

CAST

RBC

High Performance Raster-to-Block Converter Core

Digital image acquisition devices, both static and video, produce image samples on a line-by-line/pixel-by-pixel basis; a scheme well known as raster scan. On the other hand many image processing-transform algorithms work on a block-by-block basis. Typical image processing examples of this kind include types of the $N \times M$ image processing filter matrices such as smoothing filters, edge detection filters, noise reduction filters etc.

In the image transform context a well known example is the 2D-Discrete cosine transform (2D-DCT), especially the 8×8 block 2D-DCT, found among others in the MPEG video compression and in JPEG image compression. Streaming applications and applications that cannot afford a full frame buffer but still wish to use such a block-by-block based algorithm face the need of on-the-fly conversion from raster scan pixels to blocks. Our RBC Raster-to-Block converter core is designed to be the perfect standalone and on-the-fly conversion solution for the JPEG image compression algorithm. Its use can be extended also to other applications that need to work on rectangle pixel blocks.

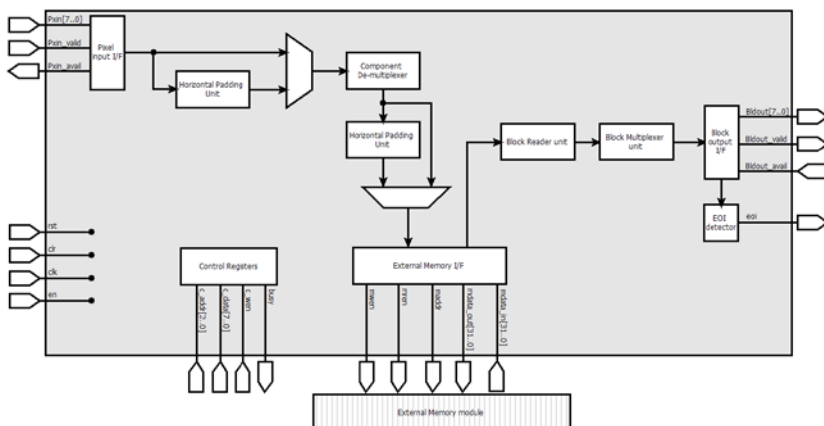
The RBC is a microcode-free design developed for reuse in ASIC and FPGA implementations. The design is strictly synchronous with positive-edge clocking, no internal tri-states and a synchronous reset; therefore scan insertion is straightforward.

Applications

The core is suitable for implementing a wide range of applications, including:

- Image compression applications using JPEG compression algorithm
- Digital camcorders
- Digital cameras
- Surveillance applications
- Scanners

Block Diagram



Features

- Raster scan to JPEG MCU block order
- Full streaming support
- Supported component sampling factors
 - ▶ 4:4:4
 - ▶ 4:2:2
 - ▶ 4:1:1 (horizontal)
 - ▶ 4:4:4:4 (CMYK)
 - ▶ 1:0:0 (grayscale)
- 8 bit sample precision
- Padding
- Sustained per cycle operation
- Does not insert extra idle cycles and compensates host stalls – perfect for video encoders
- High throughput – over 100 MSamples on FPGA platforms
- Standalone operation
- Fully stallable interfaces
- Low power standby mode – global synchronous register enable
- Hardware proven

Functional Description

The Control Interface is used for programming the operation mode and the conversion-related parameters such as image dimensions, sub-sampling format etc. There is one dedicated interface for each type of data the core accepts or produces, so there are a total of two data I/O interfaces; the Pixel input and block output interfaces. In conjunction with these two interfaces and the control registers interface that is used to program the core there are also the external memory interface, through which an external memory module that stores conversion intermediate results is accessed and the system interface.

Core accepts the raster scanned pixels through pixel input interface. Depending on the programmed image dimensions and pixel sub-sampling format the core, if necessary, performs horizontal image padding by repeating last row sample of each component. Components are then de-multiplexed and if necessary vertical padded by repeating last image row for each component. Pixel samples are then stored in the external 16-line double buffer memory. Stored pixels are finally retrieved on a block-by-block basis and they are fed to the block output interface.

Implementation Results

The following are typical performance and utilization results using a variety of implementation technologies.

Asic Silicon Vendor	Technology	Eq. Gates	f _{MAX} (MHz)
UMC	0.18 u	19,500	250
ATMEL	0.18 u	15,000	250

Support

The core as delivered is warranted against defects for ninety days from purchase. Thirty days of phone and email technical support are included, starting with the first interaction. Additional maintenance and support options are available.

Verification

The core has been verified through extensive simulation and rigorous code coverage measurements.

Deliverables

The core includes everything required for successful implementation:

- VHDL or Verilog RTL source code
- Post-Synthesis EDIF (netlist release)
- Sophisticated self-checking Testbench (Verilog versions use Verilog 2001)
- Simulation and synthesis scripts
- Place & Route scripts (netlist release)
- Documentation & Design Support