# To Do list:

## Prototype-board:

* Select support-components for all the main components (resistors, caps, et c)
* Place all components in the optimum manor
* Calculate impedance-matching of high-speed signals (like between GPU and DRAM)
* Trace & Route

## GPL-GPU core:

* Port VHDL and Verilog code to System Verilog
* Alter code to work for Lattice FPGA

# Manufacturers:

## JLCPCB:

For the moment, the minimum clearance for multi-layer PCB when using BGA is 0.127mm spacing pad to track and 0.127mm pad to pad.

3.5 mil for multiple layers

## TRING PCB:

https://pcb.ba/Home/PCB

Located in Bosnia / Herzegovina

# Specifications:

## Specification-target:

**Bus**: PCI 2.2

**Min Memory Size**: 8MB

**Memory Type**: DDR2

**Card Type**: S-VGA

**Ports**: VGA, DVI-I, USB-C (HDMI 2.1, DP 1.4b), JTAG (pcb edge con.)

**Video Acceleration**: MPEG-1 (VCD), MPEG-2 (DVD)

**Core**: 64bit

**Effective Memory Clock**: 65 (MHz)

**Memory Bus Width**: 64bit

**Memory Bandwidth**: 250 (MB/s)

**Max Memory Clock**: 72 (MHz)

**Min Memory Clock** : 50 (MHz):

**Ramdac**: 135 (MHz)

(1.88 times faster Ramdac than actual Ram.)

The best old CRT’s can achieve speeds of up to 177 Mhz pixel-rate – the DAC needs to be able to serve analogue signals that fast.

## Speed requirements calculations:

### Memory-bandwidth

You multiply the effective frequency by the bus width to get bandwidth throughput.

#### Example:

“gtx 970 has a 256-bit memory bus and GDDR5 memory running at 1753Mhz

The memory is quad pumped so its effective speed is 7012Mhz (1753\*4).   
7012 \* 256 = 1795072 Mb/S or 224GB/s”

#### My application:

Winbond: 32-bit, 1Gb, DDR2, @533 Mhz

ISSI-1: 16-bit, 2Gb, DDR3, @800 Mhz

Micron-1: 32-bit, 4Gb, DDR4, @1866 MHz

Alliance-1: 8-bit, 4Gb, DDR4, @1200 MHz

Micron-2: 16-bit, 1Gb, DDR3, @933 MHz

https://www.digikey.se/sv/datasheets/microntechnologyinc/micron-technology-inc-1gb\_1\_35v\_ddr3l

Assuming the frequency shown is the single rate then the Winbond effective rate is 1066Mhz and the ISSI is effectively 1600Mhz as DDR (double data rate) updates on both positive and negative edges of the cycle.

Winbond: 32 \* 1066 = 34112 Mb/s or 4.264 GB/s

MT53E128M32D2DS-053 WT:A

32 \* 3732 = 14,9 GB/s

**AS4C256M16D4-75BCN**

16 \* 2666 = 42656 Mb/s or 5,33 GB/s

### PCI-bus bandwidth:

PCI bus bandwidths can be calculated with the following formula:

frequency \* bit width = bandwidth

#### Example:

33.33 MHz \* 32 bits => (1067 Mbit/s)/8 = 133.32 MB/s

*(divide by 8 for final MB/s)*

#### My application:

66 mhz \* 64 bits = (4224 Mbit/s)/8

PCI 64-bit, 66 MHz: **533 MB/s (**4.266 Gbit/s**)**

66 Mhz \* 32 bits = (2112 Mbit/s)/8

PCI 32-bit, 66 Mhz: **264 MB/s**

#### Signal-translation:

Bandwidth target for a 32-bit bus: 8.25 MB/s (per data-line)

Bandwidth formula:

Frequency \* bit width

My translation-chip: 8.8 ns / 113 Mhz

100 \* 4 = 400 Mbit/s = 50 MB/s

113 \* 16 = 1808 Mbit/s = **226 MB/s**

(6\*5 = 30 MB/s)

### DAC-speed:

The DAC needs to be running at the per-pixel frequency, which at 177Hz VGA resolution would be in the vicinity of 100MHz.

There are "video DACs" designed for this kind of use case, though other general high-speed ones would also work.

**Formula:**

*Output Frequency \* Horizontal lines \* vertical lines*

(plus margins for refresh intervals.)

177 \* 640 \* 480 = 54 374 400 (54 Mhz)

177 \* 1920 \* 1080 = 367 Mhz

(probably around 400 Mhz for refresh intervals – the maximum speed of the FPGA)

# Components:

## Active:

CPLD

FPGA

VBIOS-chip

VRAM

## Passive:

Resistors

Capacitors

Inductors

# Features:

## Resolutions:

320x200 (16 colors)  
  
640x350 (16 colors)  
  
640x480 (2 colors)  
  
640x480 (16 colors)  
  
320x200 (256 colors)

320 x 200

320 x 240

320 x 400

320 x 480

360 x 200

360 x 240

360 x 400

360 x 480

640 x 400

640 x 480

648 X 480

800 x 600

960×720

1024 x 768

1280 x 720

1280 x 1024

1920 x 1080

## Colour modes:

2 colours (b/w)

16 colours

32 colours

256 colour-mode

16-bit colour mode

24-bit colour mode

32-bit colour mode

# Voltage/Power requirements:

**FPGA1**: 0.8V, 1.1V, 2.5V, (1.8, 3.3 V tolerant I/O) / 212 mA, 26 mA, 0.5 mA, 11 mA (250 mA)

(vref= 0.8V)

**DRAM**: 1.8V, 1.2V (I/O) / 30 mA

**VBIOS**: 3.3V / 100 mA

**Sig.Tran**: / 2 uA, 100 uA, 100 uA, 900 uA, 900 uA, 2uA, 20 uA, 20 uA, 50 uA (2094 uA = 2.1 mA)

(x4 = 8.4 mA)

PCI\_a: 3.3V

PCI\_b: 5V

3.3V -> 1.1, 1.8, 2.5 V

5V -> 1.1, 1.8, 2.5 V

Total Amp:

250 + 30 + 100 + 8,4 mA = **388,4 mA**

Total Watt: 1,942 W

## VRM-needs:

Shielded Inductors needed (make sure they don’t have a loud hum)

----------

For 0.8V:

R1-1= 5 Ohm

R2-1 = 10K Ohm

(special case)

--------------

For 1.1V:

R1-2 =

R2-2=

For 1.8V:

R1-3 =

R2-3 =

For 2.5V:

R1-4 =

R2-4 =

### Voltages:

0.8V, 1.1V, 1.8V, 2.5V,

### Amperage:

The PCI-bus can supply the following:

+3.3V/7.6A,

+5V/5A,

+12V/500mA,

-12V/100mA.

# Analogue signals

Separate routing as much as possible, carefully connect the ground-lines, to the same ground-plane, utilizing both coils and capacitors.

May be necessary to use a **Coupled Inductor** for the analogue-digital ground meeting.

Utilize VIA-fencing.

https://www.latticesemi.com/-/media/LatticeSemi/Documents/DataSheets/ECP5/FPGA-DS-02012-1-9-ECP5-ECP5G-Family-Data-Sheet.ashx?document\_id=50461

https://electronics.stackexchange.com/questions/128637/how-should-i-connect-agnd-and-dgnd

https://www.analog.com/en/analog-dialogue/articles/staying-well-grounded.html

# Potential Issues

Clock domain crossing

Synchronization of the PCI interface

Latency

Delays between main memory and the GPU – BIG bottleneck!

*(Developers need to work around this big limitation, by making sure to overlap transfers with computation, doing preemptive transfers and other techniques to mask the transfer overhead.)*

# References:

**Project VGA**

<http://wacco.mveas.com/index.php?entry=22>

<https://wiki.osdev.org/VGA_Hardware#Overview>

<https://www.youtube.com/watch?time_continue=102&v=EHePto95qoE&feature=emb_logo>

**GPL-GPU**

<https://github.com/asicguy/gplgpu>

**VIA Fencing**

https://en.wikipedia.org/wiki/Via\_fence

*“Via fences too close to the line being guarded can degrade the isolation otherwise achievable. In stripline, a* ***rule*** *of thumb is to* ***place the fences at least four times*** *the trace to ground-plane distance* ***away*** *from* ***the line*** *being guarded.”*

**PCI-bus interface card**

<http://markmuzzin.blogspot.com/2011/10/pci-bus-interface-card.html>

<https://pinouts.ru/Slots/PCI_pinout.shtml>

http://www.interfacebus.com/Design\_PCI\_Pinout.html

https://allpinouts.org/pinouts/connectors/buses/pci/

**Video Timings Calculator**

<https://tomverbeure.github.io/video_timings_calculator>

<https://en.wikipedia.org/wiki/Coordinated_Video_Timings>

**Frequency-choice for Voltage-switch**

<https://www.allaboutcircuits.com/technical-articles/how-to-choose-the-frequency-of-your-switching-regulator/>

**Card Edge Connectors**

<https://github.com/turingbirds/con-pcb-slot>

**JLCPCB fpga board**

https://www.eevblog.com/forum/fpga/custom-spartan-7-board-for-beginners/

**Layout-instructions**

https://www.youtube.com/watch?v=R\_Ud-FxUw0g

Ground always down

Never have lines crossing your symbols

All signals should be drawn going from left -> right

Don’t show footprint-names on IC-package symbols

## Symbiflow – Open Source FPGA – Trellis (ECP5)

https://symbiflow.readthedocs.io/en/latest/prjtrellis/docs/

https://github.com/SymbiFlow

<https://symbiflow.readthedocs.io/en/latest/symbiflow-arch-defs/docs/source/getting-started.html>

### IRC

<https://webchat.freenode.net/#symbiflow>

### Suggested contribution:

[SV-Tests](https://github.com/SymbiFlow/sv-tests)

YOSYS

Syntax 8.3

Jump statements 12.8

Array locator methods 7.12.1

FX68K m68k core fx68k

<https://antmicro.com/blog/2019/11/systemverilog-test-suite/>

**JLCPCB design rules**

<https://jlcpcb.com/capabilities/Capabilities>

**GitHub**

https://github.com/Doomn00b/Open-SVGA/blob/master/README.md

**Lattice FPGA’s:**

https://www.digikey.se/en/articles/fundamentals-of-fpgas-part-2-getting-started-with-lattice-semiconductor-fpgas