

CAST

ECP_Slave

Extended Capabilities Parallel Port Slave Core

Features

- Compliant with the IEEE 1284-2000 parallel interface protocol standard
- Supports common asynchronous transfer modes:
 - ECP Mode — fast, bidirectional, byte-wide transfer
 - Compatibility Mode — forward transfer (host-to-peripheral) using the original “Centronics” standard
 - Nibble Mode — Reverse (peripheral-to-host) transfer, four bits at a time, with broad compatibility.
- Device ID strings in ECP and Nibble modes support Plug & Play operation
- Performs Run Length Encoding (RLE) decompression in the forward direction for faster transfer of large, regular data (e.g., raster image files)
- Includes Software Development Kit with sample software for Win NT/2000/XP operating systems
- Technology-independent HDL source code

Implements an Extended Capabilities parallel Port (ECP) that makes a peripheral compliant with the IEEE1284-2000 specification.

The ECP is an asynchronous, byte-wide, bidirectional channel between a peripheral and a host computer. The ECP_Slave core implements the fast ECP mode, and also the forward Compatibility and reverse Nibble modes (useful together for bidirectional support of older devices). It supports Device ID strings in the Nibble and ECP modes, thus permitting the identification of the peripheral on Plug & Play operating systems.

The core also implements run length encoding (RLE) decompression on ECP forward direction transfers, providing real-time data compression ratios up to 64:1 and especially useful for transferring raster images with long strings of identical data.

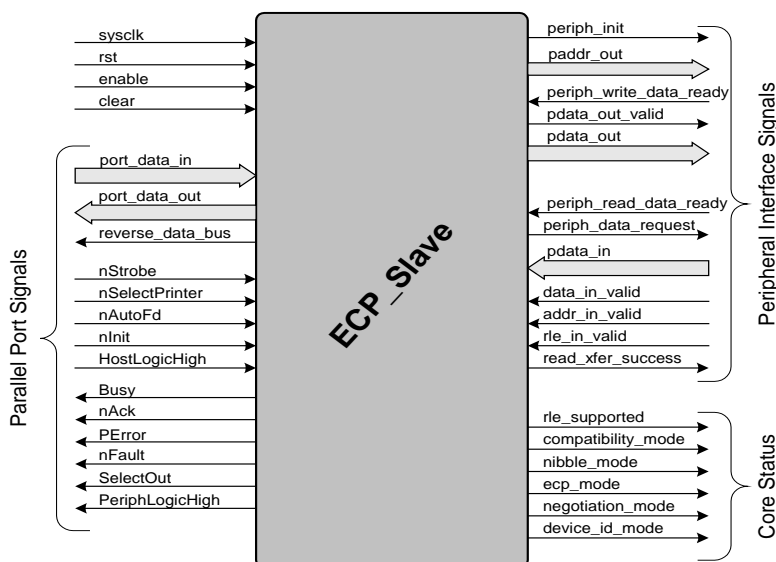
The ECP_Slave core has been developed for reuse in ASICs and FPGAs and is a fully synchronous design. It has one global clock domain, no latches, and tristate enable signals are provided to combine input and output buses to a single bi-directional bus.

Applications

The ECP_Slave core is ideal for applications that need a bi-directional connection to a PC at medium data transfer rates, with low complexity and low development cost. Examples include:

- Printers, Scanners, Multifunction Devices
- Instruments
- Experimental devices
- Data acquisition systems

Symbol Diagram



Functional Description

When the host sends data to the peripheral, a transaction in the *forward* direction is taking place, and when the peripheral is sending data to the host, a *reverse* data transfer is taking place. The parallel port modes supported by the core and the direction of data flow as specified in the IEEE1284 standard, are shown here.

Parallel Port Mode	Forward Direction	Reverse Direction
Compatibility mode	⌚	
Nibble mode		⌚
ECP mode	⌚	⌚

According to the IEEE-1284 standard, the ECP Host Controller is the master of the communication between the host and the peripheral. The ECP master initiates forward transfers and controls the direction of the channel but the peripheral (the ECP_Slave core) is responsible for initiating reverse transfers when the channel has been reversed by the master.

There are many cases where it would be more convenient if a reverse transfer was initiated by the host, e.g. when reading the contents of a status register in the peripheral. The interface of the core with the peripheral simplifies this operation, by giving the peripheral the impression that reverse transfers are initiated by the host. This does not take back the capability of the peripheral to initiate reverse transfers. The ECP_Slave core handles the task to initiate reverse transfers by requesting a byte from the peripheral every time data is available, so compliance with the IEEE1284 standard is preserved and a convenient interface to the parallel port is provided to the peripheral.

Channel Addresses

A channel addressing scheme is provided from the IEEE-1284 standard for ECP Mode, providing 128 forward and reverse channel addresses. The channel addresses may be dynamically changed while in ECP Mode. Channel address definitions are application specific. The host and peripheral channel addresses default to zero after each negotiation from Compatibility Mode to ECP Mode. After a channel address command is issued from the host, that address becomes the current channel address, present at the `paddr_out` bus, for both host-to-peripheral and peripheral-to-host communications, until another channel address command is issued or until termination.

ECP mode RLE Data Compression

ECP mode supports single-byte Run Length Encoding, which compresses strings of identical bytes. The compression is particularly useful on raster imaging devices. The core implements an RLE decoder for data transfers in the forward direction, to be able to decompress data sent by RLE-capable ECP hosts.

RLE in the reverse direction is supported, but an RLE encoder is not implemented so the peripheral logic should generate the RLE run-lengths to be transmitted by the core towards the host. In this case the ECP host must support RLE, which is indicated by the signal `rle_supported`.

Implementation Results

ECP_Slave reference designs have been evaluated in a variety of technologies. The following are sample results using describe constraints and details for getting results and optimization for speed.

ASIC Technology	Approx. Area	Frequency (f _{max})
Atmel 0.18μ process	2,000 gates	>300 MHz

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 ASIC version of this core includes everything required for successful implementation:

- HDL RTL source code
- Sophisticated self-checking Testbench (Verilog versions use Verilog 2001) including everything necessary to test the core
- Scripts for simulation and synthesis
- A Software Development Kit for Win2000/XP operating systems
- Comprehensive user documentation, including detailed specifications and design integration guidelines