Graphics Hardware

Geert Uytterhoeven*

Abstract

This paper describes the structure of modern graphics hardware.

It is mainly intended for people who want to write a Linux frame buffer device driver, but we hope it will be useful for other purposes too.

*geert@linux-m68k.org, http://www.cs.kuleuven.ac.be/~geert/

Contents

1 Introduction

...

2 A Comparison of Graphics Memory Technologies

Memory performance is characterized by two parameters: access time (t_a) and cycle time (t_c). The first memory access in a sequence takes t_a to complete, which each consecutive access to a consecutive memory location re uires t_c. For memory that supports burst accesses, $t_c < t_a$, so one can optimize performance by using long bursts as much as possible. Hence t_a specifies the latency, while t_c specifies the peak memory bandwidth.

If we measure all time relative to the memory clock cycle period, i.e. $t_a = n_a \cdot t_{\text{mclk}}$ and $t_c = n_c \cdot t_{\text{mclk}}$, synchronous memory has $n_a = 1$, and supports higher memory clocks than asynchronous memory tehcnologies like FPM and EDO. Typical values for n_a and t_a are (shown as t_a-n_a-n_a-n_a):

SDRAM:	8-1-1-1
FPM:	5-3-3-3
EDO:	5-2-2-2

SGRAM and WRAM differ from SDRAM and FPM/EDO DRAM in that they support special block operations, yielding higher performance for many graphics operations.

only one of graphics engine, CPU and acceleration engine can access memory at the same time

The memory has one read-write port shared by the CPU and the acceleration engine, and one sequential read port for the graphics engine. So both CPU/acceleration engine and the graphics engine can access the memory at the same time, increasing performance.

low acces time high cycle time ($t_c > t_{\text{mclk}}$)
low cycle time ($t_c = t_{\text{mclk}}$) high acces time

	single ported	**dual ported**
asynchronous	FPM RAM	VRAM
	EDO RAM	WRAM
synchronous	SDRAM	
	SGRAM	

Advantages and disadvantages

- Single ported memory

- ⊕ Cheap, commodity memory
- ⊖ Low performance for advanced graphics modes (high resolutions, many colors)

- Dual ported memory
  - ⊕ High performance
  - ⊖ Expensive packaging (high pin count)
- Asynchronous memory
  - ⊕ Low latency
  - ⊖ Low bandwidth
- Synchronous memory
  - ⊕ High bandwidth
  - ⊖ High latency

SRAM: e.g. Sun bwtwo, built of discrete TTL-logic

Most modern graphics cards use commodity SDRAM or SGRAM for cost-effectiveness. Hence the size and type of the memory on the graphics card are driven by advances in system memory technology as well. Since the smallest available SDRAM chips these days are 16 Mbit (2 M $\times$ 8 bit) and since many new graphics chipsets have a 128-bit bus, requiring a multiple of 16 memory chips, graphics cards with 32 MB or more of video memory are showing up.

mclk: memory clock

3 The Graphics Engine

The graphics engine is responsible for fetching pixel data from the graphics memory and generating the synchronization signals according to the wanted screen geometry.

3.1 Display Geometry

3.2 The Display FIFO

To compensate for the rate difference between the fetching of the pixel values and the supply of words from graphics memory, a FIFO buffer is used.

[cfr DSP GT mail]

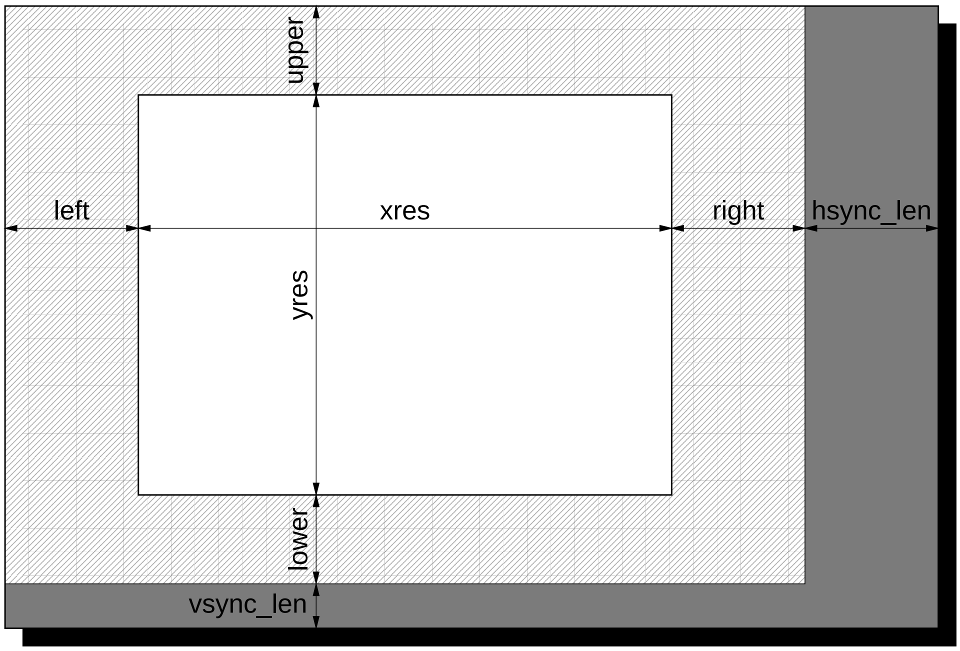

Figure 1: Display geometry.

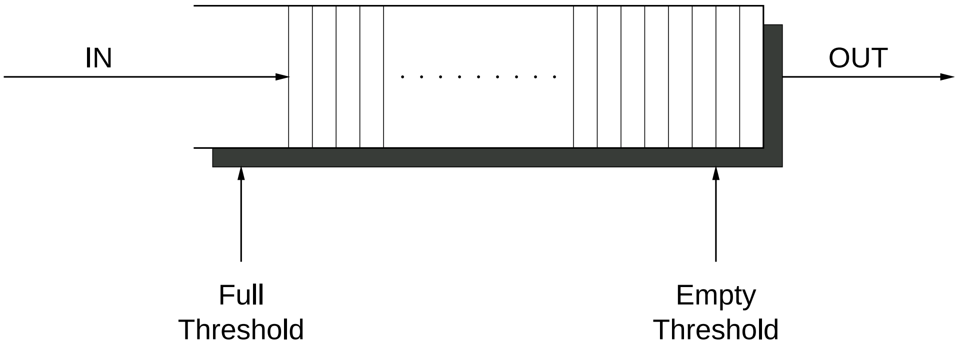

Figure 2: FIFO.

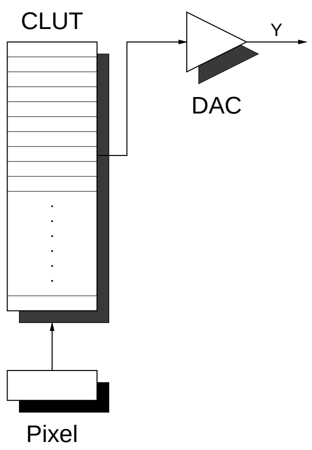

Figure 3: (a) A *grayscale* visual: each pixel value is looked up in a look-up table before being passed to the DAC.

4 The RAMDAC

The RAMDAC (RAM-based Digital-to-Analog Converter) takes care of converting the raw pixel data to analag RGB signals, suitable for sending to the monitor.

As its name already indicates, a RAMDAC consists of two parts:

RAM Random Access Memory, for storing the CLUT (Color Look-Up Table).

DAC Digital-to-Analog Convertors, for converting the digital color components to analog signals that can be understood by the monitor.

4.1 Visuals

The visual specifies the type of conversion that is used to convert raw pixel values to RGB triplets.

4.1.1 Monochrome

4.1.2 Grayscale

Grayscale Each pixel value is looked up in a look-up table before being passed to the DAC. See Figure 3.

Staticgray Each pixel value is directly passed to the DAC. This is similar to *grayscale*, but the look-up table is fixed.

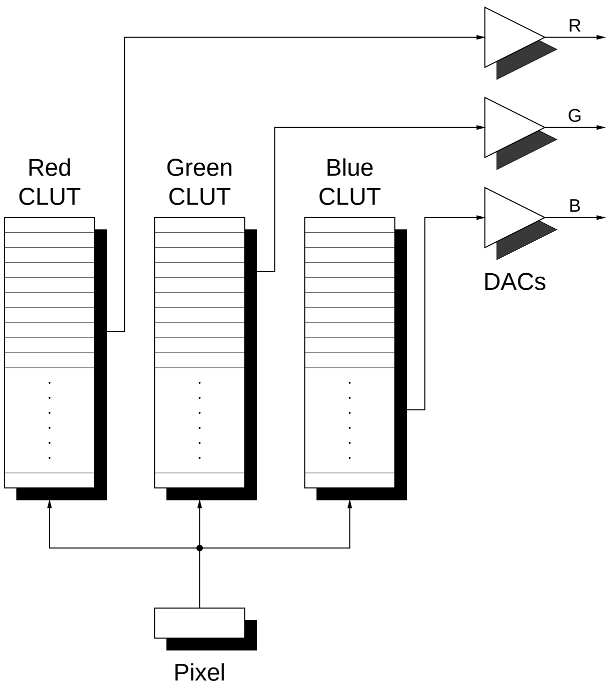

Figure 4: A *pseudocolor* visual: each pixel value is looked up in three look-up tables, one for each color component, before being passed to the DACs.

4.1.3 Color

Pseudocolor Each pixel value is looked up in three look-up tables, one for each color component, before being passed to the DACs. See Figure 4.

Static Pseudocolor This is similar to *pseudocolor*, but the look-up tables are fixed.

Directcolor Each pixel value is split in three components, which are individually looked up in three look-up tables, one for each color component, before being passed to the DACs. See Figure 5.

Truecolor Each pixel value is split in three components, which are directly passed to the DACs. This is similar to *directcolor*, but the look-up tables are fixed to the identity mapping. See Figure 6.

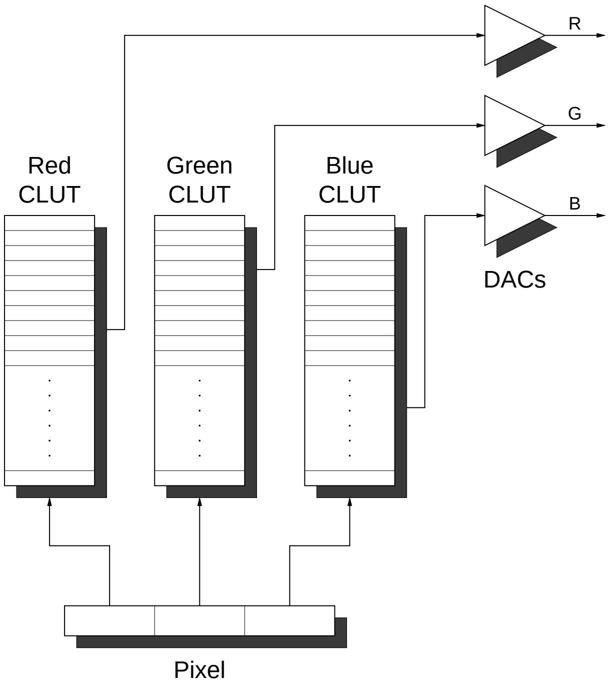

Figure 5: A *directcolor* visual: each pixel value is split in three components, which are individually looked up in three look-up tables, one for each color component, before being passed to the DACs.

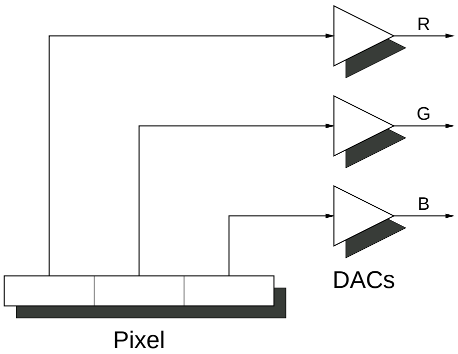

Figure 6: A *truecolor* visual: each pixel value is split in three components, which are directly passed to the DACs.

4.2 Overlays

Some RAMDACs support multiple visuals using overlays, both pseudocolor and truecolor. This is useful for showing photorealistic images (truecolor) and generated graphics (e.g. window borders and menus; pseudocolor).

4.2.1 8+24

Each pixel value is 32 bits wide, with a layout

R8	G8	B8	I8 = K
—	—	—	I8 ≠ K

Normally a pseudocolor visual is used, using field *I* as the CLUT index. However, if *I* is set to a special value (the colorkey), then the *RGB* triplets are immediately passed to the DACs, providing a truecolor visual.

4.2.2 8+15

K	R5	G5	B5
-K	I8		

The key bit *K* specifies whether the rest of the pixel data is interpreted in a truecolor (*RGB*) or a pseudcolor (*I*) visual.

4.3 Transparency

Usually RGBA: A = α-channel, for overlaying color videa data (see Section sec:video). Pseudocolor overlays (Section 4.2) can be considered a special case of this.

4.4 Color Depth versus Bits Per Pixel

Bits per pixel (bpp) is the actual number of bits each pixel occupies in memory, while color depth is the number of bits of each pixel value that is actually used for color information.

Examples:

- bpp 32, depth 24:

U8	R8	G8	B8

- bpp 16, depth 15:

U1	R5	G5	B5

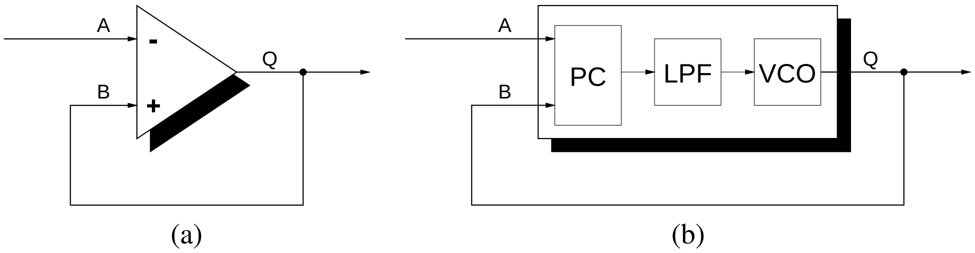

Figure 7: (a) OpAmp and (b) PLL with feedback. A PLL consists of a phase comparator (PC), a low-pass filter (LPF) and a voltage controlled oscillator (VCO).

5 Clock Signal Generation

5.1 Fixed Clock Crystals

One or more quartz crystals, and optional clock dividers (/2, /3, /4, /8, …).

Commonly used crystals: 14.31818 MHz, 28.36363 MHz, 29.498928 MHz, 44.9 MHz, 50 MHz

Not flexible.

5.2 Programmable Clock Synthesizers

Clock signal generators based on fixed clock crystals and simple frequency dividers are not very flexible:

- Only a limited number of different frequencies can be generated.
- Frequencies higher than the crystal frequency cannot be generated.

Therefore modern graphics chipsets use programmable clock synthesizers based on a PLL (Phase Locked Loop) and only one reference crystal.

A PLL consists of three parts, as is shown in Figure 7b:

PC Phase comparator

LPF Low-pass filter

VCO Voltage controlled oscillator

A PLL can be abstracted in a similar way as an OpAmp (Operational Amplifier) with a feedback loop. While an OpAmp drives its output Q such that its inputs A and B have the same *voltage*, a PLL generates an output frequency Q such that its inputs A and B have the same *frequency*.

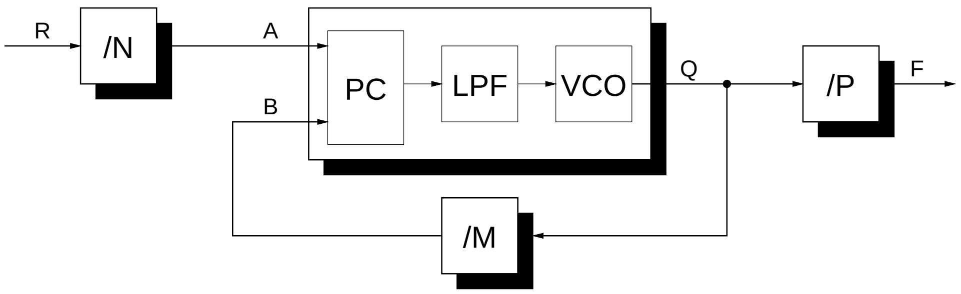

Figure 8: A PLL is fed by a reference clock R divided by reference divider N. The loop contains a feedback divider M. The output of the PLL is divided by the post divider P.

In Figure 7 we show an OpAmp and a PLL with a feedback loop: in these circuits the voltage (Figure 7a) or the frequency (Figure 7b) of the output Q will be equal to the voltage or frequency of the input A.

An OpAmp can serve as a voltage amplifier. Its amplification factor is controlled by adding a voltage divider to the feedback loop. Similarly, a PLL can be used to multiply the frequency of a signal. The multiplication factor is controlled by adding a frequency divider to the feedback loop.

In most PLL implementations for clock synthesizers, two more frequency dividers are added to the system, to obtain a scheme like in Figure 8:

M Feedback divider

N Reference divider to divide the frequency of the reference crystal

P Post divider to divide the PLL output, to obtain frequencies much lower than the operational range of the PLL.

The frequency dividers dictate the following relations:

$$
\begin{aligned}
A &= \frac{R}{N}, \\
B &= \frac{Q}{M}, \\
F &= \frac{Q}{P}.
\end{aligned}
$$

The PLL generates a frequency Q such that the frequencies of its inputs are equal, hence we have

$$\begin{aligned}
\frac{R}{N} &= \frac{Q}{M}, \\
Q &= \frac{M}{N} \cdot R, \\
F &= \frac{M \cdot R}{N \cdot P}.
\end{aligned}$$

A(n approximation to a) specified frequency can be generated by finding suitable parameters M, N and P.

PLLs are complex analog circuits and are rated to reliably generate frequencies within a limited frequency range only:

$$Q_{\mathrm{min}} \leq Q \leq Q_{\mathrm{max}}.$$

Typical values for Q_{max} are in the range of 135 MHz to 300 MHz, and increase with the advance in chipset technology. Q_{min} is usually about half Q_{max}. A programmable clock synthesizer cannot generate a frequency higher than the upper limit Q_{max}. To generate frequencies lower than the lower limit Q_{min}, the post divider must be used.

A graphics chipset usually needs two clock signals: the pixel clock (vlck) and the memory clock (mclk). This can be accomplished by using two independent clock generators. However, sometimes they may share the reference divider. In the latter case, it's recommended to chose N such that the largest range of frequencies can be generated:

$$\begin{aligned}
\frac{M_{\mathrm{max}}}{N} \cdot R &= Q_{\mathrm{max}}, \\
N &= \frac{M_{\mathrm{max}} \cdot R}{Q_{\mathrm{max}}}.
\end{aligned}$$

6 The 2D Acceleration Engine

7 The 3D Acceleration Engine

8 The Video Engine

9 The Bus Interface

10 Linux Frame Buffer Device Programming

10.1 CRT Programming

10.1.1 Visual Selection

If your graphics hardware is capable of supporting multiple visuals, you best chose the most powerful visual. Since a *truecolor* visual is a *directcolor* visual with fixed look-up tables, a *directcolor* visual is more powerful than a *truecolor* visual. Similarly, a *directcolor* visual can be emulated on a *pseudocolor* visual, thus a *pseudocolor* visual is the most powerful.

Due to the large amount of entries in the look-up table for a *pseudocolor* visual, most hardware does not allow *pseudocolor* visuals for color depths larger than 8 or 10 bit (256 or 1024 colors).

A List of Abbreviations

bpp	:	Bits Per Pixel
CLUT	:	Color Look Up Table
CRT	:	Cathode Ray Tube
DAC	:	Digital-to-Analog Convertor
DDC	:	Display Data Channel
DRAM	:	Dynamic Random Access Memory
DSP	:	Digital Signal Processor
EDO RAM	:	Enhanced Data Out Random Access Memory
FIFO	:	First In First Out
FPM RAM	:	Fast Page Mode Random Access Memory
i2c	:	Inter IC Connect
IC	:	Integrated Circuit
LCD	:	Liquid Crystal Display
MHz	:	MegaHertz
OpAmp	:	Operational Amplifier
Pixel	:	Picture Element
PLL	:	Phase Locked Loop
RAM	:	Random Access Memory
RAMDAC	:	Random Access Memory Digital-to-Analog Convertor
SDRAM	:	Synchronous Dynamic Random Access Memory
SGRAM	:	Synchronous Graphics Random Access Memory
SRAM	:	Static Random Access Memory
WRAM	:	Windowed Random Access Memory

B License