AN INTEGRATED HIGH-PERFORMANCE FASTBUS SLAVE INTERFACE CIRCUIT(*)

J. Christiansen and C. Ljuslin
CERN, Geneva, Switzerland

ABSTRACT

A high-performance Fastbus slave interface ASIC (application-specific integrated circuit) is presented. The Fastbus slave integrated circuit (FASIC) is a programmable device, enabling its direct use in many different applications. The FASIC acts as an interface between Fastbus and a "standard" processor/memory bus. It can work stand-alone or together with a microprocessor. A set of address mapping windows can map Fastbus addresses to convenient memory addresses and at the same time act as address decoding logic. Data rates of 100 MB/s to Fastbus can be obtained using an internal FIFO (first in/first out) buffer in the FASIC device.

Submitted to the Nuclear Instruments and Methods in Physics Research, section A

(*) The project has been partly funded by the Italian LAA Project directed by A. Zichichi.

Mac/Word/6011F/JC/sas
1. INTRODUCTION

Since its introduction, the Fastbus standard has been used in many high-energy physics (HEP) experiments. The Fastbus has however not been widely accepted by industry, and very few integrated Fastbus interfaces are available. To ease the task of maintaining, upgrading and building new Fastbus-based data acquisition systems, a Fastbus slave interface chip has been designed.

The module interface of the FASIC is very flexible, so it can be used in many different applications with very little external logic. The FASIC can be used stand-alone or controlled/monitored by an on-board microprocessor. Connected to a microprocessor it is capable of passing information to the processor about data transfers performed from Fastbus. A typical Fastbus module configuration using the FASIC is shown in fig. 1.

![Diagram of Fastbus module using FASIC](image)

**Fig. 1** Typical Fastbus module using FASIC.

2. ARCHITECTURE

The Fastbus port seen in fig. 2 handles the Fastbus protocol. The FASIC supports all IEEE 960 (1986)(*) addressing and data transfer modes, and complies with the standard. When the FASIC is primary address selected, data transfers on Fastbus can access internal CSR (control status register) or initiate transfers to the processor/memory bus (P-bus). If errors occur in any transfer, the FASIC returns the corresponding error status to Fastbus.

The P-bus port accesses external CSR registers or memory when Fastbus data transfers to the slave are performed. A processor connected to P-bus can also access internal control/status registers via this port.

The interface between the Fastbus port and the P-bus port contains internal Fastbus CSRs and the necessary logic to map a Fastbus data transfer to a memory access on the P-bus.

![Diagram of the FASIC architecture]

**Fig. 2** The FASIC architecture.

### 2.1 Internal CSR

The FASIC implements the following CSRs: CSR0, CSR3 and CSR7. The CSR0 contains all the main control/status bits as defined by the Fastbus standard. Some of the bits have application-specific functions, and these are connected to a set of programmable CSR0 pins. The CSR3 and CSR7 are included to support logical and selective broadcast class addressing.

### 2.2 Address mapping windows

The Next Transfer Address (NTA) from Fastbus is mapped into a P-bus address by 6 programmable address mapping windows. The mapping windows enable any Fastbus address to be mapped into convenient P-bus memory locations as shown in fig. 3.

The address mapping capabilities can also be used to simplify the address decoding logic for P-bus. Each window can be assigned its individual select bit from one of the most significant P-bus address bits. This bit is then asserted in one address window only, and can therefore be used directly as a chip select for external memory/registers.

All address mapping windows are continuously monitored by a window monitor. When a Fastbus transfer to a window has been performed, the FASIC can generate an
interrupt to a microprocessor. The microprocessor can then read a status register in the FASIC to get more detailed information about the access performed.

![Diagram showing address mapping from Fastbus to P-bus.](image)

**Fig. 3** Address mapping from Fastbus to P-bus.

### 2.3 Message protection

The Fastbus standard defines 16 interrupt message receiver blocks of 16 words each. The FASIC is capable of protecting these so they are not overwritten from Fastbus before they have been read by a processor on the P-bus. One message can be stored internally in the FASIC. The other messages must be mapped to an external memory. When a message has been received from Fastbus, an interrupt is given to the processor. At the same time the message block is marked as protected, and any Fastbus write to this block returns the Slave Status (SS) = 1 error code. The processor can then read a status register in the FASIC to find the location of the new message. After reading the message, the processor must release the protection by writing to a control register.

### 2.4 Internal data buffer

One of the main design goals of the FASIC was to obtain a very short access time from Fastbus to data in memory on the slave module. Especially block/pipelined reads are critical in large HEP experiments. In a fully handshaked protocol, like Fastbus, data transfers tend to be slow if all actions are made sequentially: master asserting Data Sync (DS), DS propagates on segment, slave receives DS, slave checks access mode, slave accesses local memory, slave drives data on Fastbus, slave asserts data acknowledge, etc. The largest delay contribution from the slave is normally the access to memory. Performance can be improved by using very fast and expensive memory, or by using a buffer to pipeline data between Fastbus and the actual memory. The two buses become decoupled when a pipeline buffer is used, and the two ports can thereby work concurrently. A 4-word deep FIFO has been used as a pipelined buffer in the FASIC.

The use of such a buffer also enables the use of a "standard" memory bus, instead of a dedicated high-speed memory interface.
The FASIC takes advantage of the address mapping windows when managing the read/write buffer. The address mapping windows are used to check if an address is legal when storing Fastbus data in the buffer and when performing read ahead from P-bus to the buffer. Even though an address is contained in an address window, the access to memory may still fail (parity error, invalid address, etc.). For some applications it is not acceptable that an acknowledged Fastbus write can fail when data are transferred into memory from the buffer. To prevent this for critical data (normally control information), it is possible to disable buffering in each address mapping window individually.

2.5 Memory/microprocessor interface

A "standard" 32-bit processor bus has been chosen as the connection between the FASIC and the memory/microprocessor. The P-bus is a simplified version of the SPARC(*) processor bus, as defined in the Cypress implementation [1]. It is a clock synchronous bus, which is pipelined to obtain high transfer rates when blocks of data are accessed. The FASIC can share P-bus with other units, under the control of a central bus arbiter.

2.6 Fastbus interface

The FASIC is implemented in a high-performance CMOS (complementary metal oxyde semiconductor) technology and it interfaces to the Fastbus segment via external drivers/receivers/translator. This isolates the FASIC from dangerous signal spikes, etc. which may occur during handling or live insertion of modules. Implementation in an (expensive) ECL (emitter coupled logic) technology was ruled out because of the large number of 25 Ω cut-off drivers needed on the chip (power consumption, ground bounce). Also the PCB (printed-circuit board) connections from the chip to the Fastbus connector would have been longer than recommended in the Fastbus standard.

3. CLOCKING AND SYNCHRONIZATION

Modern CMOS technology is very fast when communicating internally on-chip. External signals have large capacitance loads, and are not well driven by CMOS drivers. Therefore, the I/O interface of the FASIC is clocked with a 25 MHz system clock while the internal logic runs at 50 MHz.

Fastbus being an asynchronous bus and P-bus a clock synchronous bus require that a synchronization of Fastbus signals must be performed. It is, in general, considered risky to design asynchronous integrated circuits. This is caused by the poor capability of available digital simulators to verify the functionality of an asynchronous circuit under all

(*) SPARC is a an open architecture processor defined by Sun Micro Systems Inc.
conditions. The FASIC therefore samples the Fastbus handshake signals close to its inputs. In order to obtain high transfer rates for block/pipelined reads the FASIC is capable of responding asynchronously when data are already in the read ahead buffer. It also responds asynchronously to actions in which no actual data transfers are performed: DS down to DK down for NTA and random read/write and AS down to AK down. The asynchronous respond capability is implemented with some simple and fast asynchronous logic before the sampling flip-flops. This asynchronous logic is only enabled by the synchronous parts of the chip under well-defined conditions. Its functionality is simple and it has been designed very carefully to prevent any possible timing races in the circuit. In all other cases the FASIC responds synchronously.

When sampling asynchronous signals, one has two conflicting problems. Sampling at low speed wastes a lot of time in the synchronization. Fast sampling speeds may give problems with metastability [2, 3]. The sampling flip-flops in the FASIC are clocked with an early clock as seen in fig. 4 (clock taken close to the root of the clock distribution tree) and the state machines driven by the sampled signals have been optimized to have very short delays from these inputs to their state registers. This gives the sampling flip-flops more time to resolve metastable states. The estimated MTBF of the FASIC is 600 years when performing a constant data transfer rate of 1 Mwords/s. As an extra precaution the single bit active (one-hot) encoded state machines in the FASIC are self-checking. If an illegal state is detected, the FASIC deselects from Fastbus and resets all its internal state machines.

Fig. 4  Sampling of asynchronous Fastbus handshake signals.
4. PROGRAMMING

The flexibility of the FASIC is obtained by having many programmable functions in the chip: address mapping windows, use of read-ahead/write-later buffer, CSR0 pins, internal message block, Fastbus wait timing, P-bus arbitration time out, etc.

Programming of the FASIC is achieved by loading a serial data stream using three dedicated pins: clock, data and reset. The 523 bits of programming data can be loaded from a serial PROM or from a microprocessor. Programming data in the FASIC is volatile, and it must be loaded when power has been applied. Before the FASIC is properly programmed it is completely inactive on both Fastbus and P-bus.

5. DESIGN METHODOLOGY

The FASIC was designed with a strict top-down design methodology. The architecture of the FASIC was described at a behavioural level, in the Verilog hardware description language (HDL(*). At the same time a complete behavioural model of the system surrounding the FASIC was implemented in Verilog(**) (Fastbus system with Fastbus master, P-bus memory, microprocessor, etc.). This virtual system was used to optimize the architecture of the FASIC. Next, the whole datapath (regular structures) was mapped into gates. The state machines, still described in Verilog, were then fine tuned in this mixed level environment. When all state machines were validated, at the behavioural level, they were finally mapped into the gate domain, with the help of logic synthesis tools(**).

6. IMPLEMENTATION

The FASIC has been implemented in a 1.0 μm 3-level metal HD CMOS sea-of-gates array from Motorola(***). The 25 000 gates design has been mapped into a 50 000 gates master packaged in a low-cost plastic 208 pins quad flat pack (QFP) package. The 3-level metal sea-of-gates array technology actually allows up to 75% gate utilization. In our case, the design was mapped into the large gate master, in order to fit the low-cost and high-pin count QFP package.

The design speed of the FASIC is 50 MHz internally and 25 MHz for external I/O (worst case). The first silicon produced worked perfectly. No bugs have been found in the design during several months of running and testing in Fastbus systems. The prototype chips have all worked reliably with an internal clock speed of up to 80 MHz.

(*) Verilog is a trademark of Cadence Design Systems Inc.
(**) Design Compiler is a trademark of Synopsis Inc.
(***) HDC sedries design reference guide, Triple layer metal. high-density CMOS arrays 1 μ Motorola.
The power consumption is below 0.5 W when used at design speed in normal applications, so no special cooling of the chip is needed.

7. PERFORMANCE

The performance of a Fastbus module using the FASIC depends to a large extent on the delay of the external drivers/receivers/translators interfacing to the Fastbus segment. Even so, the performance of the FASIC remains very high. As previously stated, the FASIC is optimized to obtain high transfer rates for block reads. With a fast Fastbus master initiating data transfers, the FASIC is capable of using the full bandwidth of P-bus (25 MHz P-bus = 100 MB/s). In fig. 5 the word transfer time of the FASIC (DS to DK) is shown as a function of the response time of the master (DK to DS) for block reads.

![Graph showing word transfer time as a function of master response time for block read](image)

**Fig. 5** Word transfer time as a function of master response time for block read including drivers/receivers. The FASIC connected to 25 MHz zero wait cycle P-bus (P-bus cycle time = 40 ns).

When read ahead is used the performance curve has two regions, one where the transfer time is limited by the data bandwidth of P-bus and an other where the transfer time is mainly limited by the master itself. The slave delay is very short when the master delay is above 30 ns (which is the case for most masters implemented to date). The short slave delay is obtained by the asynchronous respond capability and the read ahead buffer. When one data transfer is performed on Fastbus the next word is concurrently read into the read ahead buffer from the P-bus.

If the read ahead capability is not used it can be seen that the performance degrades significantly. In this case the FASIC does not respond asynchronously (results in a
staircase shaped curve) and Fastbus has to wait for data to be accessed in the memory on the P-bus (table 1).

<table>
<thead>
<tr>
<th>TIMING</th>
<th>AS to AK geographical addressing</th>
<th>AS to AK logical addressing</th>
<th>AS to DK read ahead data in buffer</th>
<th>AS to DK write to buffer or internal register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>43 ns</td>
<td>83 ns</td>
<td>13 ns</td>
<td>43 ns</td>
</tr>
<tr>
<td></td>
<td>(55 ns)</td>
<td>(95 ns)</td>
<td>(25 ns)</td>
<td>(55 ns)</td>
</tr>
</tbody>
</table>

**Table 1** Typical Fastbus timing of FASIC connected to a 25 MHz P-bus. The values in parenthesis include delays of typical ECL drivers/receivers.

8. **FASTBUS SLAVE INTERFACE CARD**

A Fastbus slave interface sub-card is currently being designed based on the FASIC chip [4]. This card contains the FASIC chip itself plus all the necessary drivers/translators/ receivers and a serial PROM with the programming data. It is made with SMD (surface mount device) components only and is very compact (70 mm × 170 mm). It is implemented as a piggy back module and it can be used for prototype Fastbus slave modules, and also for production series if one does not want to use SMD technology on the Fastbus module itself.

**Acknowledgements**

We would like to thank our colleagues A. Marchioro and M. Letheren for their constant support during the development of the chip. The project has been partly funded by the Italian LAA Project directed by A. Zichichi.

**REFERENCES**


