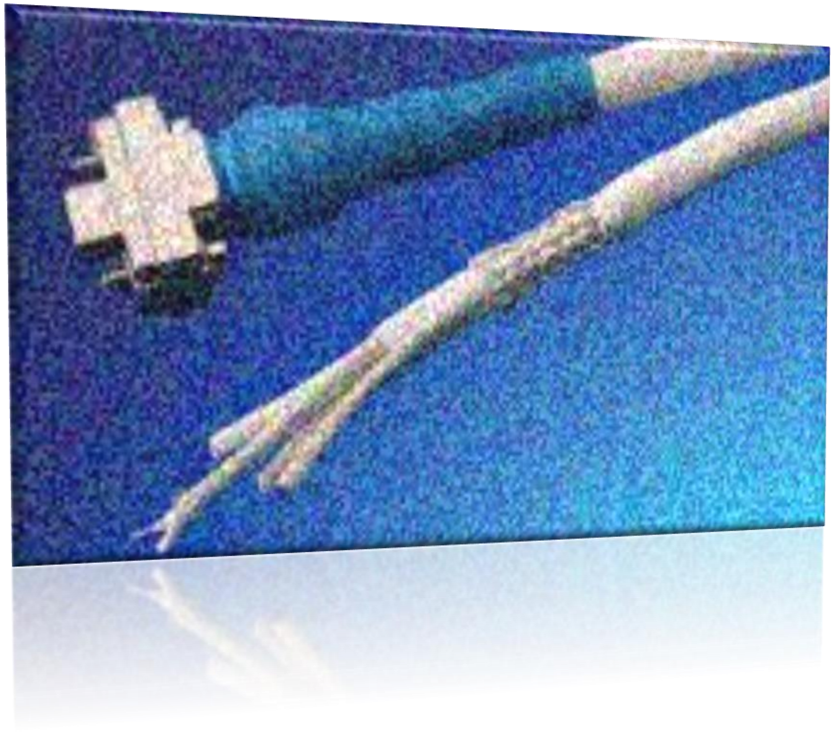
SpaceWire Node Interface IP Core



Design, implementation and validation of the SpaceWire Node Interface IP

TEC-ED & TEC-SW Final Presentation Days 2016, 8th May 2017
Noordwijk, Netherlands

Background and Objectives

- **ESA TRP Study (ESA contract number 4000113046)**
- **Develop an configurable multifunction IP-core providing:**
  - Multiprotocol SpW functionality targeted space FPGA designs.
  - A means to implement complex SpW functions rapidly.
  - A simplified interface to configure and instantiate a particular design.
  - Allow hardware support for most common SpW protocols.
- **Initial requirements regarding protocols/functionality support:**
  - SpaceWire (ECSS-E-ST-50-12C)
  - Protocol Identification (ECSS-E-ST-50-51C)
  - RMAP (ECSS-E-ST-50-52C)
  - CPTP (ECSS-E-ST-50-53C)
  - NDCP – Basic Subset (ECSS-E-ST-50-54C)
  - Time Distribution Protocol (draft specification)
  - SpaceWire – Deterministic Transfer Protocol (Target only)

Background and Objectives

- **The SpW Node IP core supports:**
  - Node with a Single SpW Port, or,
  - Node with an integrated SpW Switch -> **number of SpW ports are configurable.**
  - VHDL generics allowing instantiation of protocol engines and I/O configuration.
  - Simplified system configuration due to IP-XACT descriptions.
  - AMBA AHB/APB as baseline for inclusion of the core in SoCs
- **Synthesis and demonstration on typically used FPGA technologies**
  - Xilinx Virtex (V5 FX130)
  - Microsemi ProAsic 3
  - Support synthesis on Microsemi RTAX

Requirements

- **Definition of requirements by Thales Alenia Space and Airbus Defense and Space**
- **Desired functionality instantiated through VHDL generics**
- **AMBA based system for LEON SoC integration**
- **DMA based operation (single AHB Master)**
- **IP Core to support SpW protocol engines**:
  - RMAP with SpW-D compatible performance
  - NDCP (v1.7) accessed through AHB or SpW
  - CPTP/Raw SpW packet
  - Time Distribution protocol (TDP)
  - SpW Interrupts (ECSS-E-ST-50-12 C Rev.1)

- **SpW Switch**
  - 4-32 ports (2 – 30 external ports)
  - Regional addressing
  - Group Adaptive Routing (GAR)
  - RMAP/NDCP configuration

- **Reliability**
  - Switch/port time-out
  - Truncation if max packet length is exceeded (Rx and Tx)
  - EEP transmission on protocol engine reset
  - Discard packet if Rx pointer does not exist (programmable function)
  - Maskable interrupts on numerous functions (protocol errors, programmable number of packets Tx/Rx…)
  - Programmable max DMA size to avoid monopolizing the AHB, LOCKED transactions 8 beats
  - Nominal operation/error counters

Agenda

IP Design

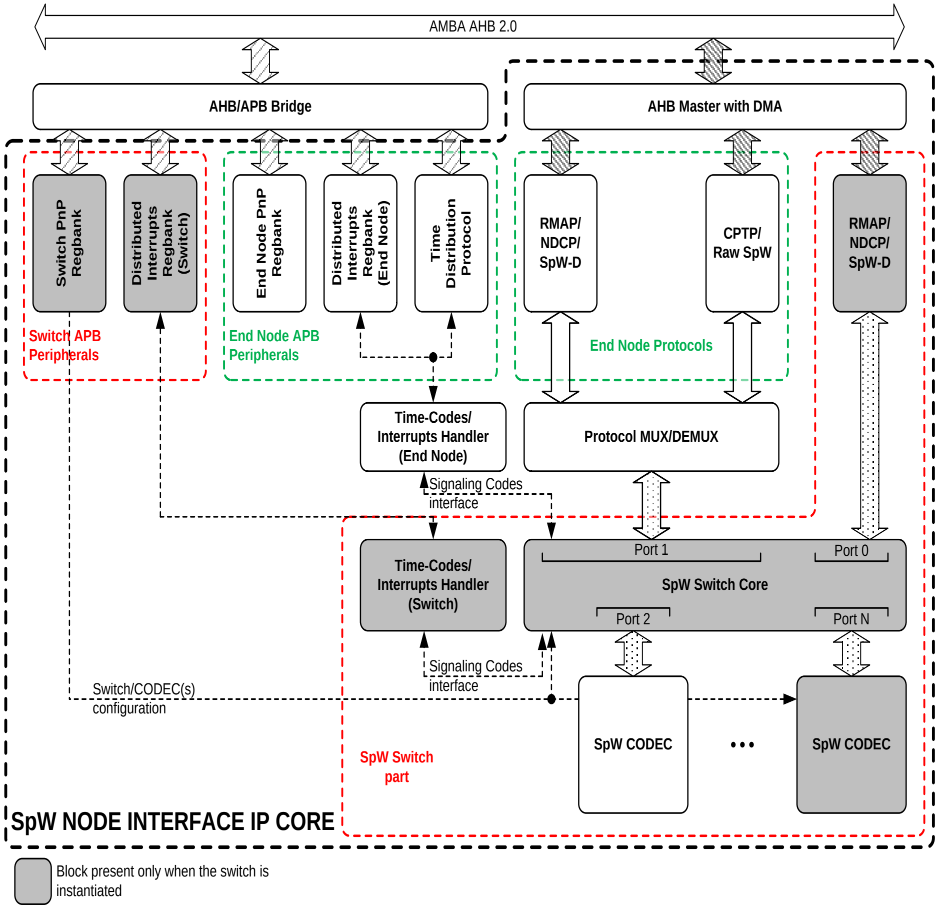
- Top Level
- CPTP/Raw SpW packets block
- RMAP
- RMAP – NDCP Extensions
- SpW Switch

Implementation, Metrics and Validation results

- Target Technologies
- Synthesis Results
- Functional tests
- Performance tests

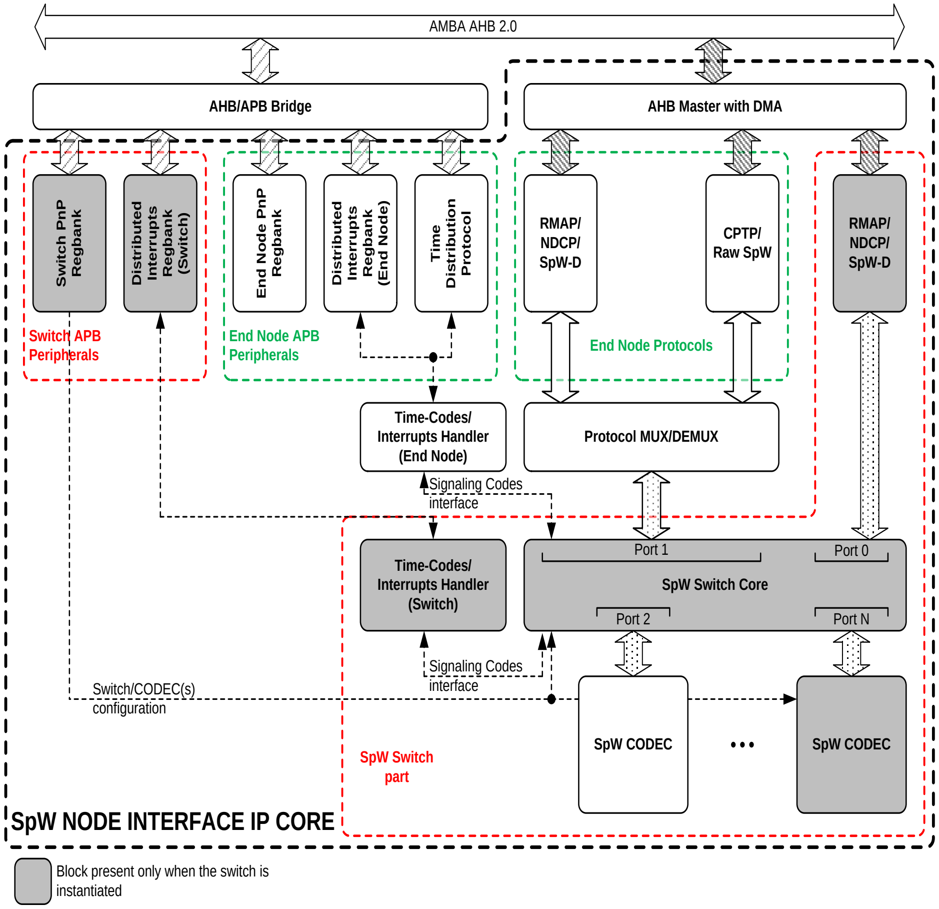
Top Level Design (1/3)

- **End Node only or EndNode with Switch**
  - SpW Node with a single link
  - SpW Node with multiple links with SpW Switch instantiated
- **SpW CODEC(s)**
  - One or more SpW CODECs (2 - 30)
  - De-facto standard 9-bits FIFO SpW CODEC interface
  - Choose from ESA SpW-b CoDec or TELETEL SpW CoDec.
- **SpW Switch**
  - Instantiated with user configurable number of ports (4 – 32)
  - RMAP/NDCP Switch configuration
- **Protocol MUX/DEMUX**
  - Discriminates/Dispatches received packets to the protocol engines, inserts EEP on truncation
  - Multiplexes packets for transmission, inserts EEP on truncation/protocol reset
- **RMAP/NDCP/SpW-D**
  - TELETEL's RMAP Core used
  - Extensions for NDCP support
  - Response time in the range of 1 usec


AMBA AHB 2.0
AHB/APB Bridge
AHB Master with DMA
Switch PnP Regbank
Distributed Interrupts Regbank (Switch)
End Node PnP Regbank
Distributed Interrupts Regbank (End Node)
Time Distribution Protocol
RMAP/ NDCP/ SpW-D
CPTP/ Raw SpW
RMAP/ NDCP/ SpW-D
Switch APB Peripherals
End Node APB Peripherals
End Node Protocols
Time-Codes/ Interrupts Handler (End Node)
Protocol MUX/DEMUX
Signaling Codes interface
Time-Codes/ Interrupts Handler (Switch)
Port 1
Port 0
SpW Switch Core
Port 2
Port N
Signaling Codes interface
Switch/CODEC(s) configuration
SpW CODEC
...
SpW CODEC
SpW Switch part
SpW NODE INTERFACE IP CORE
Block present only when the switch is instantiated

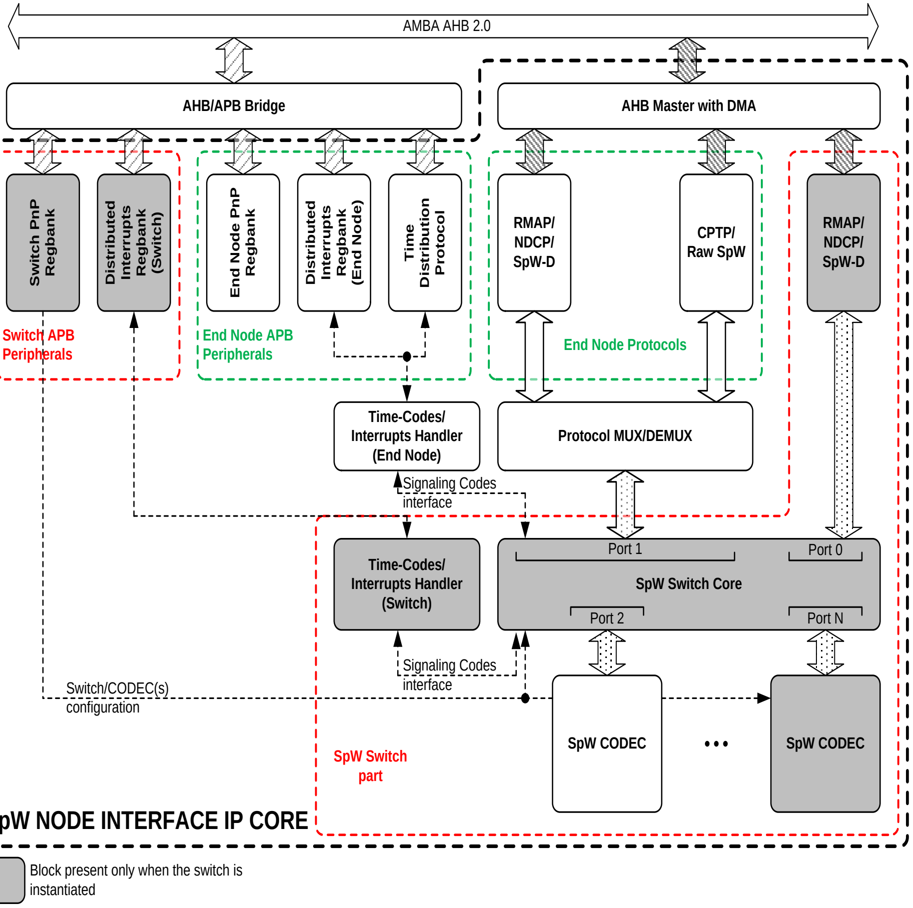
Top Level Design (2/3)

- **CPTP/Raw SpW**
  - Newly developed, Handles CPTP and Raw SpW packets (packets not handled by any other instantiated protocol engine)
  - Offloads the user logic from time-consuming operations (CRC/PEC, packet length)
- **Time Distribution Protocol (TDP)**
  - Developed by COBHAM Gaisler
  - Integrated as is for SpW Time Distribution
- **SpW Time Codes/Interrupts Handler**
  - Existing block from SpW Evolutions study
  - Handles the Signaling Codes at both the End Node and the Switch
- **AMBA based design**
  - TELETEL AHB Master handling Protocol engines DMA requests
  - DMA read/write operations to/from the system memory, APB peripherals
  - Offers user configurable number of DMA clients (depending on instantiated protocol engines)
  - Supports ATOMIC transactions required for RMAP RMW and NDCP CAS


AMBA AHB 2.0
AHB/APB Bridge
AHB Master with DMA
Switch PnP Regbank
Distributed Interrupts Regbank (Switch)
End Node PnP Regbank
Distributed Interrupts Regbank (End Node)
Time Distribution Protocol
RMAP/ NDCP/ SpW-D
CPTP/ Raw SpW
RMAP/ NDCP/ SpW-D
Switch APB Peripherals
End Node APB Peripherals
End Node Protocols
Time-Codes/ Interrupts Handler (End Node)
Protocol MUX/DEMUX
Signaling Codes interface
Time-Codes/ Interrupts Handler (Switch)
Port 1
Port 0
SpW Switch Core
Port 2
Port N
Signaling Codes interface
Switch/CODEC(s) configuration
SpW CODEC
...
SpW CODEC
SpW Switch part
SpW NODE INTERFACE IP CORE
Block present only when the switch is instantiated

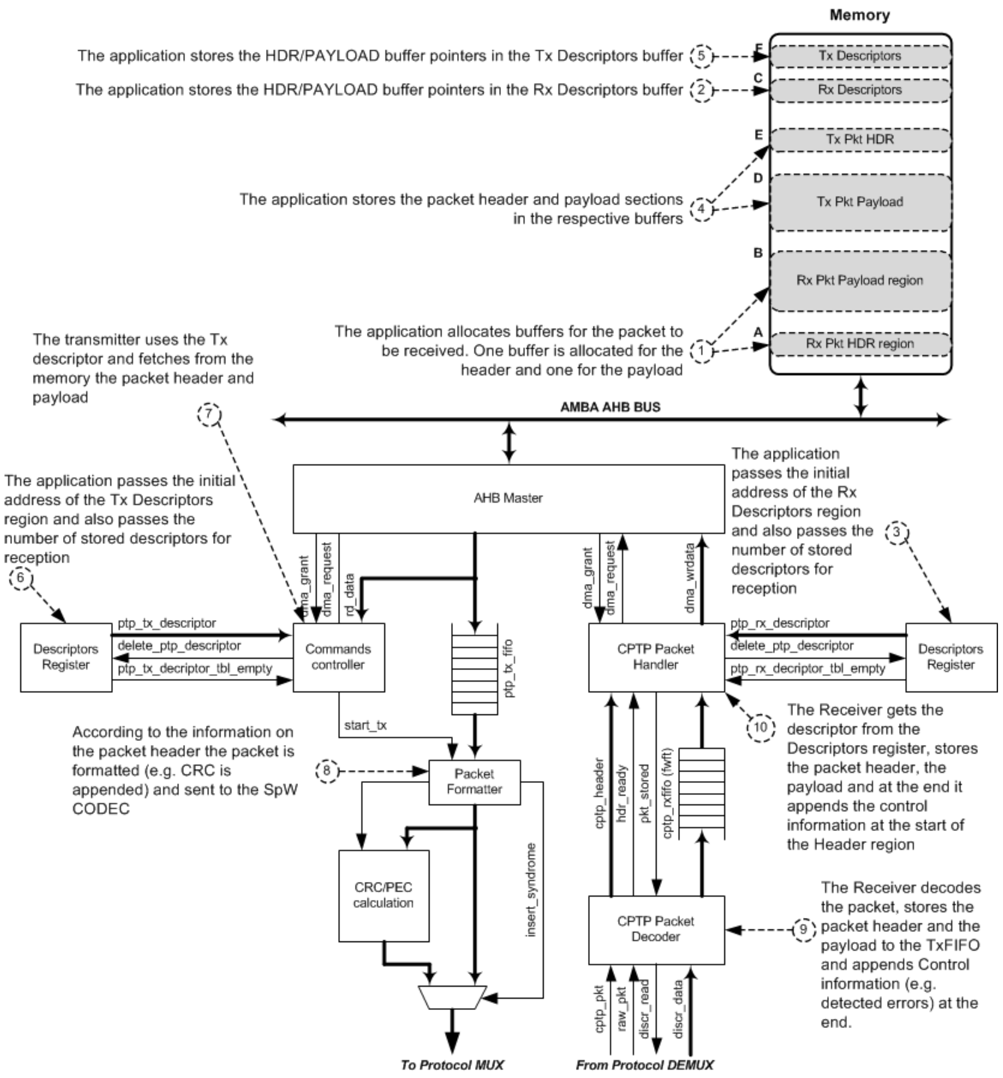
Top Level Design (3/3)

- **IP Core configuration options**
  - VHDL Generics or IP-XACT
  - For overall Core and for each block
- **Overall Core configuration**
  - Target technology
  - Number of SpW ports
  - NDCP/CPTP/TDP support, Switch NDCP support
- **RMAP configuration**
  - Initiator/Target support
  - Initiator/Target DMA length
  - Verify Buffer size
  - Target supported commands
- **NDCP configuration**
  - Vendor/Device IDs, Versions
  - RAM/ROM based address translation
  - Address translation table depth
- **CPTP configuration**
  - Tx/Rx descriptors width/depth
  - Maximum packet DMA size
  - Discard Rx packet on memory buffer unavailable


AMBA AHB 2.0
AHB/APB Bridge
AHB Master with DMA
Switch PnP Regbank
Distributed Interrupts Regbank (Switch)
End Node PnP Regbank
Distributed Interrupts Regbank (End Node)
Time Distribution Protocol
RMAP/ NDCP/ SpW-D
CPTP/ Raw SpW
RMAP/ NDCP/ SpW-D
Switch APB Peripherals
End Node APB Peripherals
End Node Protocols
Time-Codes/ Interrupts Handler (End Node)
Protocol MUX/DEMUX
Signaling Codes interface
Time-Codes/ Interrupts Handler (Switch)
Port 1
Port 0
SpW Switch Core
Port 2
Port N
Signaling Codes interface
Switch/CODEC(s) configuration
SpW CODEC
•••
SpW CODEC
SpW Switch part
SpW NODE INTERFACE IP CORE
Block present only when the switch is instantiated

CPTP Block (1/5)

- **Supports**
  - Raw SpW i.e. w.o. known PID
  - CCSDS Packet Transfer Protocol (CPTP**)**
- **Multiple packets Rx/Tx without processor intervention**
- **All structures stored in local memory (DMA descriptors, packets header/payload)**
- **Header and payload stored at different memory locations**
- **Supports**
  - disabled PEC/CRC,
  - insertion/verification of PEC/CRC,
  - verification of payload length field
- **Circular buffer logic for DMA descriptors (no update required)**



5 The application stores the HDR/PAYLOAD buffer pointers in the Tx Descriptors buffer

2 The application stores the HDR/PAYLOAD buffer pointers in the Rx Descriptors buffer

4 The application stores the packet header and payload sections in the respective buffers

1 The application allocates buffers for the packet to be received. One buffer is allocated for the header and one for the payload

7 The transmitter uses the Tx descriptor and fetches from the memory the packet header and payload

6 The application passes the initial address of the Tx Descriptors region and also passes the number of stored descriptors for reception

3 The application passes the initial address of the Rx Descriptors region and also passes the number of stored descriptors for reception

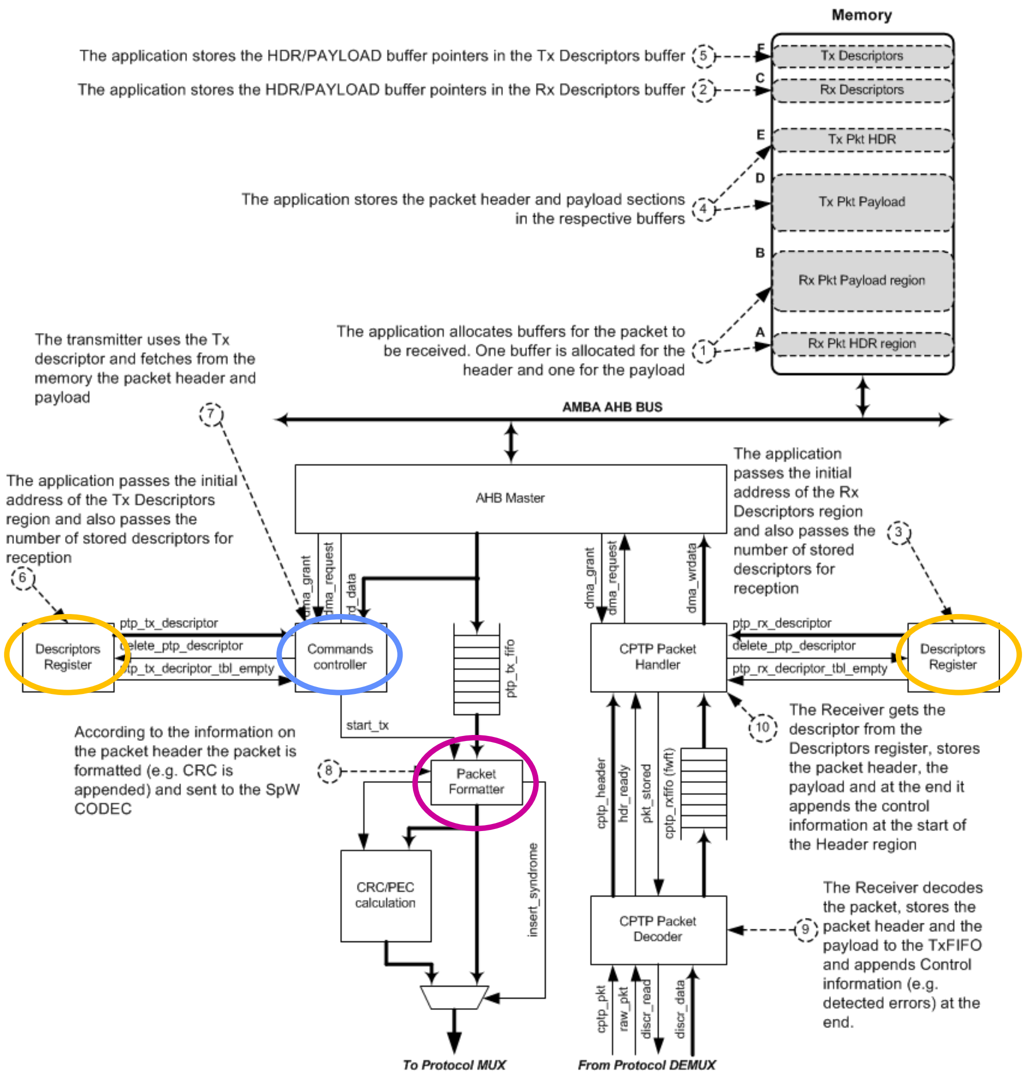
8 According to the information on the packet header the packet is formatted (e.g. CRC is appended) and sent to the SpW CODEC

10 The Receiver gets the descriptor from the Descriptors register, stores the packet header, the payload and at the end it appends the control information at the start of the Header region

9 The Receiver decodes the packet, stores the packet header and the payload to the TxFIFO and appends Control information (e.g. detected errors) at the end.

CPTP Block (2/5)

- **Descriptor Register block**
  - Contains the header/payload address in the memory
  - Contains N sets of Tx/Rx pointers
  - Circular architecture (no need to update descriptors)
- **Commands Controller block**
  - Requests header/payload pointers from Descriptors Register
  - Fetches the packet header and payload from memory and stores fetched data in *CPTP_TX_FIFO*
- **Packet Formatter block**
  - Reads header from FIFO and
  - Determines whether Raw/CPTP packet, CRC/PEC will be inserted, EoP/EEP termination
  - Asserts IRQ signal on completion



The application stores the HDR/PAYLOAD buffer pointers in the Tx Descriptors buffer

The application stores the HDR/PAYLOAD buffer pointers in the Rx Descriptors buffer

The application stores the packet header and payload sections in the respective buffers

The transmitter uses the Tx descriptor and fetches from the memory the packet header and payload

The application allocates buffers for the packet to be received. One buffer is allocated for the header and one for the payload

The application passes the initial address of the Tx Descriptors region and also passes the number of stored descriptors for reception

The application passes the initial address of the Rx Descriptors region and also passes the number of stored descriptors for reception

According to the information on the packet header the packet is formatted (e.g. CRC is appended) and sent to the SpW CODEC

The Receiver gets the descriptor from the Descriptors register, stores the packet header, the payload and at the end it appends the control information at the start of the Header region

The Receiver decodes the packet, stores the packet header and the payload to the TxFIFO and appends Control information (e.g. detected errors) at the end.

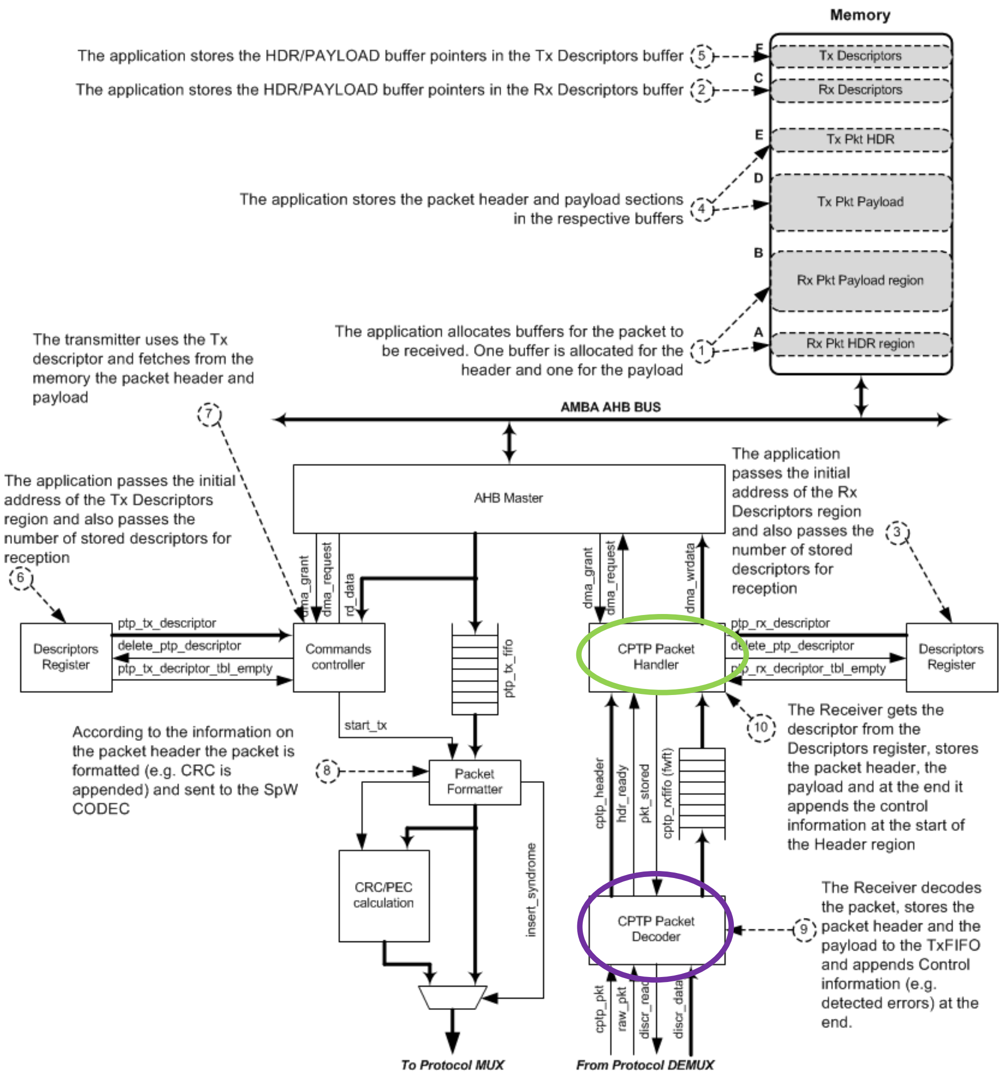
CPTP Block (3/5)

Packet Decoder

- Stores the **packet header** until completion of reception
- **Verifies PEC/CRC**, packet length and updates header structure
- **Truncates packet** based on **programmed max. length**
- **Discards** incoming packet **if Rx pointer does not exist** (programmable function)
- **Buffers** payload and header to be stored in system memory

Packet Handler

- **Retrieves header/payload pointers** from Descriptor register
- **Stores payload** to the system memory
- **Stores header** (with error info appended) to the system memory
- Informs the Packet Decoder on completion of operations
- **Asserts IRQ signal** on completion



(5) The application stores the HDR/PAYLOAD buffer pointers in the Tx Descriptors buffer

(2) The application stores the HDR/PAYLOAD buffer pointers in the Rx Descriptors buffer

(4) The application stores the packet header and payload sections in the respective buffers

(1) The application allocates buffers for the packet to be received. One buffer is allocated for the header and one for the payload

(7) The transmitter uses the Tx descriptor and fetches from the memory the packet header and payload

(6) The application passes the initial address of the Tx Descriptors region and also passes the number of stored descriptors for reception

(3) The application passes the initial address of the Rx Descriptors region and also passes the number of stored descriptors for reception

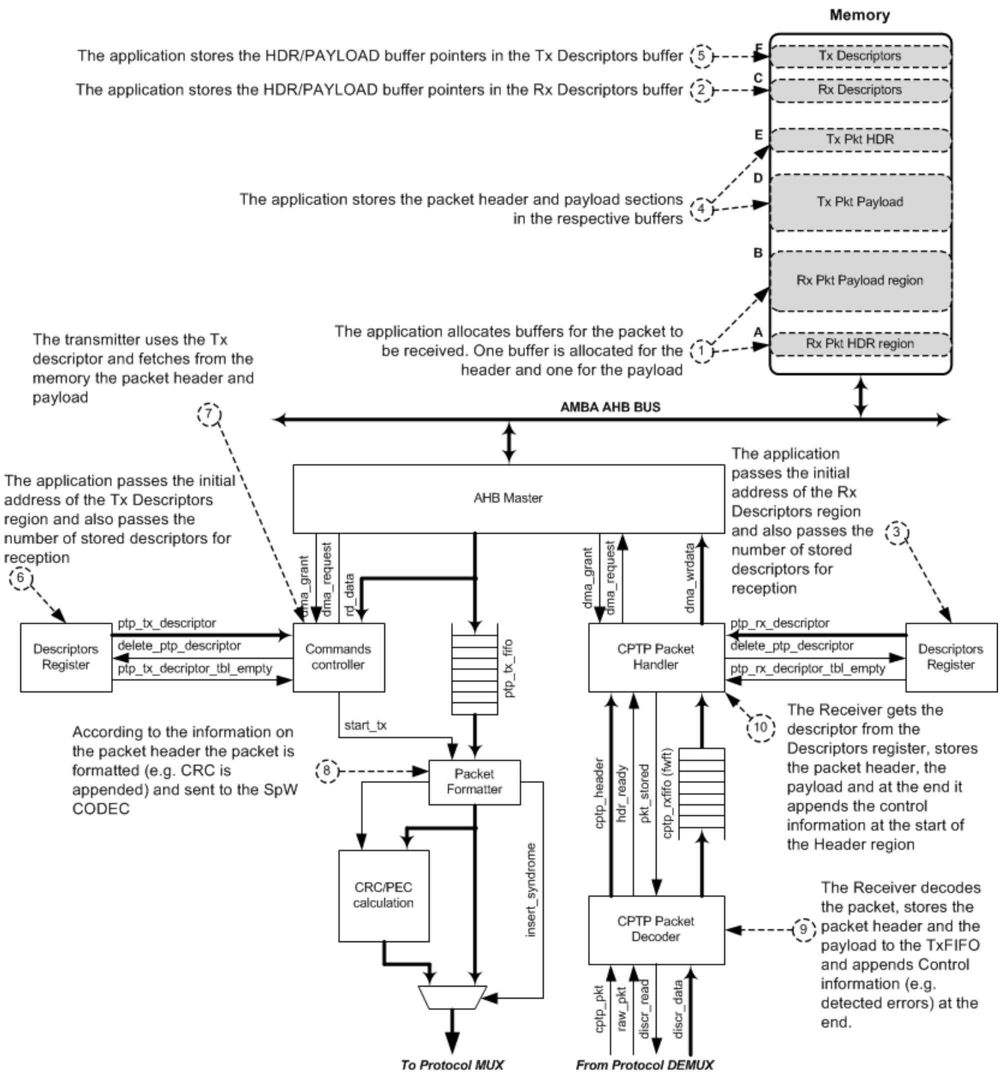
(8) According to the information on the packet header the packet is formatted (e.g. CRC is appended) and sent to the SpW CODEC

(10) The Receiver gets the descriptor from the Descriptors register, stores the packet header, the payload and at the end it appends the control information at the start of the Header region

(9) The Receiver decodes the packet, stores the packet header and the payload to the TxFIFO and appends Control information (e.g. detected errors) at the end.

CPTP Block (4/5)

- **Operation - Reception/Transmission**
  - User allocates header and payload areas in the memory
  - User stores a set of pointers in the Rx/Tx Descriptors memory area
  - User writes pointer to the Rx/Tx pointers area in the Descriptors register
  - User writes in Descriptors area how many pointers are available
- **Subsequent Reception/Transmission**
  - Descriptors area already contains pointers to allocated regions – No need to update them
  - Descriptors area already contains pointers to the Rx/Tx Descriptors area
  - Simply write the number of pointers in the Descriptors register
  - Descriptors register has circular logic so it will use the previously programmed pointers



The application stores the HDR/PAYLOAD buffer pointers in the Tx Descriptors buffer (5)

The application stores the HDR/PAYLOAD buffer pointers in the Rx Descriptors buffer (2)

The application stores the packet header and payload sections in the respective buffers (4)

The application allocates buffers for the packet to be received. One buffer is allocated for the header and one for the payload (1)

The transmitter uses the Tx descriptor and fetches from the memory the packet header and payload (7)

The application passes the initial address of the Tx Descriptors region and also passes the number of stored descriptors for reception (6)

The application passes the initial address of the Rx Descriptors region and also passes the number of stored descriptors for reception (3)

According to the information on the packet header the packet is formatted (e.g. CRC is appended) and sent to the SpW CODEC (8)

(10) The Receiver gets the descriptor from the Descriptors register, stores the packet header, the payload and at the end it appends the control information at the start of the Header region

(9) The Receiver decodes the packet, stores the packet header and the payload to the TxFIFO and appends Control information (e.g. detected errors) at the end.

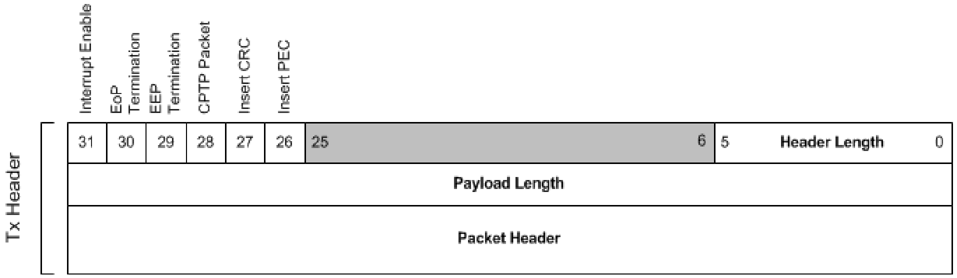
CPTP Block (5/5)

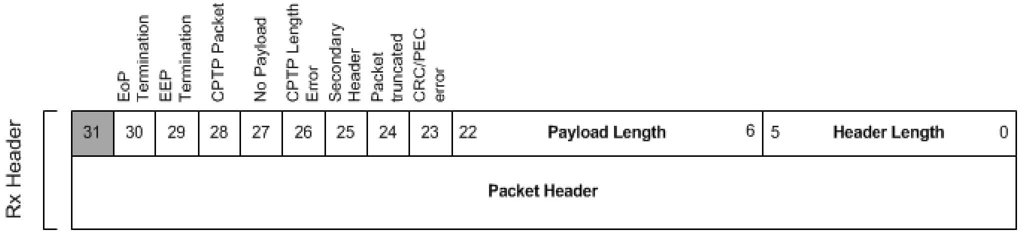
Tx Header – control bits

- **Interrupt Enable:** Commands CPTP transmitted to assert IRQ upon packet transmission
- **EoP/EEP termination:** Determines whether the packet will be EoP/EEP or not terminated
- **CPTP:** Packet determines whether the packet shall be transmitted as CPTP or raw
- **Insert CRC/PEC:** Determines whether CRC, PEC or nothing will be inserted (applicable for CPTP only)
- **Header length:** Number of NCHARs that constitute the packet header (not calculated in CRC/PEC)

Rx Header – status bits

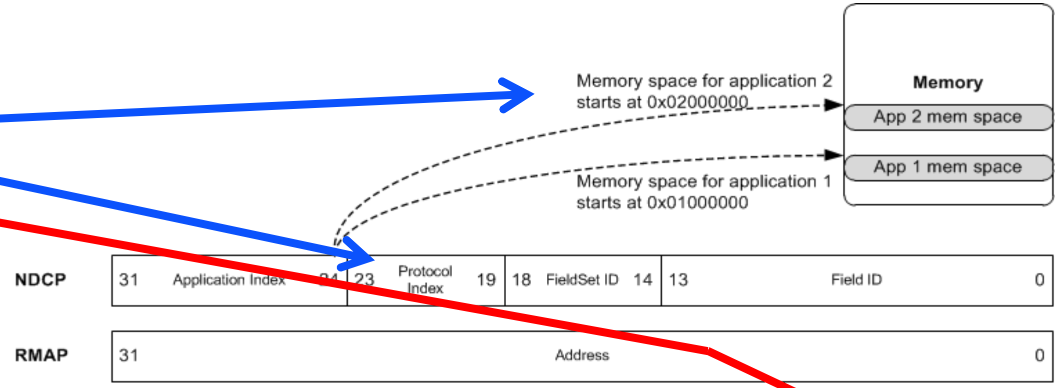
- **EoP/EEP termination:** Packet terminated by EoP EEP
- **CPTP packet:** Indicates whether the packets if CPTP/Raw
- **No Payload:** Indicates that the packet does not have payload
- **CPTP Length Error:** Indicated mismatch between packet data field and actual packet length
- **Secondary Header:** Indicates that CPTP has secondary header
- **Packet truncated:** Indicates that the packet is truncated
- **CRC/PEC error:** Indicated that the packet had CRC or PEC error
- **Header Length:** Number of NCHARs that constitute the packet header (not calculated in CRC/PEC)





RMAP Block – NDCP Extensions

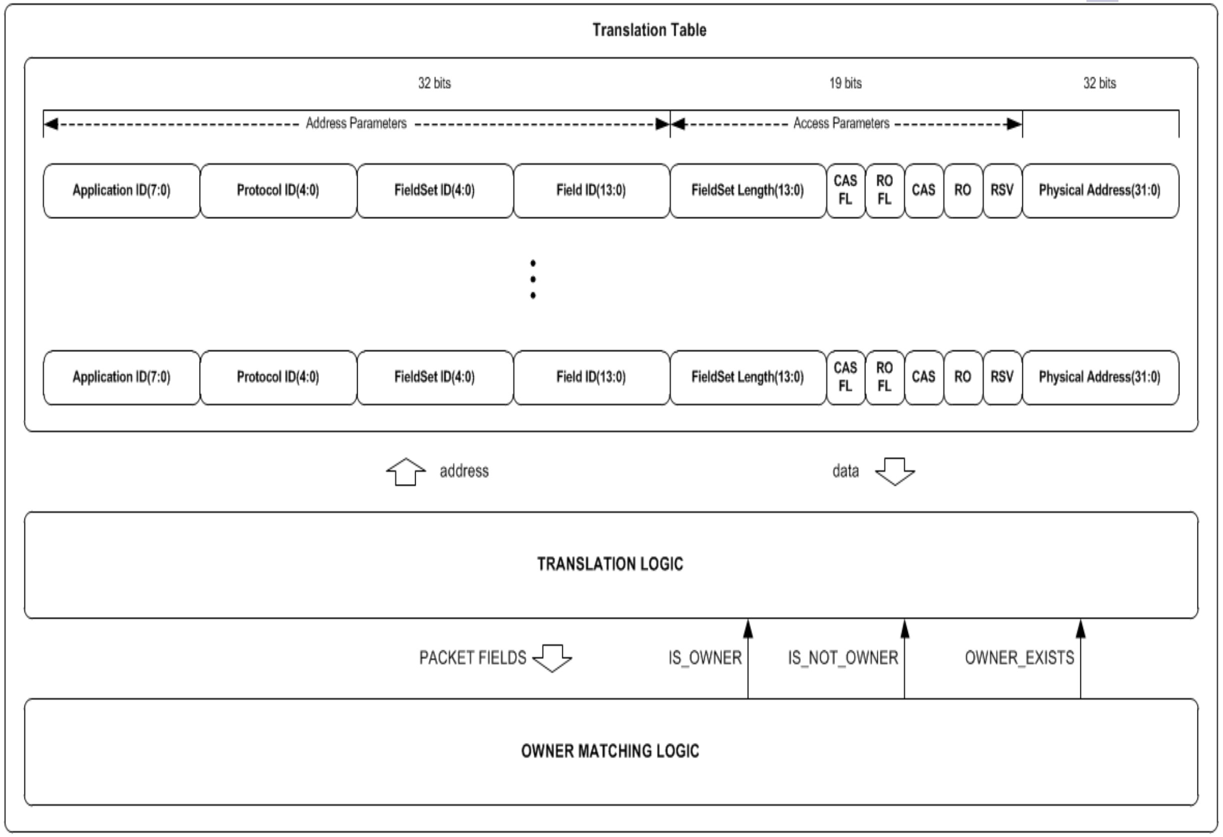
- **NDCP is based on RMAP with the following differences:**
  - Address Field structure: Application, Protocol, FieldSet, Field
  - Results in Highly fragmented memory space
  - Field ID increases by 1 although each field is 32 bits wide
  - 32-bit implementation is assumed
  - CAS functionality is not left to implementers (as RMW) but it is standardized
  - Only the device “owner” can perform write to the configuration space
- ⇨ **If NDCP is instantiated, the data bus width shall always be 32-bits wide**
- ⇨ **Address Translation block is required in order to:**
  - ⇨ **Result in a more flat memory space**
  - ⇨ **Move memory space to a fixed and constrained address space**
- ⇨ **Authorization Logic shall be extended to handle ownership**



Category	Field Set ID	Field ID	Used address space	Unused address space
Device Information **Application Index 0** **Protocol Index 0**	0: Device Identification	0 - 10: Used 11 - 16383: UNUSED	0x00000000 - 0x0000000A	0x0000000B - 0x00003FFF
	1: Vendor/Product Strings	0 - 16383: Used	0x00004000 - 0x00007FFF	-
	2: Protocol Support	0 - 31: Used 32 - 16383: UNUSED	0x00008000 - 0x0000801F	0x00008020 - 0x0000BFFF
	3: Application Support	0 - 511: Used 512 - 16383: UNUSED	0x0000C000 - 0x0000C1FF	0x0000C200 - 0x0000FFFF
	4 - 31: UNUSED	-	-	0x00010000 - 0x0007FFFF
SpW Protocol **Application Index 0** **Protocol Index 1** ⋮	0: Device Configuration	0 - 7: Used 8 - 16383: UNUSED	0x00080000 - 0x00080007	0x00080008 - 0x00083FFF
	1: Link Configuration	0 - 7: UNUSED	-	0x00084000 - 0x00084007
		8 - 255: Used 256 - 16383: UNUSED	0x00084008 - 0x000840FF	0x00084100 - 0x00087FFF
	2: Routing Table	0 - 511: Used 512 - 16383: UNUSED	0x00088000 - 0x000881FF	0x00088200 - 0x0008BFFF

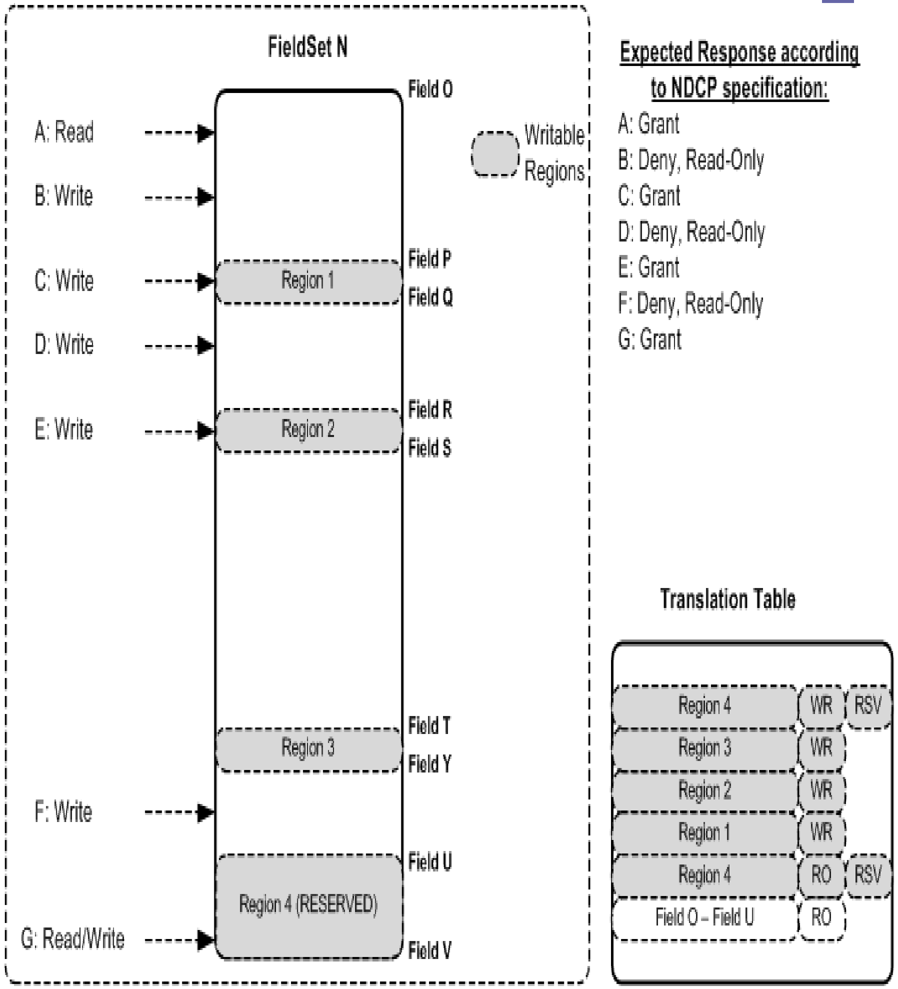
RMAP Block – NDCP Extensions, Address Translation

- **The EndNode and Switch retain each one, an Address Translation Block**
  - Address Translation Table: Contains the associations between NDCP address and physical address
  - Translation Logic: Engine which receives NDCP address, searches in the Table and return the physical address
  - Ownership Logic: Assesses whether the received packet was sent by the peripheral owner or not
- **Address Parameters**
  - Application Index
  - Protocol Index
  - FieldSet ID
  - Field ID number of the lowest Field ID for this entry
- **Access Parameters**
  - FieldSet length: Length of the FieldSet or the FieldSet subset
  - RO/CAS: Indicates if this region is Read-only/CAS modifiable
  - RSV: Indicates if this region is Reserved
  - CAS FL/RO FL: Indicates if this region is followed by a CAS modifiable/Read-only region (described in the next slide)
- **Physical Address**
  - Address of the FieldSet/Field within the Node’s memory map
  - Returned to RMAP Target State Machine to perform the access



RMAP Block – NDCP Extensions, NDCP v 1.7 peculiarities (1/2)

- **Table is scanned sequentially and when the first matching entry is found grant/deny is returned**
- **(5.3.2.2.d, i, k) Write operation to a RO/CAS field and CAS operation to a RO field are not accepted**
  - If the first entry in the table corresponds to the RO Field O – field U then write to Region 1 will return Deny although write is allowed in Region 1
  - ⇨ Read Only regions shall be inserted after Writable and CAS modifiable regions in the table.
  - Assuming that a Write operation starts in a region and spans more than its upper boundary then this shall not be allowed if the region that follows is RO/CAS
  - ⇨ CAS/RO Follows flags in the table used to identify such cases and reject commands that cross such regions
- **(5.3.2.2 a) It shall be possible to read any field in a FieldSet which has one or more defined fields**
  - If only Regions 1 to 4 are defined in the table then an read access to region A, D, F will reject the command
  - ⇨ The entire FieldSet is inserted as RO at the end of the table



Expected Response according to NDCP specification:

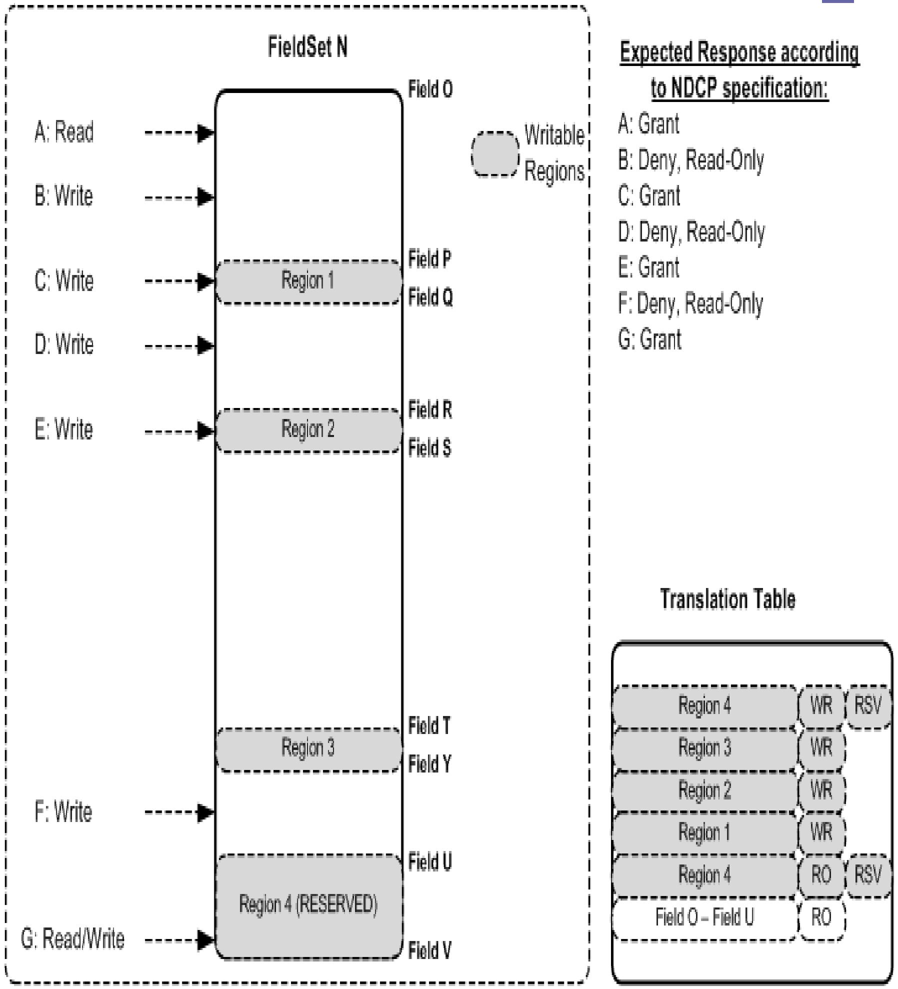
A: Grant
B: Deny, Read-Only
C: Grant
D: Deny, Read-Only
E: Grant
F: Deny, Read-Only
G: Grant

Translation Table

Region 4	WR	RSV
Region 3	WR	
Region 2	WR	
Region 1	WR	
Region 4	RO	RSV
Field O – Field U	RO	

RMAP Block – NDCP Extensions, NDCP v 1.7 peculiarities (2/2)

- **(5.3.2.2.b) It shall be possible to read any field identified as reserved**
- **(5.3.2.2.c) The contents of any field identified as reserved shall be zero**
- **(5.3.2.2.f) Any information written to a reserved field which is not identified as read only shall be discarded**
  - The State Machine shall know if a field is reserved in order to return zeros on read commands
  - The State Machine shall know if a field is reserved in order not to perform any AHB access on write commands and just return a NDCP Reply
  - ⇨ Reserved regions shall be explicitly programmed in the table
  - ⇨ In the example shown here, Region 4 is programmed as RSV&Writable at the beginning of the table and also as RSV&RO at the end of the table in order to handle above cases.



Expected Response according to NDCP specification:

A: Grant
B: Deny, Read-Only
C: Grant
D: Deny, Read-Only
E: Grant
F: Deny, Read-Only
G: Grant

Agenda

IP Design

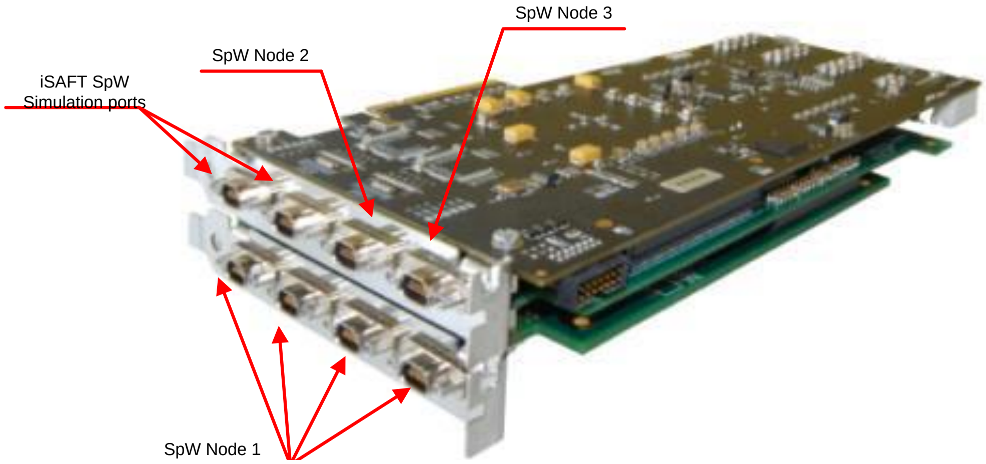
- Top Level
- CPTP/Raw SpW packets block
- RMAP
- RMAP – NDCP Extensions
- SpW Switch

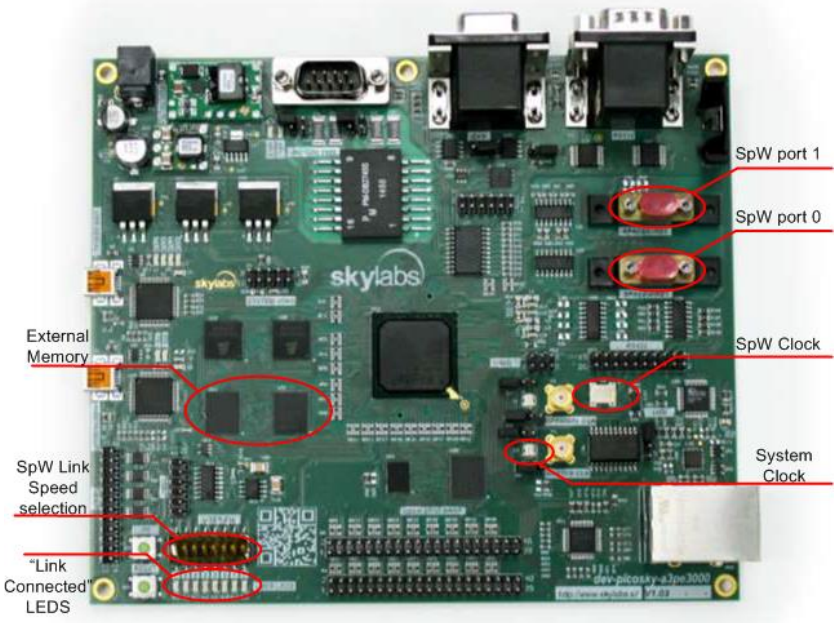
Implementation, Metrics and Validation results

- Target Technologies
- Synthesis Results
- Functional tests
- Performance tests

Xilinx & MicroSemi Targets

- **Synthesis on Xilinx V5/V6, MicroSemi ProASIC3/RTAX**
- **Demonstrations on Virtex 6, ProASIC3**
- **All instances stimulated by iSAFT Simulator**
- **Virtex 6 implementation based on TELETEL Octal SpaceWire G2 board**
- **Three Node IP Instances**
  - 1 x EndNode with SpW Switch and four external ports. All protocol engines instantiated
  - 1 x EndNode only, all protocols instantiated, ROM based NDCP Address Translation Table
  - 1 x EndNode only, all protocols instantiated, RAM based NDCP Address Translation Table
- **ProASIC3 implementation based on Skylab’s PicoSky board**
- **One Node IP Instance**
  - 1 x EndNode with SpW Switch and two external ports. All protocol engines instantiated
  - ROM based translation tables





RMAP with NDCP Results

RMAP with NDCP support, Verify buffer size 4K, Max DMA length 128 Bytes, Virtex 5QV FX130			
	LUTs	**Registers**	**BRAMs**
RMAP Block	2060	1775	3
RMAP Initiator	900	831	1
RMAP Target	804	647	1
RMAP Packet Validator	332	242	1
Statistics Block	22	55	-

RMAP with NDCP support, Verify buffer size 4K, Max DMA length 128 Bytes, ProASIC3			
	Total Cells	**Registers**	**RAM Blocks**
RMAP Block	8564	1851	21
RMAP Initiator	3842	703	11
RMAP Target	3614	738	2
RMAP Packet Validator	1415	351	8
Statistics Block	127	60	-

RMAP with NDCP Results

RMAP with NDCP support, Verify buffer size 4K, Max DMA length 128 Bytes, Spartan6 LX150

	LUTs	Registers	BRAMs
RMAP Block	2286	1896	4
RMAP Initiator	1036	845	1
RMAP Target	832	725	1
RMAP Packet Validator	386	266	2
Statistics Block	32	60	-

NDCP Results

NDCP Address Translation, Virtex 5QV FX130 (End Node, RAM implementation)

Address Translation Table depth	LUTs	Registers	BRAMs
64	389	353	-
128	372	274	3

NDCP Address Translation, Virtex 5QV FX130 (SpW Switch, ROM implementation)

Number of SpW Ports	LUTs	Registers	BRAMs
4	373	221	5
8	373	221	5
16	373	221	5
32	373	221	5

NDCP Address Translation, ProASIC3 (End Node, RAM implementation)

Address Translation Table depth	Total Cells	Registers	RAM Blocks
64	1390	277	6
128	1390	284	6

NDCP Address Translation, ProASIC3 (SpW Switch, ROM implementation)

Number of SpW Ports	Total Cells	Registers	RAM Blocks
4	1506	257	-
8	1506	257	-
16	1506	257	-
32	1506	257	-

NDCP Results

NDCP Address Translation, Spartan 6 LX150 (End Node, RAM implementation)			
Address Translation Table depth	**LUTs**	**Registers**	**BRAMs**
64	375	353	-
128	381	274	3

NDCP Address Translation, Spartan 6 LX150 (SpW Switch, ROM implementation)			
Number of SpW Ports	**LUTs**	**Registers**	**BRAMs**
4	358	221	3
8	358	221	3
16	358	221	3
32	358	221	3

CPTP Results

CPTP, Max packet size 64K, Max Tx/Rx descriptors 16, Max DMA length 128 Bytes, Virtex 5QV FX130			
	LUTs	**Registers**	**BRAMs**
CPTP Block	1149	853	2
Descriptors Register	96	72	-
Command Controller	164	170	-
Packet Formatter	230	120	-
Packet Decoder	269	166	-
Packet handler	173	125	-
CPTP CRC	16	16	-
CPTP PEC	74	32	-
CPTP Rx FIFO	88	102	1
CPTP Tx FIFO	25	26	1

CPTP, Max packet size 64K, Max Tx/Rx descriptors 16, Max DMA length 128 Bytes, ProASIC3			
	Total Cells	**Registers**	**RAM Blocks**
CPTP Block	4206	869	4
Descriptors Register	396	76	-
Command Controller	857	184	-
Packet Formatter	746	152	-
Packet Decoder	793	136	-
Packet handler	820	135	-
CPTP CRC	86	18	-
CPTP PEC	218	34	-
CPTP Rx FIFO	337	104	2
CPTP Tx FIFO	140	28	2

CPTP Results

CPTP, Max packet size 64K, Max Tx/Rx descriptors 16, Max DMA length 128 Bytes, Spartan 6 LX150

	LUTs	Registers	BRAMs
CPTP Block	1251	881	2
Descriptors Register	98	76	-
Command Controller	243	181	-
Packet Formatter	220	120	-
Packet Decoder	294	166	-
Packet handler	181	132	-
CPTP CRC	24	17	-
CPTP PEC	91	33	-
CPTP Rx FIFO	88	102	1
CPTP Tx FIFO	28	26	1

Full SpW Node Implementation

- All protocol engines instantiated
- RMAP Initiator & Target
- RMAP Verify Buffer Size = 4 K
- Full RMAP implementation (all commands, incrementing-address only)
- Max CPTP packet size = 65 K
- Max CPTP Tx/Rx descriptors = 16
- DMA sizes (RMAP initiator/reply handler, CPTP Rx/Tx) = 128 Bytes
- NDCP translation table depth = 64

Overall Core, Virtex 5QV FX130 target

	LUTs	Registers	BRAMs
Without Switch	8405 (9%)	7475 (8%)	10
With 2 external ports	14906 (16%)	12315 (13%)	19
With 6 external ports	18830 (21%)	15409 (17%)	19
With 14 external ports	27407 (30%)	21961 (24%)	19
With 30 external ports	50329 (54%)	37448 (41%)	19

Overall Core, ProASIC3 target (With/Without RMAP initiator, no interrupts block)

	Total Cells	Registers	RAM Blocks
Without Switch	35807 (48%)/31472 (42%)	7225/6348	28/17
With 2 external ports	49483 (66%)/47337 (72%)	11220/10493	47/36
With 6 external ports	64978 (86%)/60596 (81%)	14205/13314	59/48

Full SpW Node Implementation

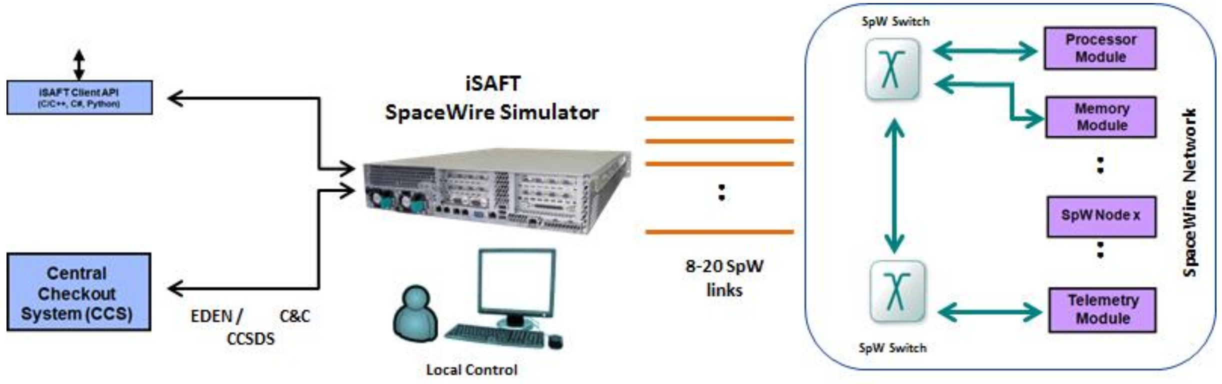
- All protocol engines instantiated
- RMAP Initiator & Target
- RMAP Verify Buffer Size = 4 K
- Full RMAP implementation (all commands, incrementing-address only)
- Max CPTP packet size = 65 K
- Max CPTP Tx/Rx descriptors = 16
- DMA sizes (RMAP initiator/reply handler, CPTP Rx/Tx) = 128 Bytes
- NDCP translation table depth = 64

Overall Core, Spartan6 LX150 target

	LUTs	Registers	BRAMs
Without Switch	7985 (9%)	7051 (4%)	6
With 2 external ports	16128 (18%)	13030 (7%)	14
With 6 external ports	20464 (22%)	16319 (9%)	14
With 14 external ports	29962 (32%)	23255 (13%)	14
With 30 external ports	56344 (61%)	38361 (21%)	14

Validation – iSAFT Simulator & Recorder

- Advanced EGSE platform with traffic generation capabilities that simulates SpaceWire devices or instruments, enabling S/C integration tests before the availability of Flight Models.
- Provides an 8 – 20 port SpaceWire interface with advanced traffic generation and asynchronous transmission capabilities (RMAP/CPTP libraries, (time) triggered transmission, IRIG time-stamping, …).
- Suitable for the verification of new protocols and protocol variations.



Data Bus Front End

Instrument Simulation

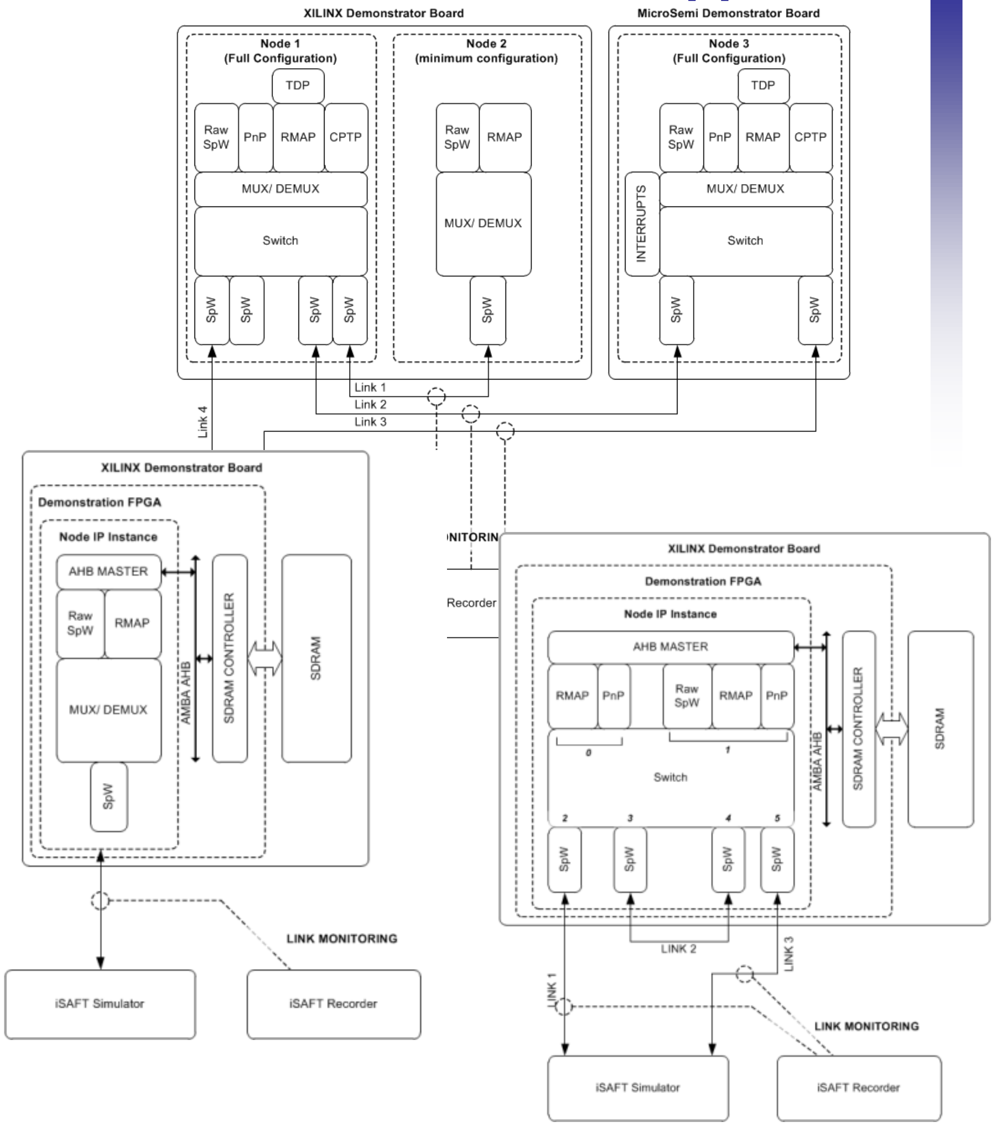
OBC Simulation (PLM EGSE)

Suitable for various EGSEs

AIT/AIV, FMEA, CE

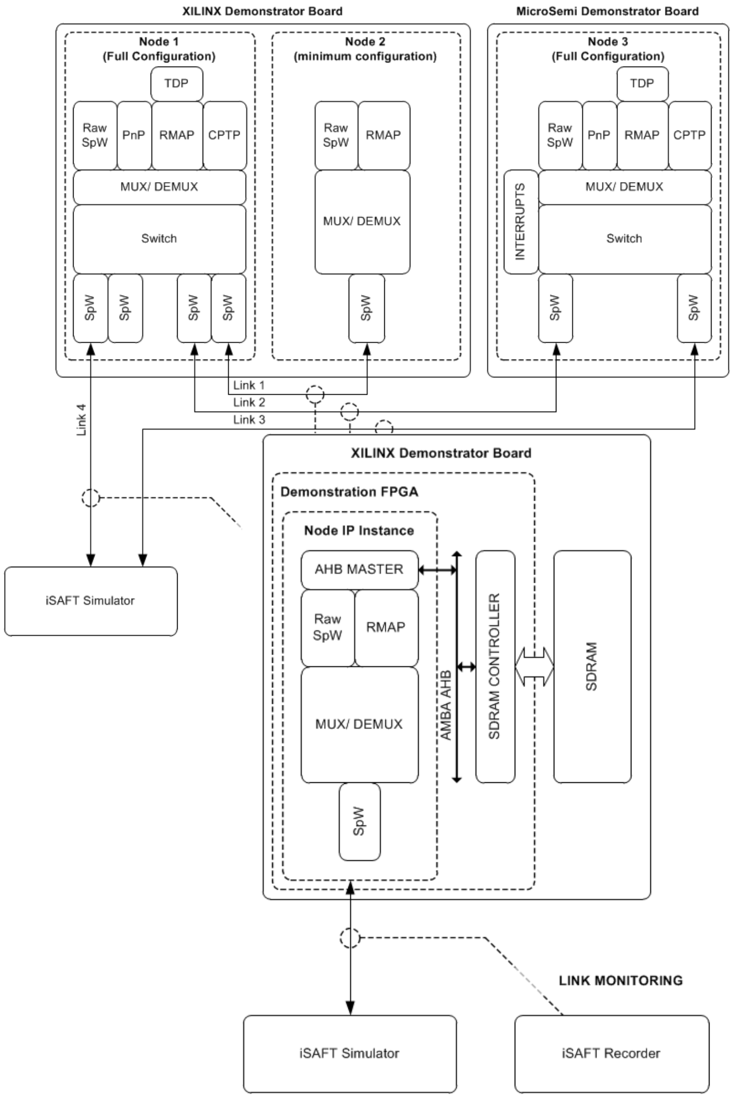
Validation Approach

- Incremental Validation Approach
- Five different test set-ups
- Validation of EndNode implementations followed by full implementations including SpW Switch
- Configuration and readback through RMAP
- Configuration through RMAP, readback through NDCP
- Configuration through NDCP, readback through RMAP
- Stimulation of all core functions (CPTP transmission, registers programming …) externally through RMAP
- Use of COTS tools (iSAFT Recorder & Simulator)
- Validation/assessment of core’s performances
- Validation through low level SW
- Demonstration on iSAFT TestRunner (18 Test Cases implemented)
- All tests passed successfully



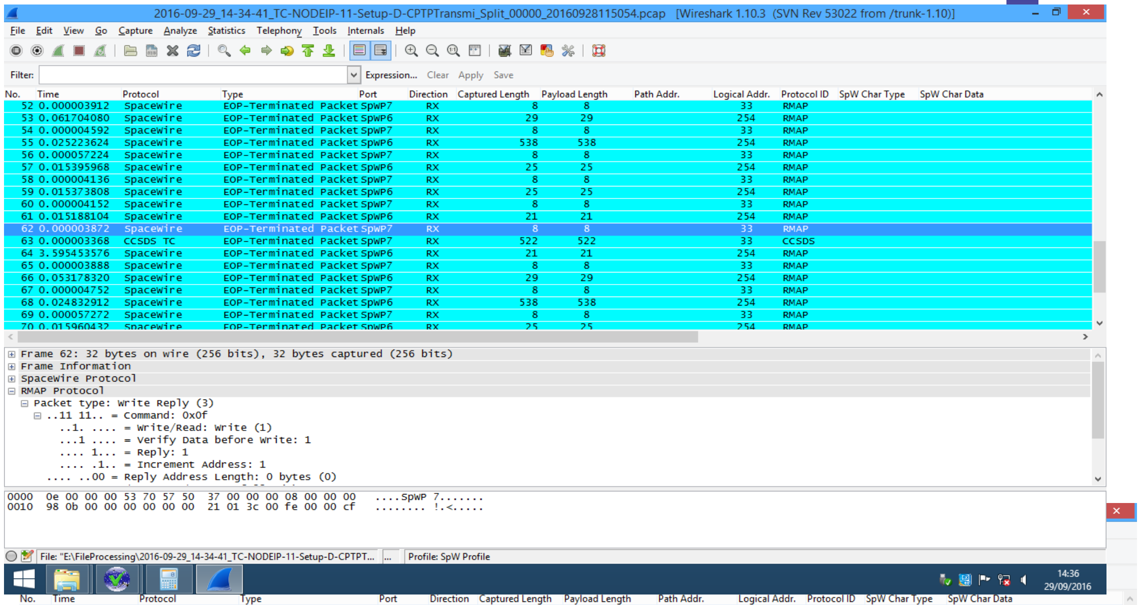
Validation Scenarios(1/2)

- **RMAP functionality tests**
  - Validation of RMAP functionality is performed against TELETEL’s RMAP Conformance Test Suite
  - Validation initially performed on a Node without a switch instantiated to allow for RMAP block validation without potential problems of the SpW switch
- **NDCP functionality tests**
  - The tests include a mix of NDCP and RMAP commands
  - RMAP Reads validate values returned through NDCP read
  - Validation of the NDCP Address Translation block which converts the “highly-segmented” NDCP memory space to a contiguous memory space
- **After RMAP/NDCP validation at End Node, tests were repeated on nodes with instantiated SpW Switch**



Validation Scenarios(2/2)

- **CPTP/Raw SpW Transmitter tests**
  - RMAP commands used to configure the CPTP/Raw SpW Transmitter
  - iSAFT Simulator checks all the fields of the CPTP/Raw SpW packets transmitted by the Node
  - The tests are repeated for different payload lengths and configurations (EOP/EEP, CRC/PEC, CPTP Secondary header)
- **CPTP/Raw SpW Receiver tests**
  - RMAP commands used to configure the CPTP/Raw SpW Receiver
  - iSAFT Simulator transmits CPTP/Raw SpW packets to the Node and validates with RMAP commands if the packet is received correctly by the Node
  - The tests are repeated for different payload lengths and configurations (EOP/EEP, packet truncation, CRC/PEC with or without error, CPTP Secondary header)

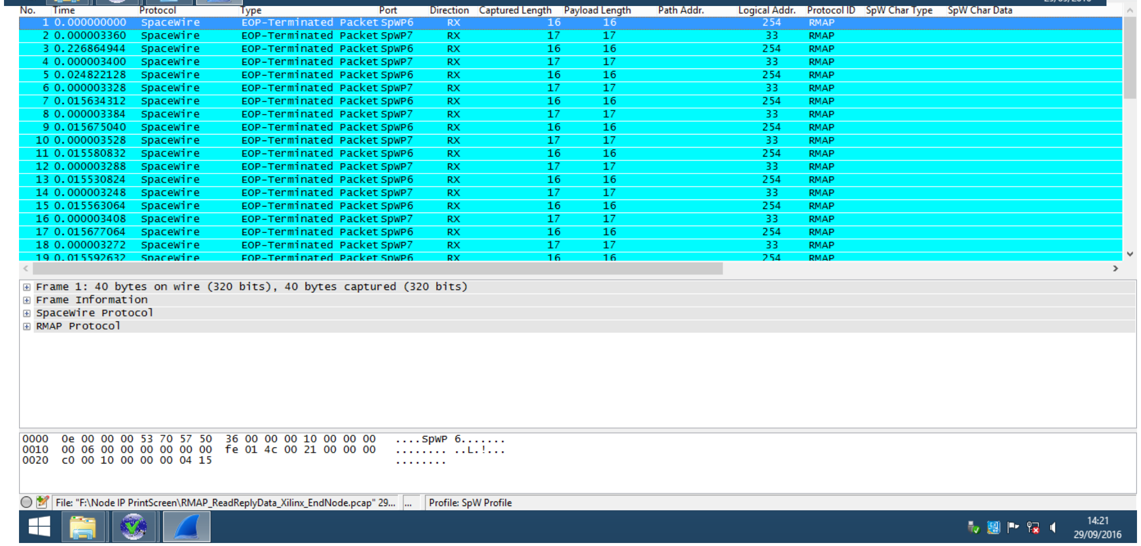

2016-09-29_14-34-41_TC-NODEIP-11-Setup-D-CPTPTransmi_Split_00000_20160928115054.pcap [Wireshark 1.10.3 (SVN Rev 53022 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

No.	Time	Protocol	Type	Port	Direction	Captured Length	Payload Length	Path Addr.	Logical Addr.	Protocol ID	SpW Char Type	SpW Char Data
52	0.000003912	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
53	0.061704080	SpaceWire	EOP-Terminated Packet	SpWP6	RX	29	29		254	RMAP		
54	0.000004592	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
55	0.025223624	SpaceWire	EOP-Terminated Packet	SpWP6	RX	538	538		254	RMAP		
56	0.000057224	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
57	0.015395968	SpaceWire	EOP-Terminated Packet	SpWP6	RX	25	25		254	RMAP		
58	0.000004136	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
59	0.015373808	SpaceWire	EOP-Terminated Packet	SpWP6	RX	25	25		254	RMAP		
60	0.000004152	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
61	0.015188104	SpaceWire	EOP-Terminated Packet	SpWP6	RX	21	21		254	RMAP		
62	0.000003872	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
63	0.000003368	CCSDS TC	EOP-Terminated Packet	SpWP7	RX	522	522		33	CCSDS		
64	3.595453576	SpaceWire	EOP-Terminated Packet	SpWP6	RX	21	21		254	RMAP		
65	0.000003888	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
66	0.053178320	SpaceWire	EOP-Terminated Packet	SpWP6	RX	29	29		254	RMAP		
67	0.000004752	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
68	0.024832912	SpaceWire	EOP-Terminated Packet	SpWP6	RX	538	538		254	RMAP		
69	0.000057272	SpaceWire	EOP-Terminated Packet	SpWP7	RX	8	8		33	RMAP		
70	0.015960432	SpaceWire	EOP-Terminated Packet	SpWP6	RX	25	25		254	RMAP		

```
⊞ Frame 62: 32 bytes on wire (256 bits), 32 bytes captured (256 bits)
⊞ Frame Information
⊞ SpaceWire Protocol
⊟ RMAP Protocol
  ⊟ Packet type: Write Reply (3)
    ⊟ ..11 11.. = Command: 0x0f
        ..1. .... = Write/Read: Write (1)
        ...1 .... = Verify Data before Write: 1
        .... 1... = Reply: 1
        .... .1.. = Increment Address: 1
      .... ..00 = Reply Address Length: 0 bytes (0)
```

```
0000  0e 00 00 00 53 70 57 50  37 00 00 00 08 00 00 00   ....SpWP 7.......
0010  98 0b 00 00 00 00 00 00  21 01 3c 00 fe 00 00 cf   ........ !.<.....
```

File: "E:\FileProcessing\2016-09-29_14-34-41_TC-NODEIP-11-Setup-D-CPTPT... ... Profile: SpW Profile



No.	Time	Protocol	Type	Port	Direction	Captured Length	Payload Length	Path Addr.	Logical Addr.	Protocol ID	SpW Char Type	SpW Char Data
1	0.000000000	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
2	0.000003360	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
3	0.226864944	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
4	0.000003400	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
5	0.024822128	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
6	0.000003328	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
7	0.015634312	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
8	0.000003384	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
9	0.015675040	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
10	0.000003528	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
11	0.015580832	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
12	0.000003288	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
13	0.015530824	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
14	0.000003248	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
15	0.015563064	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
16	0.000003408	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
17	0.015677064	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		
18	0.000003272	SpaceWire	EOP-Terminated Packet	SpWP7	RX	17	17		33	RMAP		
19	0.015592632	SpaceWire	EOP-Terminated Packet	SpWP6	RX	16	16		254	RMAP		

```
⊞ Frame 1: 40 bytes on wire (320 bits), 40 bytes captured (320 bits)
⊞ Frame Information
⊞ SpaceWire Protocol
⊞ RMAP Protocol
```

```
0000  0e 00 00 00 53 70 57 50  36 00 00 00 10 00 00 00   ....SpWP 6.......
0010  00 06 00 00 00 00 00 00  fe 01 4c 00 21 00 00 00   ........ ..L.!...
0020  c0 00 10 00 00 00 04 15                            ........
```

File: "F:\Node IP PrintScreen\RMAP_ReadReplyData_Xilinx_EndNode.pcap" 29... ... Profile: SpW Profile

Latency Measurements

- From reception of a RMAP command to the time the response was transmitted by the IP Core
- From the stimulation of the RMAP Initiator to the transmission of the RMAP command (Initiator was stimulated by the iSAFT Simulator through a RMAP write command)
- The reception of a NDCP command, transmitted from the iSAFT Simulator, to the time the response was transmitted by the UUT (NDCP Latency)
- The stimulation (through a RMAP command) of the CPTP Transmitter, to the time the CPTP/Raw SpW Packet was transmitted by the UUT (CPTP latency)

Measured Latencies on Virtex 6 (125 MHz core clock, 100 Mbps link speed)

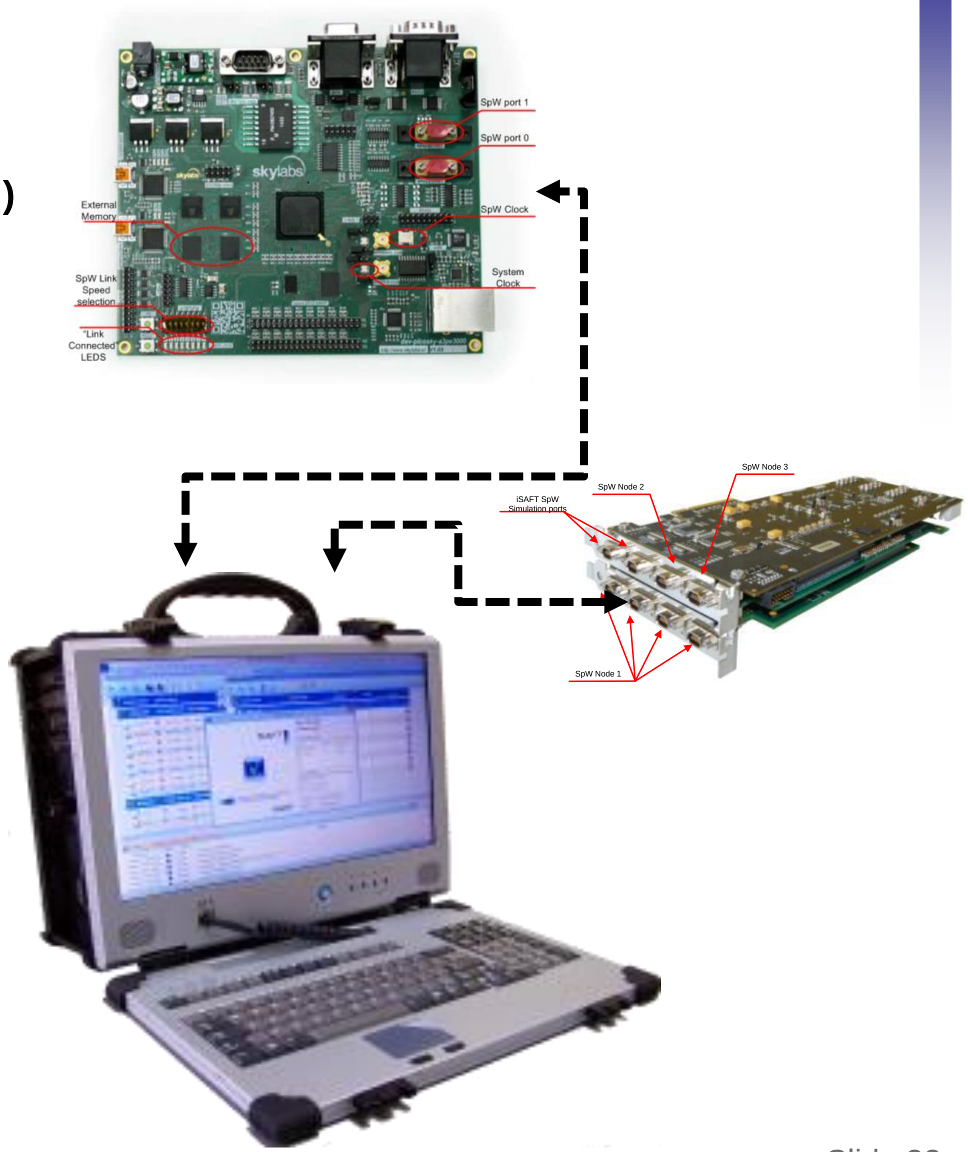
	With Switch (us)	**Without Switch (us)**
RMAP Target	1.58 us (SDRAM read), 1.24 us (reg. Write, RMW)	1.30 us (SDRAM read), 1.0 us (reg. Write, RMW)
RMAP Initiator	2 us (incl. SDRAM access)	1.76 us (incl. SDRAM access)
NDCP target	1.35 us (Read/Write) 1.4 us (CAS)	1.10 us (Read/Write), 1.15 us (CAS)
CPTP	3.25 us (incl. SDRAM access)	3 us (incl. SDRAM access)

Measured Latencies on ProASIC3 (20 MHz core clock, 100 Mbps link speed)

	With Switch (us)	**Without Switch (us)**
RMAP Target	6.4 us (SRAM Read/Write)	4.96 us (SRAM Read/Write)
NDCP target	6.6 us (reg. Read), 7.3 us (reg. Write, CAS)	5.1 us (reg. Read), 5.8 us (reg. Write, CAS)
CPTP	12.4 us (incl. SRAM access)	10.9 us (incl. SRAM access)

Robustness tests

- The core has been tested during long runs (overnight, 3-days tests)
- iSAFT Simulator Traffic Generation capability was exploited (mixed traffic, programmable packet to packet delays – at microsecond level)
- Simple tests included the back-to-back transmission of RMAP/NDCP commands to assess Target robustness
- Tests to stimulate the RMAP Initiator or the CPTP blocks included transmission of packet sequences with programmed delays between them
- Traffic was captured by the iSAFT Recorder
- Tests on both Xilinx and MicroSemi implementations
- No packet loss!



Conclusions

- Most of the code relies on generic VHDL
- Technology dependent primitives are only required for:
  - Asynchronous FIFOs used in the SpW CoDec
  - DDR used in the SpW CoDec
  - Clock multiplexer in the SpW CoDec

⇨ Technology dependent macros required for the SpW CoDec ONLY. Users who already use a SpW CoDec are not required to use any (new) technology dependent macros

- Minor issues faced with MicroSemi tool related to the instantiation of the RMAP verify buffer (the same memory is correctly synthesized by Xilinx and Libero for other blocks in the design)

⇨ Current version explicitly uses MicroSemi TPRAM primitive for the RMAP Verify Buffer

- The SpW Node IP will be available for ESA projects through the ESA IP-core service.