SPACEWIRE VHDL CORE IP PRESENTATION

INTRODUCTION

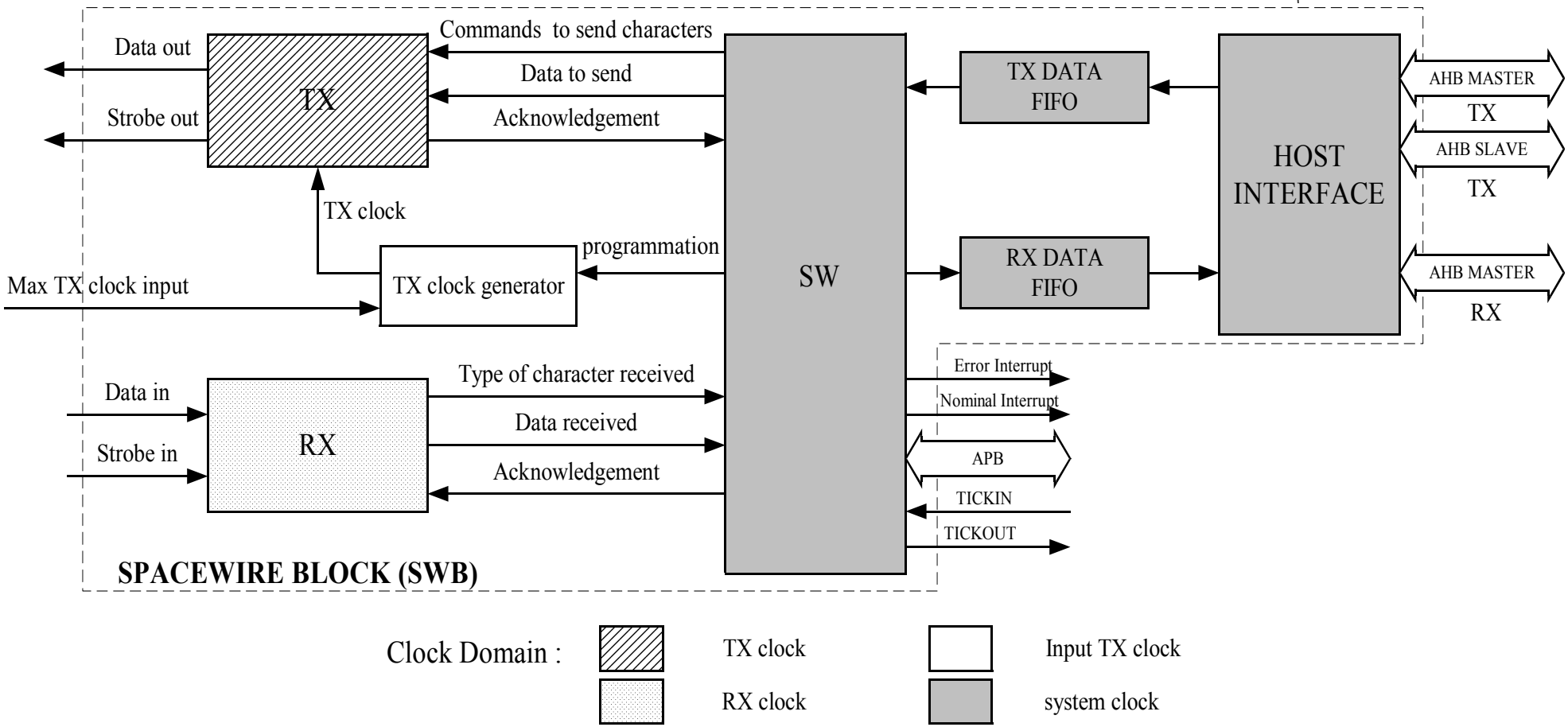
- The SW IP was developped in the frame of the ESA 13345/#3 contract "Building block for System on a Chip"

- This presentation describes the SpaceWire Block developed as part of the ScoC project. It presents the interfaces, the architecture of the block and the performances

- The SpaceWire is a serial high speed link compliant with the ECSS-E-50-12 specification from ESA

- The SpaceWire block interfaces AHB and APB interfaces within the SCoC and LEON processor architecture

BLOCK-DIAGRAM



DESCRIPTION OF THE DIFFERENT BLOCKS

- The Host Interface block is an interface with the AMBA AHB and APB buses. It contains the management of the data sent by the host. It manages the storage of data into the host memory
- The TX Data FIFO block is a FIFO containing the data to be transmitted
- The RX Data FIFO block is a FIFO containing the data to be stored into the host memory
- The SW block manages the initialisation protocol. This block selects the character to be transmitted and checks any error occurrence
- The TX block sends the character at the transmission frequency
- The RX block identifies the received character type
- The TX clock generator block generates the clock transmission rate.

INTRODUCTION TO THE INTERFACES

- The basic interface contains CLOCK, TEST and RESETN signals
- The APB interface is used to configure the SWB and to retrieve status
- The TX AHB master interface performs the TX DMA
- The TX AHB slave interface is used when the data transmission is in charge of the host
- The RX AHB master interface performs the storage of received data into the host memory
- The link interface brings together the data and strobe signals of the transmission and the reception.
- The Time interface manages the transmission and the reception of time code
- The Interrupt interface is used to warn the host when a specific event appears.

EMISSION

- The SWB receives packets of data from the host through the AHB interface. Two modes are possible for this data transfer:
  - AHB master mode, which performs the transfer with a chained capability DMA mechanism
  - AHB slave mode which performs the transfers with the length of the packet and the 32-bit data of the packet
- The 32-bit data is split to 9-bit data to be stored into the TX data FIFO
  - The 9-bit data is composed of 8 bits of real data and 1 bit for particular character such as EOP and EEP
- When the TX module receives a command from the SW module, the character corresponding to the command is transmitted through the LVDS link (Data and Strobe)
  - The TX module automatically transmits NULL characters when no other transmission is requested
- The transmission frequency is programmable through the APB interface.
  - The TX clock generator creates the required TX frequency, which can be up to 4 times the system clock frequency

RECEPTION

- The RX module performs the recognition of the received character type.
  - The RX clock is built from the data and strobe input signals
  - The RX module also indicates the received characters to the SW module
- Each time that information of character type is received from the RX module, the SW module generates an acknowledgement.
  - A 9-bit word is stored into the RX data FIFO when a data is received.
  - The SW module also activates the TICKOUT signal when a right time code is received
- When the RX data FIFO is not empty, the host interface fetches its 9-bit data. Each time four 9-bit data are available, the host interface builds a 32-bit word from these four 9-bit data and stores it into the host memory through the AHB bus
- When any error is detected from the AHB transfer or from the transmission link, the SWB generates an error interrupt to warn the host.
- The configuration of the SWB is done through the APB interface.

FUNCTIONAL MODES

- The SWB supports the following functional modes:

- RESET mode
  - TX and RX blocks are inactive
  - Host interfaces are inactive

- ACTIVE mode
  - TX block is inactive and RX block is active (when entering the ACTIVE mode)
  - Host interfaces are always on

PERFORMANCES and CONTACTS

- Gates number based on XILINX XCV2000E: 1800 LUT and 1040 flip-flops
  - Expected ASIC gates: between 15000 and 20000 gates

- SW bit rate: Around 30 Mbps at 40 MHz based on XILINX implementation (speed limited by the XILINX)

- Contacts:
  - Jean-François COLDEFY:
    - TEL: 33 1 39452623
    - EMAIL: jean-francois.coldefy@astrium.eads.net

A3M APPLICATION

A3M Context and Objectives

- Develop middleware building blocks supporting the distribution of software functions

  Consensus and coordination services ensuring consistent and reliable decisions

  Implementation of innovative algorithms developed by INRIA (UCS and UCN)

- Maintain real time and fault tolerance properties

  Limit the overhead brought by the distribution scheme without sacrificing the global fault tolerance

- Implement asynchronous algorithms, i.e. not relying on synchronized distributed nodes

  An essential merit of asynchronous algorithms: their logical behaviour (safety and liveness properties) is fully independent of the underlying timings

  For this reason, the functional testing complexity is orders of magnitude smaller than that of synchronous algorithms

A3M Demonstration Domains

- Distributed consistent processing under active redundancy

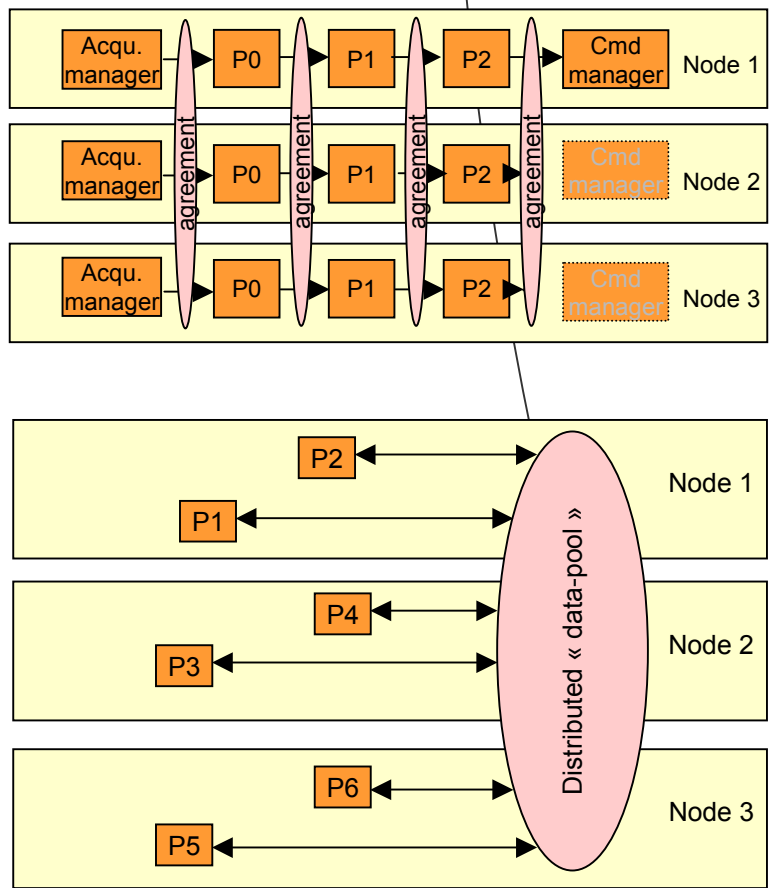
  Some applications require active redundancy with error masking, in order to ensure a very high level of dependability.

  Most solutions implemented today relies on synchronous mechanisms (the computers must be synchronized)

- Distributed replicated data consistency, program serialization & program atomicity

  Classical OBSW includes a standardized mechanism to share / exchange data between applications : the data pool

  The reliable implementation of such data pool function is a key to distribution



A3M Demonstration Platform

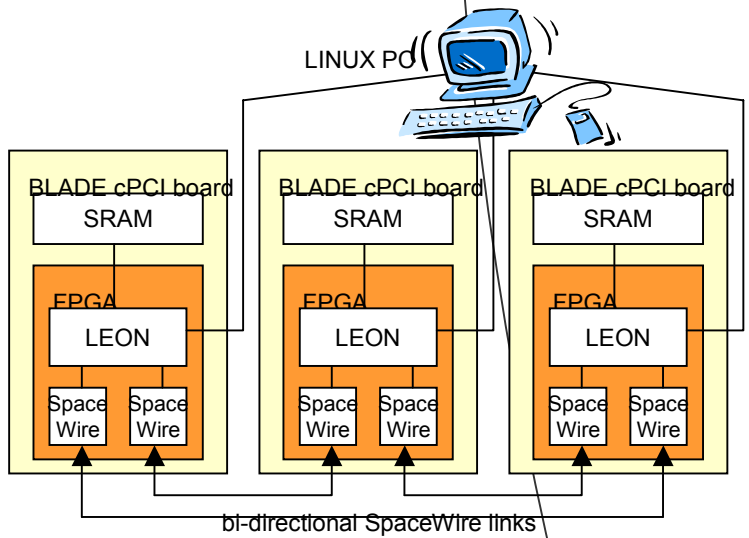
- The hardware platform has been built on ESA standards for near future missions

  LEON SPARC microprocessor, running at 20 MHz

  SpaceWire high speed communication links, up to 4 times the processor frequency (80 Mbit/s)

- POSIX 1003.1b/1.c standard API has been selected for the software platform

  Practically, Tornado/VxWorks environment allowed an incremental development (simulator, commercial h/w, final hardware platform)

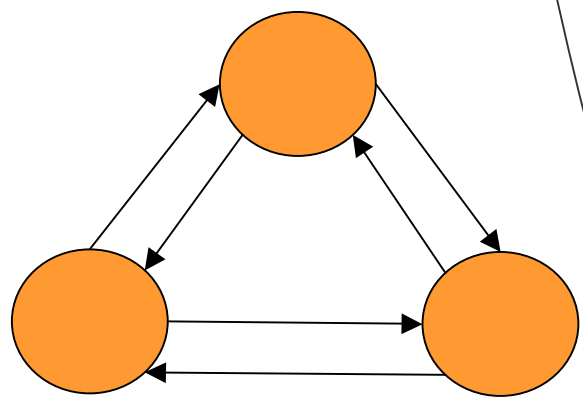


A3M Results related to SpaceWire (1)

- Memory-to-memory data transfer performance

3 interconnected BLADE boards with simultaneous bi-directional exchanges							
FREQ_RUN parameter	**0**	**1**	**2**	**4**	**8**	**16**	**32**
Raw bit rate (Mbit/s)	**80,0**	**40,0**	**26,7**	**16,0**	**8,9**	**4,7**	**2,4**
Usable data rate (Mbit/s)	**60,9**	**30,5**	**20,3**	**12,2**	**6,8**	**3,6**	**1,8**
packet size = 1024 bytes, receive buffer size = 1024 packets							
Data rate (Kbytes/s)	n/a	3 700	2 478	1 487	826	437	225
Data rate (Mbit/s)	n/a	30,3	20,3	12,2	6,8	3,6	1,8
% usable data rate	n/a	**99,5%**	**100,0%**	**100,0%**	**100,0%**	**99,9%**	**99,8%**

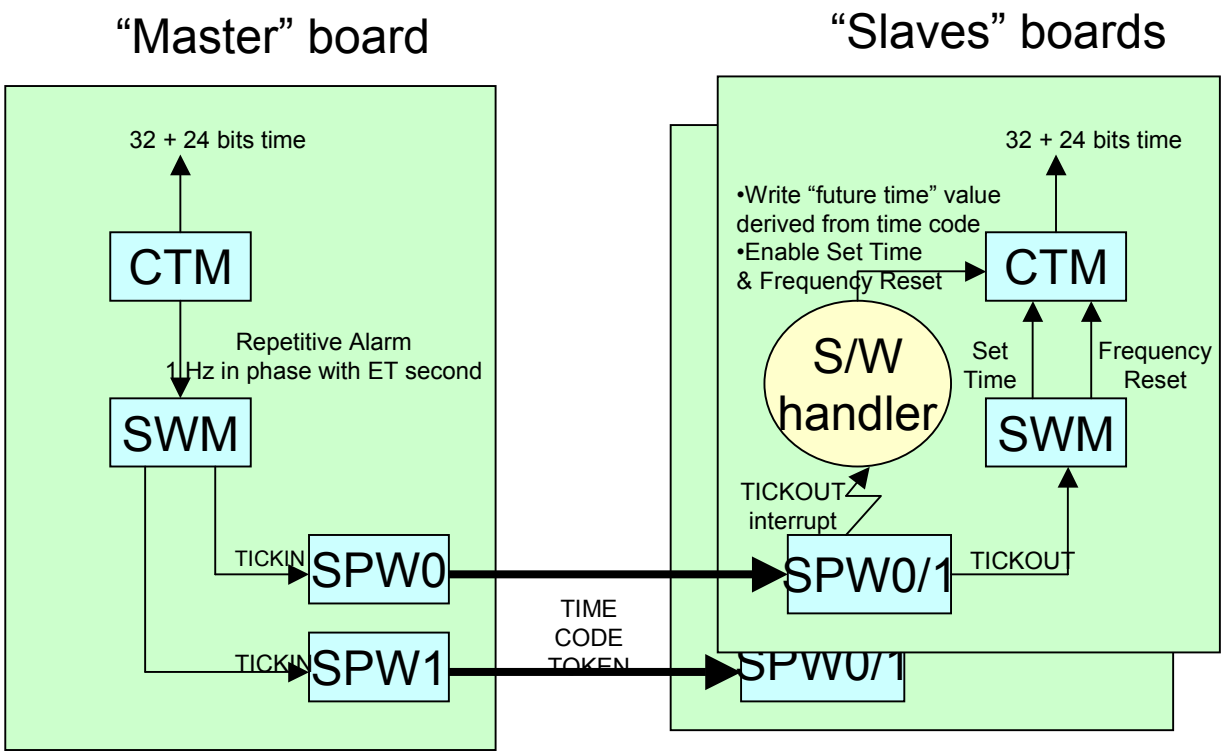
Useful data bytes	1 024	
Useful data bits	8 192	8 bits / byte
Character additional bits	2 048	2 bits / data byte
Overhead FCT	512	1 FCT token / 8 data character
Overhead EOP	4	1 EOP token / packet
Bits/packet	10 756	
Useful bits/total bits	**0,762**	



A3M Results related to SpaceWire (2)

- Time synchronization for performance measurement

  The implemented SpaceWire interface provides in-built clock synchronisation atop the SpaceWire “time code token”, used in a master/slave approach



A3M Results related to SpaceWire (3)

- The A3M middleware (UCS/UCN/Services) fulfil its objectives.

  A3M interfaces are very simple and makes the distribution transparent to the developer of applications.

  With respect to the platform used for the tests (LEON1 at 20 MHz), the performances are good.

- SpaceWire is compliant with constraints or requirements leading to the distribution of software functions.

  Very efficient inter-processor communication: with intelligent controllers like those in the demonstration board, very high data rate can be achieved from memory to memory.

  The observed limitation is at SW level. Efficient implementation of protocols atop SpaceWire requires to limit the number of SW layers (to the detriment of genericity/portability).

  The software must take into account the flow control mechanism in order to avoid overload propagation from one processor to the others.

- High precision time synchronization via the SpaceWire links with very low software overhead is very interesting, especially for distributed software performance evaluation.

Perspectives

- Optimisation of the implementation:

  Implementation of the FD mechanisms in hardware: this can tremendously decrease the maximum detection delay of a failure on the one hand and the CPU load on the other hand.

  FD mechanisms required by A3M algorithms could take advantage of the SpaceWire disconnection detection (no use of bandwidth).

  Implementation of the UCS/UCN algorithms in hardware (warning: may excess the capacity of the current space compatible FPGA or ASIC).
- Take into account a fully compliant SpaceWire network, i.e. including dynamic routers.
- Apply A3M middleware to next-to-come SpaceWire-based data handling architectures, e.g. future multi-processor scalable payload architecture.