THE "XOP PROCESSOR IN FASTBUS"

J.LECOQ - M.MOYNOT - G.PERROT - P.BAHLER [*] - C.LJUSLIN [*]

ABSTRACT

Dans le cadre de l'expérience LEP 3, un interface FASTBUS Maître a été conçu pour être associé au microprocesseur XOP.

Cet article décrit la philosophie de l'interface, son fonctionnement interne et la programmation de FASTBUS par XOP. Quelques exemples sont donnés dans les différents modes de fonctionnement ainsi que les performances de l'ensemble.

A FASTBUS MASTER interface has been designed for the XOP microprocessor used in the L3 experiment.

This paper describes its design philosophy, its internal sequences and the related FASTBUS XOP software. Examples are given in the different modes and the corresponding performances are quoted.

[*] CERN - DIV.DD.

XOP is a fast microprogrammable processor developped at CERN.(1) Its reliability and its modularity allow special user extension. To implement XOP as a trigger processor in the L3 experiment we have designed a fastbus master interface which operates as an internal XOP module, running in parallel with the other modules.

This module is an XOP dedicated interface designed as an XOP extension. It consists of an XOP card linked to one Fastbus card per master (Maximum : 2 masters per interface). (2)

The Fastbus instruction is an horizontal extension of the XOP micro-instruction which is increased from 160 to 190 bits.

This new field is as independant as possible of the old ones to allow simultaneous processings.

Fast execution is obtained by the use of two internal sequencers. One is used to share the 4 typical fastbus operation on a single XOP microinstruction (i.e. arbitrate, primary address, secondary address, data cycle).

The other is used to manage a complete pipe line transfer at a programmable speed (up to 125 ns/32 bit word).

All these operating modes, timings and busses source/destination are defined in the 29 bits of the microcode described below.

Fastbus addresses, data, Control and Status are pipelined into 12 16-bits registers to allow a good synchronization between XOP and Fastbus.

Two extra registers are used to programm the word count and the speed of transfers executed in pipe line mode.

Synchronous and asynchronous modes are both available to avoid the dead time due to cycle by cycle synchronization. The synchronization of the module on XOP master clock is accessible by software via a special bit in the fastbus field. (hold)

A flag, named fastbus flag has been added to the old ones (carry, zero, overflow, sign, counter zero). This flag, testable during any other XOP action, allows a constant check of the fastbus operations.

FASTBUS OPERATIONS AND SEQUENCES

The module is activated on the fastbus side by a non zero configuration on its "Do Something" field.(Cf table 1 micro instruction XOP, fastbus field.) Fastbus operation is then executed at the fastbus speed.

On the XOP side, it is activated by a non zero configuration on its XOP connection field,(bits 28, 29) Data transfer is then executed at the XOP speed (50 ns/ 16 bit data).

Possible Fastbus actions are :

Arbitrate, Do primary address cycle, Do secondary address cycle, Do data cycle, Do release AS/AK, Do release master ship.

They can be activated one by one or in thesame XOP instruction and then shared by the sequencer.

Because Fastbus and XOP CPU are running at the same speed and simultaneously, DMA mode is not implemented, i.e. each transfer is to be programmed.

The error checking is managed via the control and the status registers . (cf. table 2).

PROGRAMMATION OF THE INTERFACE

In the XOP microinstruction, the programmation of fastbus is independant of the other XOP fields.

In the example given, the programmation of the non fastbus XOP field will be ignored to simplify the corresponding instructions.

ARBITRATION : the "Do arbitrate" order can be given at any time, if the sequencer is not busy.

If Master is enabled and Running (CSR Ø)
and Arbitration not inhibited (CSR 8, A I), the arbitration
sequence is started.

```
XOP Program  :              Do Arbitrate, hold % wait to the end of ARB

                            JMP Error if FBFLAG Set
```

Note that as arbitrate is a "slow" fastbus action, it is recommended to execute it in the asynchronous mode (no hold).

PRIMARY ADDRESS CONNECTION

This sequence is started by the corresponding bit in the sequencer.

If it is possible (mastership thrue and primary address connection false),it generates the effective sequence at the address contained in the primary address registers FBPAH and FBPAL.

XOP PROGRAM IS :

```
LD, FBPAH               load primary address  registers (high and low)
LD, FBPAL
DO PAC, HOLD            Do primary address cycle
CJMP  , FBFLAG          Jump error if fastbus flag set.
```

As the fastbus flag test can be done in parallel with the next instruction, the execution time can be estimated at 250 ns.

As the arbitrate cycle and the memory management of XOP are independant they can be executed at the same time. For example : a pipelined concatenation of the two previous examples can be executed in only 3 XOP instructions.

SECONDARY ADDRESS CYCLE

If possible, this sequence generates the corresponding fastbus signals according to the contents of the secondary address registers FBSAH - FB SAL. An error generates fastbus flag.

Possible errors are :

. No primary address connection,
. time-out,
. parity error,
. S.S. response ≠ 0.

In case of error the corresponding bits are set in the status register FBS.

The programming sequence is :

- Load secondary address registers
- Start the cycle (DO SAC) with hold
- Test the fastbus flag.

This sequence can be pipelined with the two sequences seen above to spend only one instruction more.

DATA CYCLE

The sequence is similar.

The data registers are to be loaded before a write operation or read after a read cycle.

A sequence error, a time-out, a parity error, or an S.S. response ≠ 0 will set the flag and the corresponding status.

All these sequences can be started at the same time. In this case only one hold is necessary to resynchronize the interface and all the parameters can be fetched in XOP memory during the arbitration and the other fastbus sequences.

In an example of a random data read we can perform the complete programm in 4 XOP instructions.

This program includes :

- The loading of the four address registers with value fetched in XOP data memory,
- The execution of arbitration - primary address cycle secondary address cycle - data cycle,

- The storage of the 32 bits data read,
- The check of error.

The corresponding execution time can be estimated at only 200 ns more than the fastbus execution time itself.

In block transfer mode the software loop can be reduce to one single instruction including the two memory cycle to load or store the 32 bits data and the word count management.

The execution time is then reduced up to 150 ns/data.

In pipe line mode the transfer speed and the word count are both managed by hardware.

These parameters are programmed via two registers from 100 ns to 800 ns for the data rate, up to 64 K for the word count register. In this way the maximal fastbus speed is possible, only depending on the slave speed.

In the L3 experiment this interface will be used with the multiport multievent buffer designed at L.A.P.P. (3).

This slave module uses the fastbus state data transfer protocol chip "DATPRO" designed at L.A.P.P.(4) and now running.

With this fast coupler associated with fast ECL memories we hope to obtain transfert speed in pipe line mode up to 125 ns/word.

MICRO INSTRUCTION XOP

Fastbus Instruction

0 1 2	Ms code : Primary address
3 4 5	Ms code : Data cycle
6	R/W Secondary address cycle
7	R/W Data cycle
8	Hold
9	Do arbitrate
10	Do primary address
11	Do secondary address
12	Do data
13	Do release AS/AK
14 15	Release Mastership code

Source destination register-busses

16 17 18 19	Register code during first 50 ns
20 21 22 23	Register code during past 50 ns
24	R/W first 50 ns
25	R/W past 50 ns
26 27	Bus code first 50 ns
28 29	Bus code past 50 ns

00	No connection
01	RD bus
10	WR bus
11	No connection

TABLE 1

XOP Status Register

0	SS code or arbitration status or sequence status
1	
2	
4	Sequence error
5	Time out error
6	Time out wait state
7	Parity error
8	Sequencer state
9	
10	
11	SR
12	RE
13	BUSY
14	ERROR

XOP Control Register

0	Master 1/2
1	Reset Master Interface
2	Reset Bus
3	Parity enable
4	non stop on parity error on data cycle
5	non stop if ss=2 on data cycle
6	" " ss=3 " "
7	" " ss=6 " "
8	" " ss=7 " "
9	Set EG

PIPELINE Counter Register

16 bits ⇒ 65 K words

PIPELINE Transfer Speed Register

5 bits ⇒ 100 ns to 800 ns in 25 ns steps.

TABLE 2

REFERENCES

-o-o-o-o-o-o-o-o-o-

1 - Tor LINGJAERDE, C.LJUSLIN - XOP, main documentation internal report - 1983,

2 - J.LECOQ, M.MOYNOT, G.PERROT - XOP - Fastbus master interface F 682 C internal report - 1985,

3 - J.LECOQ, M.MOYNOT, G.PERROT - Multiport Multievent Buffer - F 682 B internal report - 1985,

4 - H.BONNEFON, M.MOYNOT, G.PERROT, JM.THENARD - An ECL Fastbus slave coupler - internal report January 1985.

=o=o=o=o=o=o=o=