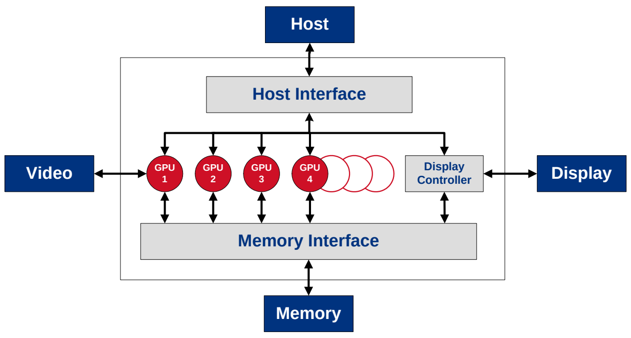
BADGE
BitSim Accelerated Display Graphics Engine

Product Brief



BADGE Architecture

BADGER - BADGE Reference design

Functional overview

BitSim Accelerated Display Graphics Engine, BADGE, is tailored to drive displays in Embedded Systems. BADGE off-loads the host CPU, accelerates text and 2D graphics, overlays graphics on video and controls the display.

BADGE provides hardware acceleration for drawing common graphical shapes such as lines, rectangles and text, and for copying rectangles. BADGE is an Intellectual Property (IP) core used in both FPGAs and ASICs.

The modular design of BADGE allows it to be customized for specific needs. It can be optimized for cost or performance. The architecture makes it easy to add new functions to a product platform.

Tools

BADGE is provided together with a user friendly API, BADGElib, which makes the IP core easy to use from software. BADGElib is supplied in source C code. It allows the programmer to to use all hardware acceleration from any C-based development and debug environment.

Design and development of GUIs and content can be done with regular desktop tools. BitSim provides SW utilities that translates fonts and pictures from the desktop to the format used in the target system, or in the BADGER Reference Design.

Drivers for Windows CE and Linux can be supplied. High End Graphics Libraries are supported through Qt/Qtopia.

Reference design

The BADGER Reference design makes it easy to evaluate and demonstrate the intended use early in the development cycle. BADGER can serve as a starting point for the design of the target system.

BADGE
BitSim Accelerated Display Graphics Engine

Benefits

Performance

- High resolution – up to 4096 x 4096 pixels
- High color depth – up to 24-bit color, plus alpha
- Quality graphics acceleration
- Cost optimized versions

Easy to use

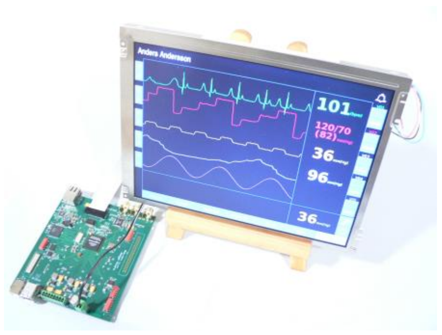
- Clean register structure
- BADGElib – API provided in C source
- Graphics design and development with standard desktop tools. SW utilities translates the data for the target system

Future proof

- Product life cycle ownership
- Migrate when you choose
- Avoid EOL and obsolescence
- Wide selection of available FPGA components and ASIC libraries
- A multitude of FPGAs, packages and temperature ranges

Flexibility

- Easy to adapt and modify
- Any display format, memory type and processor interface
- Any FPGA family or ASIC library



Demo running on BADGER

Features

- Text & 2D graphics acceleration (incl. BitBLT and Raster Operations)
- Text/Graphics overlay on video
- Up to 4096 x 4096 pixels display resolution
- Color depth up to 24 bits per pixel
- Anti-aliasing
- Alpha blending (2 types)
- Fully synchronous, synthesizable and technology independent RTL code
- Scalable for cost or performance

- Support for TFT displays
- Support for STN displays * Option *
- Support for multiple video sources
- Rotation and scaling of video
- Deflate option for data compression/ decompression
- CPU data buses of 32, 16 and 8 bits supported, or serial buses

- Reference design available for evaluation and development
- C-based API, BADGElib
- SW Utilities – Easy development of graphics in desktop environment
- Windows drivers
- Linux drivers

- Flicker free – Supports multi-buffered frame memory
- Sprites – Hardware cursor support
- Parallel LVTTL, serial LVDS interfaces
- Display power sequencing
- Portrait mode

General Description

BitSim Accelerated Display Graphics Engine, BADGE, accelerates graphics and controls displays. BADGE provides text and 2D acceleration, and manages overlay of text and graphics on video. BADGE can be scaled and adapted to meet high performance requirements or fit into cost optimized applications. BADGE enables advanced graphics without the cost or power increase of a high performance processor.

BADGE is a Graphic Accelerator between the Host Processor and the display. A number of Graphics Processing Units, GPUs, execute in parallel to generate the accelerated graphics. Each GPU performs a dedicated function. Example of GPU tasks:

- Draw pixels, lines and rectangles
- Write text of various fonts, sizes and colors
- Copy, resize and recolor objects of any shape
- Do Bit Block Transfers “BitBLT” and Raster Operations, ROP
- Draw/Move graphical objects – e.g. ”sprites”
- Handles analog and digital video

BADGE can perform Bit Block Transfers, BitBLT. This is a combined copy and raster operation. It is a highly effective method to put an object of any shape onto any type of background. It is also an easy way to create basic animations.

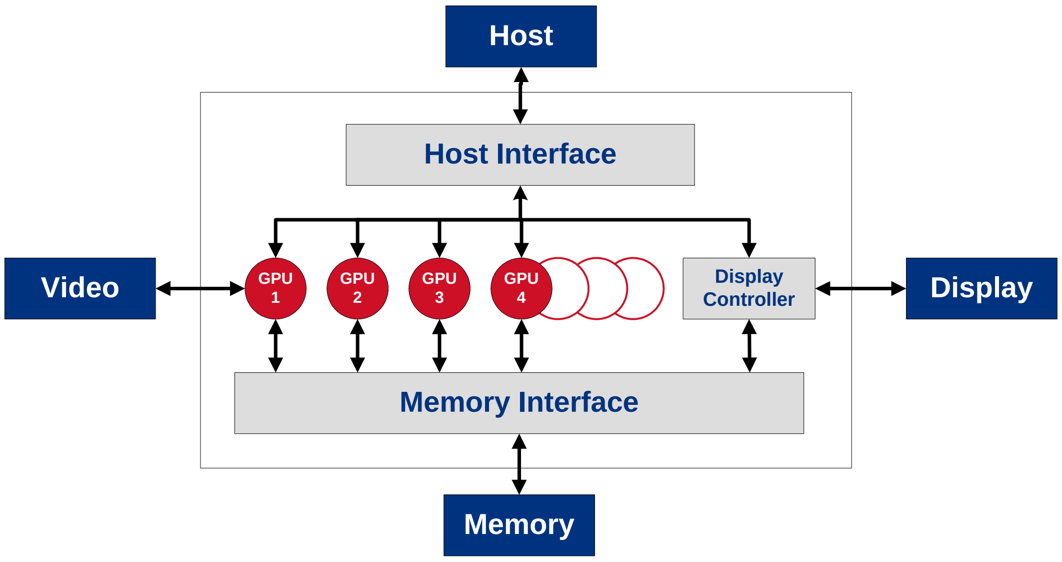
BADGE provides hardware acceleration for drawing common shapes, such as lines, rectangles and text.

BADGE is a modular design where only the required GPUs will be included in the design allowing to save cost, power and size. It is possible to tailor BADGE for specific applications and make a trade-off between hardware accelerated features and software generated graphics.

A dedicated Graphics memory is used for storing the displayed image and graphical objects, such as symbols, picture elements and text fonts. The Graphic memory performance will be scaled to match the requirements of the application. A set of tightly coupled memory controllers, for different memory types and sizes, make it possible to provide a wide range of predicable graphics performance. Implementations are also possible for shared memory systems, where the basic memory arbiter is connected to a shared memory system controller.

A variety of Host interfaces aimed at different CPU buses, and a variety of display controllers aimed at different display types are supported.

The user friendly C-based API, BADGElib, makes it possible to fully utilize the acceleration provided by the hardware. Extensions of BADGElib is a straight forward way to implement any fully accelerated, partially accelerated or non-accelerated operation.



Functional Description

BADGE Graphic Command Processing

In a system, the Host Processor issues BADGE commands to the Host Interface block. This routes the command to the associated Graphics Processing Unit, GPU. They transform the command into memory operations. The GPUs can execute in parallel. Command queues are available for the Host communication and for each GPU.

Host Interface

Practically any host CPU bus can be supported. Proven interfaces exists for a range of host controllers such as: General Programmable, Xilinx – MicroBlaze and PowerPC, Altera – NIOS, Intel - XScale and 386, Freescale PowerPC and Coldfire, Renesas M16C, NEC, etc. Adaptation to new host processors, or buses, are frequently done. Where the application allows, a serial interfaces like SPI, I2C, CAN, etc. can be used.

Memory Controller

A tightly coupled memory is essential in order to reach good system performance. The Memory Controller handles the interface to external graphics memory. It includes an arbiter, which allows data transfers between the GPUs and the graphics memory.

Memory controllers have been implemented for different memory types and sizes. They allow a wide range of graphics performance. Memory Controllers are available for SDRAM, DDR, SRAM and ZBT-SRAM with different bus widths. Alternatively the basic arbiter can be connected to the system memory controller for the ASIC or FPGA.

Display Controller

The display controller handles the interface to the LCD/TFT or STN displays. It reads image data from the graphics memory and outputs it to the display together with display clock, sync and enable signals. The display controller provides a hardware cursor in the form of a sprite. A Color Look Up Table, CLUT, can be implemented to save memory and memory bandwidth. Optionally, a selectable number of hardware layers can be implemented.

The interface to the display can either be a parallel LVTTL, a serial LVDS, a DVI compatible interface, or a combination of these. Special needs like multiple displays, or odd formats, can be supported.

BADGE Graphics Processing Units

Hardware Acceleration of BADGE commands – graphics operations – are performed by Graphics Processing Units, GPUs. Each GPU handles a specific task. The specific task is translated into a sequence of memory operations by the GPU.

CHRGPU

The Character GPU is used to accelerate text drawing with various fonts, sizes and colors. Simple commands relieve the host processor from complex text rendering operations. Support for Unicode fonts allows for a huge selection of languages and font types. Anti-aliased fonts provide a high quality image.

RCCGPU

The Rectangle Copy GPU performs rectangle copying including Raster OPerations (ROP), so called BitBLTs (Bit Block Transfers). ROP is normally used in Graphical User Interfaces (GUI), for example inverting and shadowing of icons. Animations and other 2D effects are easy to create with this GPU. Use of BitBLT creates dramatic acceleration of the system’s graphics performance.

SPDGPU

The Simple Pixel Drawing GPU is used for drawing points, lines and rectangles, with a specified color.

MDAGPU

The Memory Direct Access GPU is used for non-accelerated transfers between the Host processor and the graphics memory.

BADGE Processing Units

VPU

The Video Processing Unit converts the received video signal into RGB format. The VPU performs deinterlacing and color conversion before storing video frames into the graphics memory.

With the video frames in Graphics Memory, any graphics or text can be multiplexed over the video. A multiplexer switches between the BADGE graphics and the video from the VPU on a pixel-by-pixel basis.

NTSC and PAL detection is automatic. The video frame can be positioned to any part of the screen.

Options for video alpha-blending, scaling and rotation are available. The alpha blender makes it possible to gradually mix different percentages of video respective graphics within the same pixel.

The VPU accepts standard video, ITU-R BT.601/656, as input. The input data is selectable between 8-bit width at 27 MHz, or 16-bit width at 13.5 MHz.

Digital uncompressed video signals, such as SDI, are supported. For SDI, an external deserializer is required. For analog video, a composite video decoder or ADC is required external to the FPGA, such as composite video (CVBS), S-video or RGB as input.

HWE

The HardWave Engine, HWE, generates smooth, anti-aliased and gamma-enhanced waveforms. The waveforms support flicker free updating and scrolling. HWE manages this with a minimum effort by the host CPU and does not add to BADGE's external memory bandwidth or memory size requirements.

Multiple HWE units can be instansiated in BADGE, and each HWE can generate multiple waves. The waveform information is stored in a local on-chip memory, and the shape of the waveform is regenerated in real time for each frame. This eliminates the need for writing to the external graphics memory when updating the wave, and thus eliminates extra memory bandwidth requirements.

There is one uniqe on-chip memory for storing data for each waveform. The waveforms handled by one single HWE unit cannot overlap each other. One wave is displayed below the previous wave. The waves from different HWE units can overlap each other, though.

SPIGPU

Data compression will reduce storage size and transfer times. The SPIGPU can decompress and transfer data from storage memory into the graphics memory. There are versions for parallel memory or serial Flash (SPI) available.

The compression, RFC 1951, is a lossless data compression algorithm. It is called Deflate and used in applications like zip and png. The algorithm achieves very good compression results on contrast rich material. Compression ratios between 10 and 50 are common with material like picture elements, symbols and font tables.

The SPIGPU is complemented with a desktop SW utility, Deflate4BADGE. This helps the user to compress the data files before they are loaded on to the target system.

User Interface and Programming Model

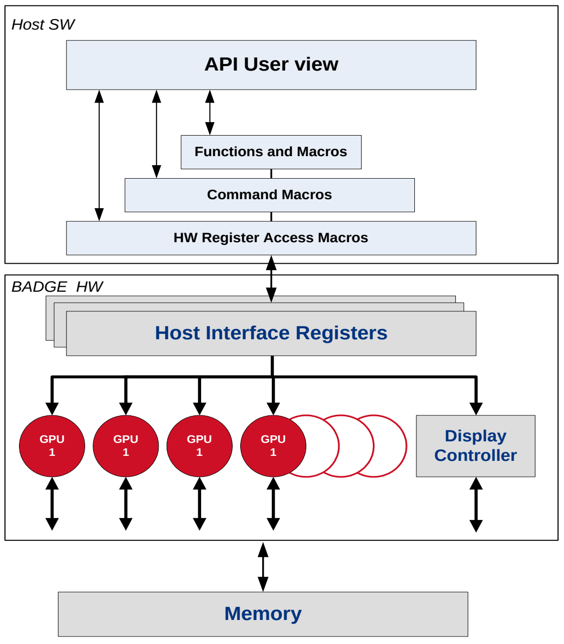
BADGElib

BADGElib is a C-based API. This software basis can be used both for applications and for driver development. Primarily BADGElib contains all BADGE commands, but also all low level details like the hardware registers definitions, register field definitions, command constants, etc.

BADGElib contains functions that allows the user to perform graphics operations like drawing lines and rectangles. API calls for transferring data to, from, or within the graphics memory are part of BADGElib. System start up and initialization are made easy with the higher level API calls.

BADGElib works both stand-alone and in an OS evironment.

Extensions of BADGElib is a straight forward way to implement any fully accelerated, partially accelerated or non-accelerated operation.



Register model

Three registers are visible to the host processor and the user. Through these registers the information are provided to and obtained from the Host.

The registers are:

GCAR - GPU Command and Access Register

Provides command-based accesses to and from the GPUs

SCR - Status and Control Register

Enables monitoring of active GPUs and interrupt control.

ISR - Interrupt Status Register

Defines the interrupt source.

OS Support

Drivers for Windows CE 4.2, 5 and 6 are available. Linux drivers can be supplied.

Graphics Libraries

High End Graphics Libraries are supported through Qt/Qtopia.

Tools – BADGE utilities
To simplify for the user, there is a set of utilities that enables graphics design with standard desktop tools. Graphics and pictures can be designed, generated and manipulated in a standard PC environment. The files are then converted and transferred to the Reference Design, BADGER, or to the target system with the help of the utilities.

Fnt2BADGE.
Font to BADGE converts True Type or Free Type fonts to the format used by BADGE. The use of standard fonts and Unicode makes it easy and practical to handle any language in the targeted embedded system.

Pic2BADGE
Pic2BADGE converts bmp or png files to the format used by BADGE. Similar to the font handling, the use of standard picture formats makes it simple to develop any picture material to be used in the target design, or in the embedded system.

Deflate4BADGE
Deflate for BADGE compresses data files, before they are loaded onto the target system or in the embedded system.

The Deflate algorithm (RFC1951) performs lossless data compression. It has proved stable, fast, and gives very good compression for contrast rich material, like text and symbols. RFC1951 is used in file formats such as ZIP and PNG.

BADGE Reference design – BADGER (Obsolete)

The BADGER Reference design makes it easy to evaluate and demonstrate the intended use very early in the development cycle. Users can try out the functionality of the BADGE IP, the user interface, mimic the target graphics and run demonstrations of the intended use. The reference design, with schematics etc., gives an ahead start for the hardware design of the target system.

The BADGER kit includes HW and SW. The hardware consists of a PCB, sample display, power supply and cables. The board is delivered with an evaluation version of BADGE loaded in the FPGA. On power-up a standard demo is started. This gives the design team a flavor of BADGE’s potential.

As SW the kit includes BADGElib for easy control of the graphics and the BADGEutilities to facilitate the creation of content.

With the Host processor running a boot monitor and executing BADGElib functions, a user can control the execution and build up a demonstration of the target system. This process can be started before any hardware development have been completed, or even begun.

Features of the BADGER board: Cost effective, feature rich FPGA.

Host Controller with peripherals: Atmel ARM9, AT91RM9200. Flexible communication and debug interfaces. Ethernet, USB, RS232, JTAG.

Connectors for three Video signals directly to the board. Connectors for Displays, GPIO, debug and extensions. Over 200 possible I/Os. Versatile memory interface: 28-256 Mbytes 133MHz SDRAM, 2 + 8 Mbytes Serial FLASH.

Basic IP configurations

Configuration	Description
BADGE Lite	The Display-Controlling device. Host Processor rendering. Pixel-by-pixel access, HW Cursor. Modules: Host Interface, Memory Controller, Display Controller, and MDAGPU.
BADGE 2D	Adds text & 2D Acceleration to BADGE Lite Modules: Host Interface, Memory Controller, Display Controller, MDAGPU, SPDGPU, RCCGPU, CHRGPU
BADGE Video	Adds Video to BADGE Lite Modules: Host Interface, Memory Controller, Display Controller, MDAGPU, VPU
BADGE 2D Video	Adds both text & 2D Acceleration and Video to BADGE Lite Modules: Host Interface, Memory Controller, Display Controller, MDAGPU, SPDGPU, RCCGPU, CHRGPU, VPU

IP Core configurations and adaptation

In addition to the basic configurations, a number of options and alternatives are possible. Literally any combination of the 2D GPUs and the optional Processing Units (SPIGPU, VPU and HWE) are possible.

The GPUs, Processing Units and Controller blocks have features that can be included or excluded in the customer design. Some examples: Bus width and memory type can be configured for the Memory Controller. The Host Interface can be adapted to fit any Host processor bus. The VPU can be configured to include or exclude rotation and scaling, etc.

IP Core modifications

BADGE can be scaled and modified to meet a wide range of needs. The result is highly optimized implementations meeting targets for high performance, or highly compact solutions. Contact BitSim AB to discuss adaptations and modifications.

Examples:

- New Host interfaces: new CPUs, PCI, Serial Interfaces, etc.
- □□New memory or display interfaces
- New Processing Units – adding new functionality
- □□Multiple displays or non-standard displays

Technology

BADGE is proven with Altera Cyclone-families, Xilinx Spartan- and Kintex-families and ASIC technologies. Adaptation to other target technologies can be done on request.

BADGE

BitSim Accelerated Display Graphics Engine

More Information

BitSim http://www.bitsim.com

Document Revision History

Release revision M.