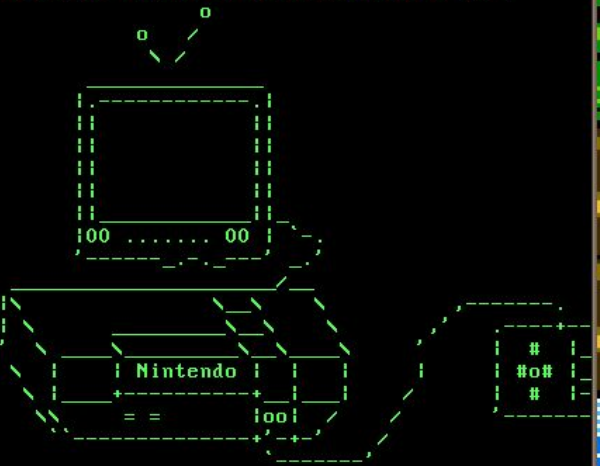
Thank you for using Quafios. Quafios is free software; any user has
the freedom to run, copy, distribute, study, change and improve Quafios.
For more information, please visit (http://www.quafios.com/).

```
# ---------------------------------------------#
# Inputs: N/A                                  #
# ---------------------------------------------#
# Outputs: N/A                                 #
################################################

main:
    # print splash
    ls      $a0, "%a"
    ori     $a1, $0 , 0x0A
    bios    printf
    lui     $a0, %hi(splash)
    addiu   $a0, $a0, %lo(splash)
    bios    printf
```

```
make[1]: Nothing to be done for
make[1]: Leaving directory '/hom
cd mipsemu && ./mipsemu ../firmw
```

MIPS

```
Loading /boot/nes.bin (inode 4) to 0x80010000... done
```




```
Booting from NES image...
```

Nintendo PPU

Outline

- High-Level Architecture
- System components:
  - Processor
  - Memory
  - VGA
  - PS/2 Keyboard
  - Timer
  - Interrupt Controller
- Emulation
- Software
  - Firmware
  - OS
- Conclusion

Conclusion

- We implemented a general-purpose computer using Nexys II FPGA board with Spartan-3E FPGA chip.
- Hardware:
  - CPU (implementation for MIPS-I instruction set with cache and TLB).
  - Memory interface: 16MB ROM chip + 16MB RAM chip.
  - I/O devices: VGA, keyboard, timer, and interrupt controller.
- Emulator.
- Software: Firmware + Quafios.
- Future Work:
  - Improvements for the pipeline.
  - Sound, USB, SDCard, Ethernet, and other interfaces.

Thank You