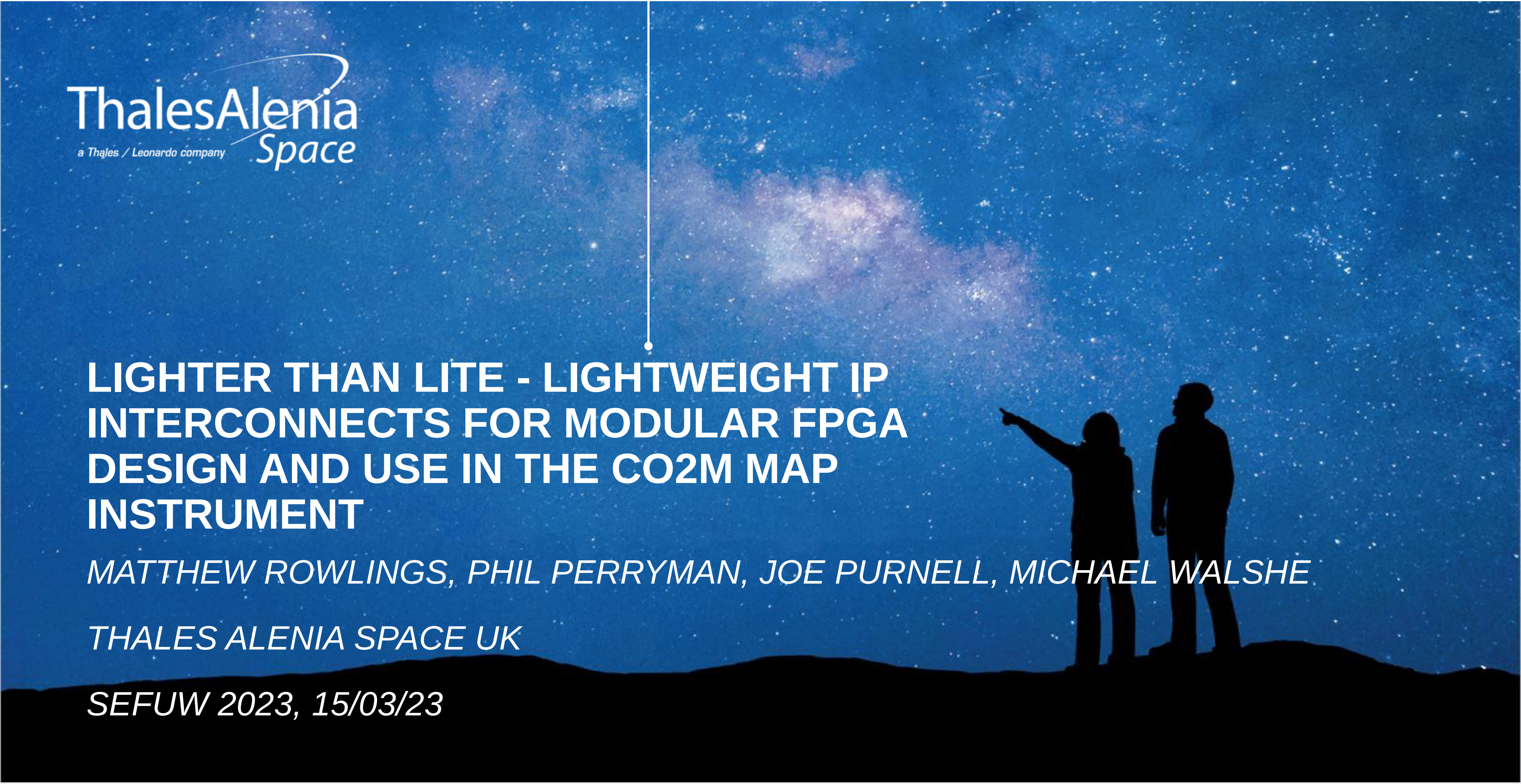


LIGHTER THAN LITE - LIGHTWEIGHT IP INTERCONNECTS FOR MODULAR FPGA DESIGN AND USE IN THE CO2M MAP INSTRUMENT

MATTHEW ROWLINGS, PHIL PERRYMAN, JOE PURNELL, MICHAEL WALSHE

THALES ALENIA SPACE UK

SEFUW 2023, 15/03/23

1. OVERVIEW OF CO2M MAP FPGA ARCHITECTURE

2. IP INTERCONNECT CONCEPT

3. IP INTERCONNECT TOOLBOX

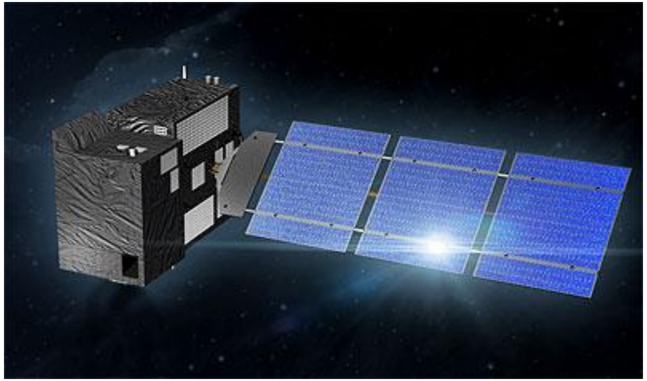
4. SUMMARY

OVERVIEW OF CO2M MAP FPGA ARCHITECTURE

CO2M MAP INSTRUMENT

Multi-Angle Polarimeter (MAP)

- x4 detectors with polarising filter stackup (CIS120 detector, Te2v)
- Image processing and compression to send 4Gbps detector data over a 100Mbps SpaceWire link
- x4 DDR2 interfaces (DDC 97D2H series, 4Gbit per device)
- x4 mechanism controllers for control of calibration filters in front of detectors
- PID thermal control of optical assemblies to keep to 1°C of operating temperature

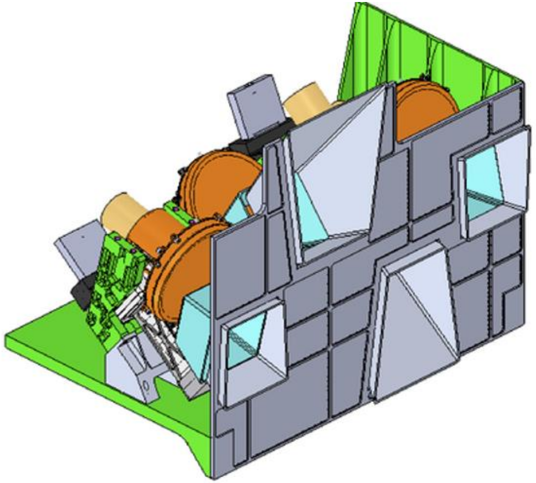


https://space.skyrocket.de/doc_sdat/co2m.htm

As well as the above, the MAP FPGA also needs to provide instrument control and platform interfaces:

- HKTM reporting (thermistors, supplies)
- Power supply control for cameras, motors, heaters
- MIL1553 TMTC interface
- PPS and OBT management
- CCSDS packet generator for Science SpaceWire interface
- EEPROM for system and detector configuration

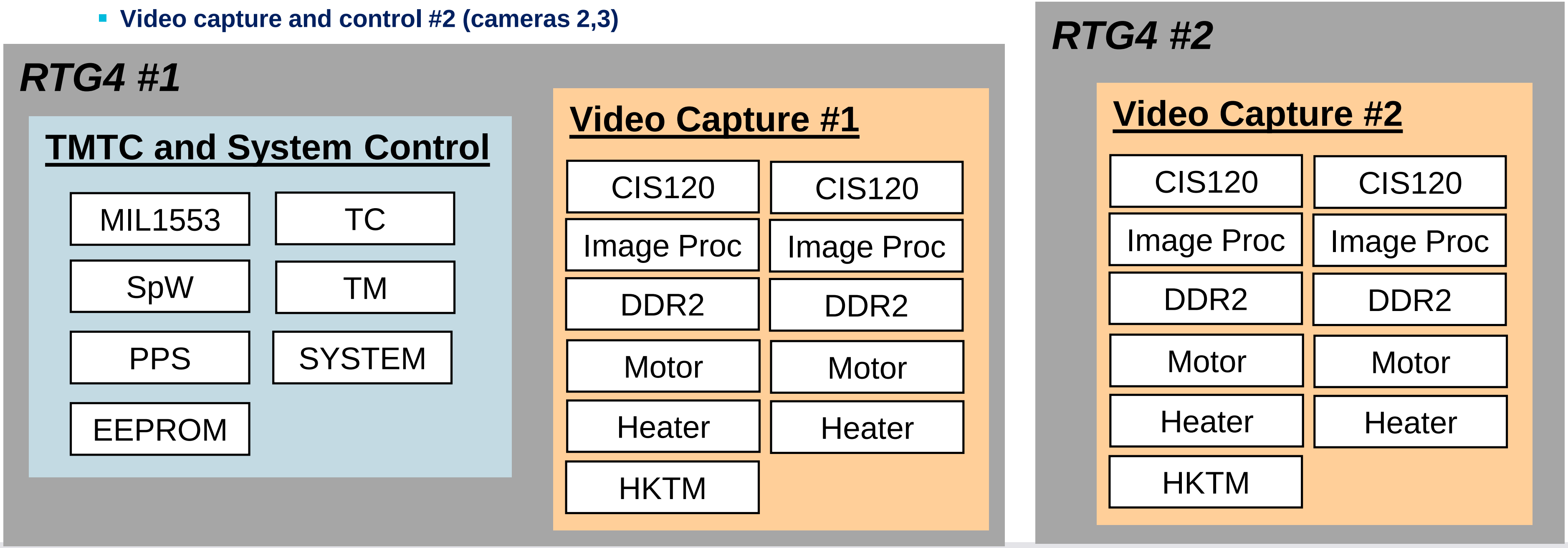
No processor on MAP to support these functions



The Multi Angle Polarimeter (MAP) on board ESA's Copernicus Carbon Dioxide Monitoring mission (CO2M) ICSO 2021

CO2M MAP FPGA ARCHITECTURE #1

- x2 RTG4s required, mostly driven by IO requirements, and performance of the image acquisition and processing
- System partitioned into three segments:
  - TMTC and System control
  - Video capture and control #1 (cameras 0,1)
  - Video capture and control #2 (cameras 2,3)



RTG4 #1

TMTC and System Control

MIL1553
TC
SpW
TM
PPS
SYSTEM
EEPROM

Video Capture #1

CIS120
CIS120
Image Proc
Image Proc
DDR2
DDR2
Motor
Motor
Heater
Heater
HKTM

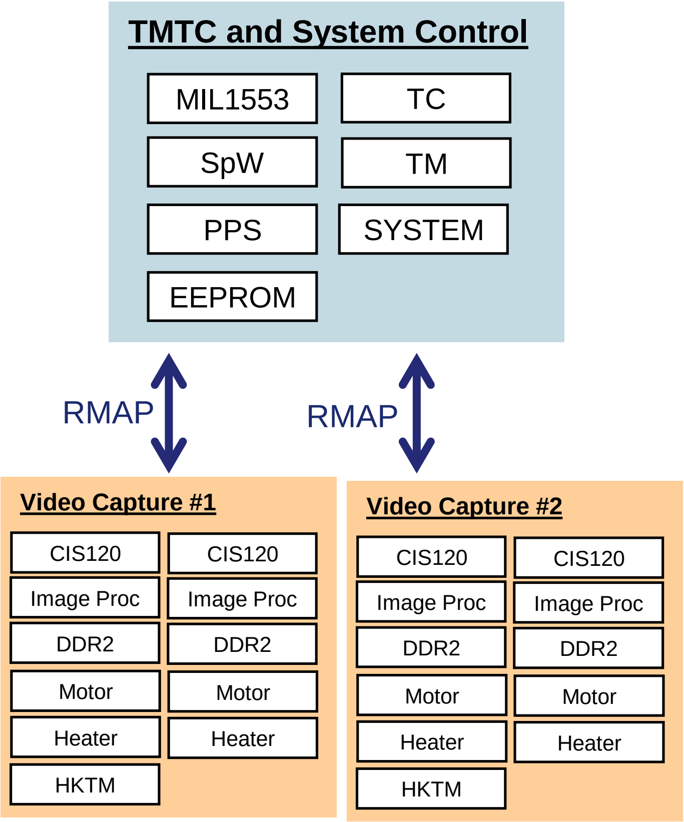
RTG4 #2

Video Capture #2

CIS120
CIS120
Image Proc
Image Proc
DDR2
DDR2
Motor
Motor
Heater
Heater
HKTM

CO2M MAP FPGA ARCHITECTURE #2

- **It is easy to draw these blocks in a Visio diagram**
  - But how do we connect them together?
  - How do we bridge the RTG4s efficiently?
- **Further challenges:**
  - 16MB of camera calibration data to be copied out of EEPROM at boot (into both RTG4s)
  - Thermistor data must be shared between both RTG4s
  - Support for collection of over 250 HKTM parameters
  - Support for over 30 defined TCs
- **The solution: a memory mapped interconnect between all of these blocks**
  - As much control and status for a block as possible is placed in registers
  - Interconnect Masters allow data to be transferred between blocks
  - Allows access to any control/status register in the system from anywhere else in the system
- **By using Remote Memory Access Protocol (RMAP) over SpW to bridge the RTG4s, the memory map for the whole system is encapsulated within one large address space**



TMTC and System Control

MIL1553, TC, SpW, TM, PPS, SYSTEM, EEPROM

RMAP

RMAP

Video Capture #1

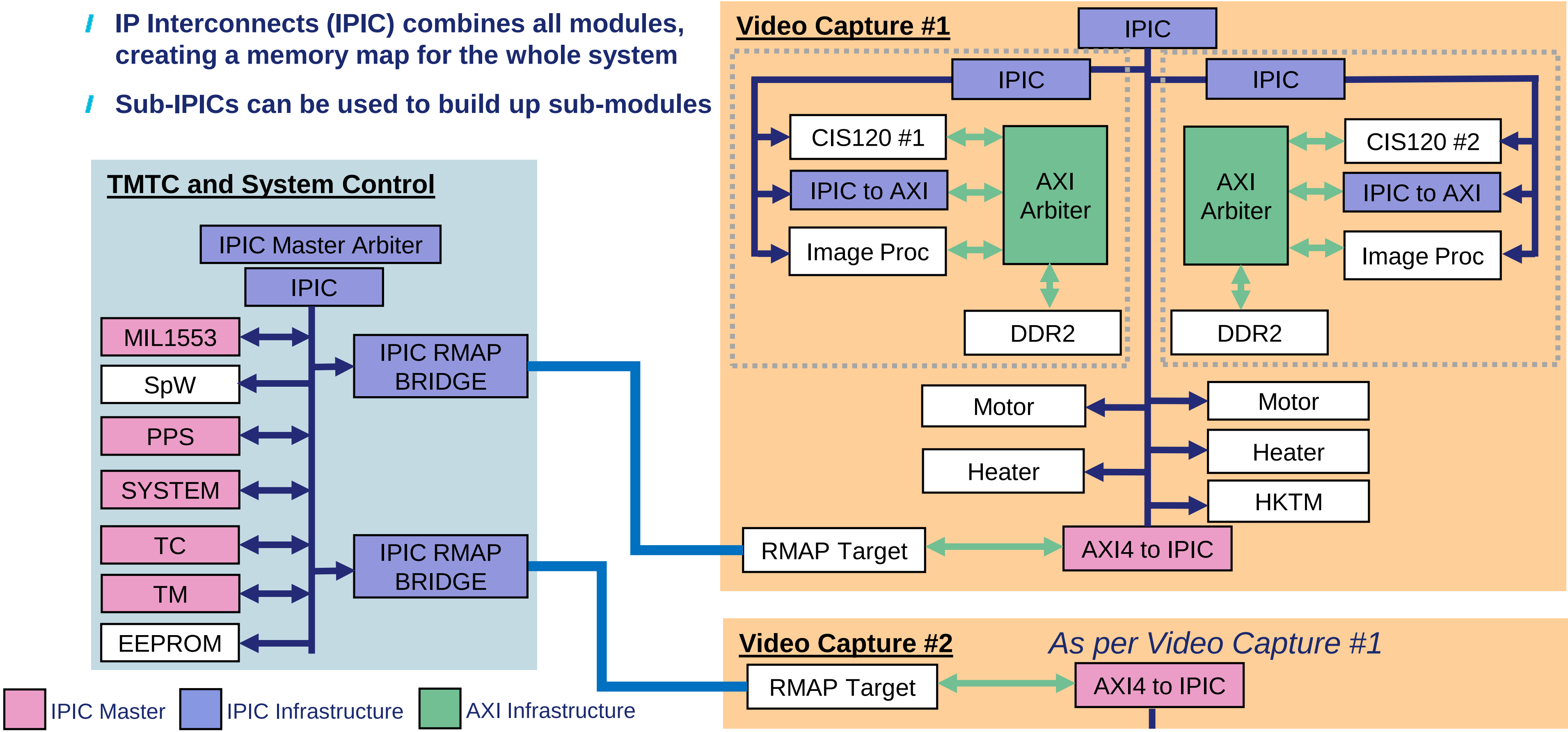
CIS120, CIS120, Image Proc, Image Proc, DDR2, DDR2, Motor, Motor, Heater, Heater, HKTM

Video Capture #2

CIS120, CIS120, Image Proc, Image Proc, DDR2, DDR2, Motor, Motor, Heater, Heater, HKTM

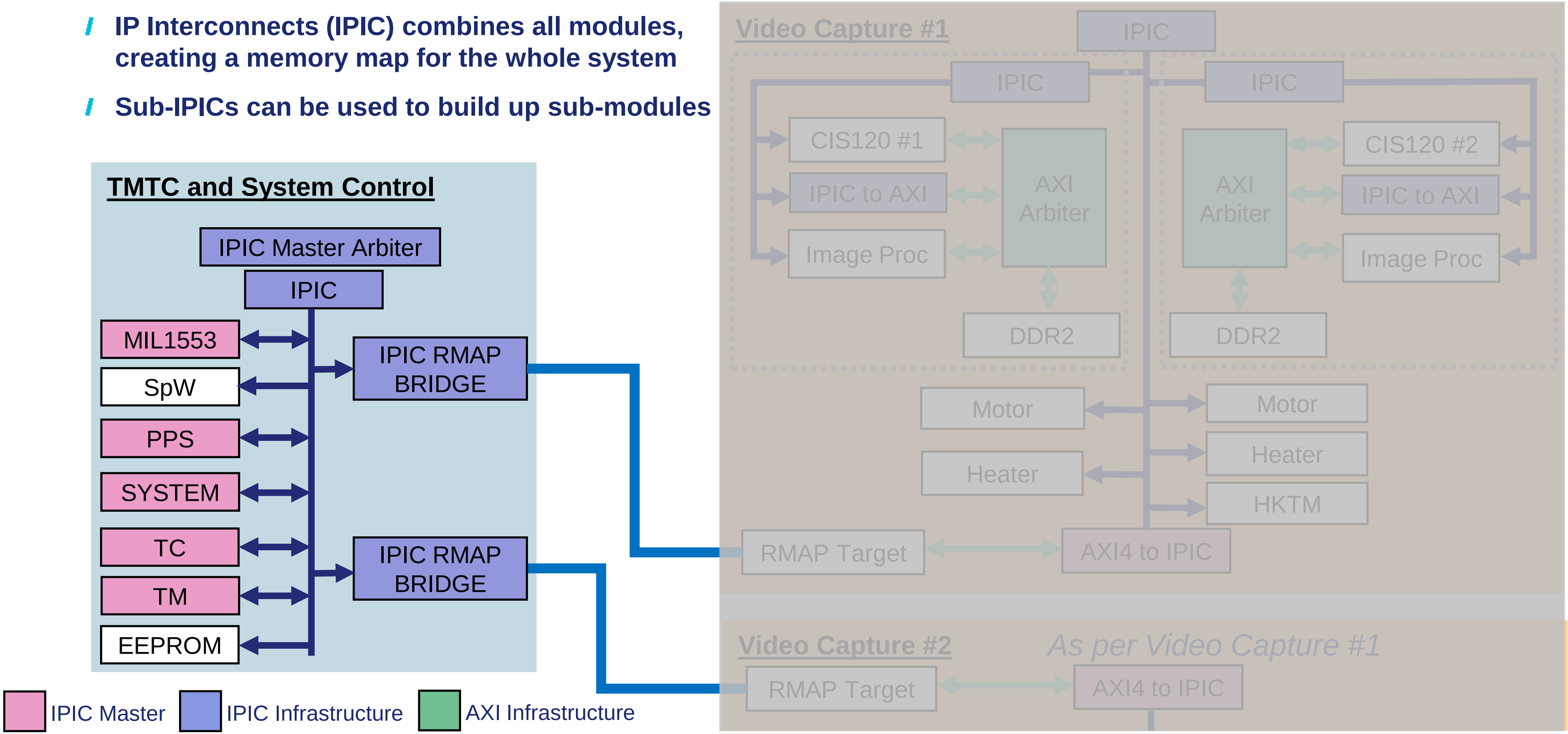
CO2M MAP FPGA ARCHITECTURE #3

- IP Interconnects (IPIC) combines all modules, creating a memory map for the whole system
- Sub-IPICs can be used to build up sub-modules



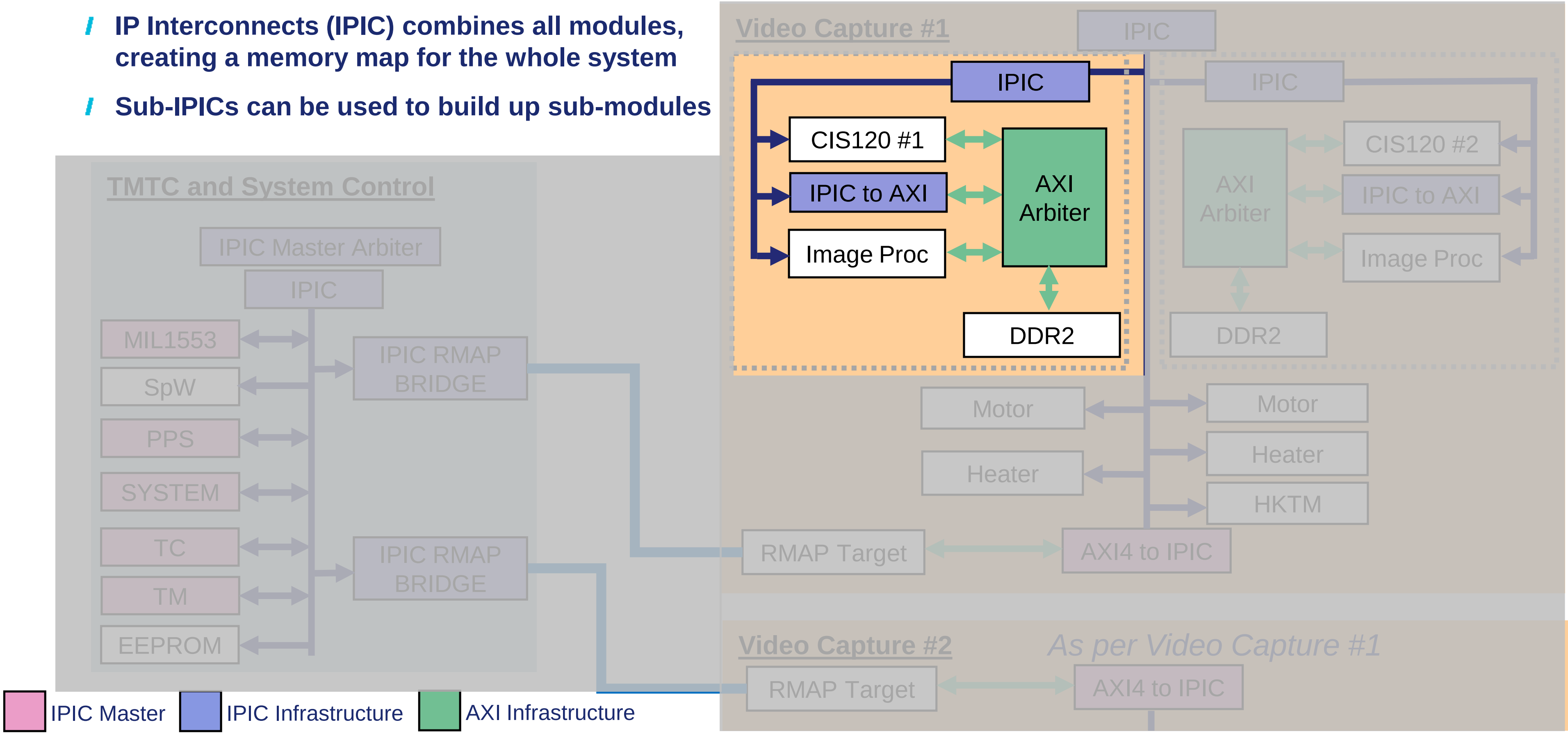
CO2M MAP FPGA ARCHITECTURE #3

- **IP Interconnects (IPIC) combines all modules, creating a memory map for the whole system**
- **Sub-IPICs can be used to build up sub-modules**



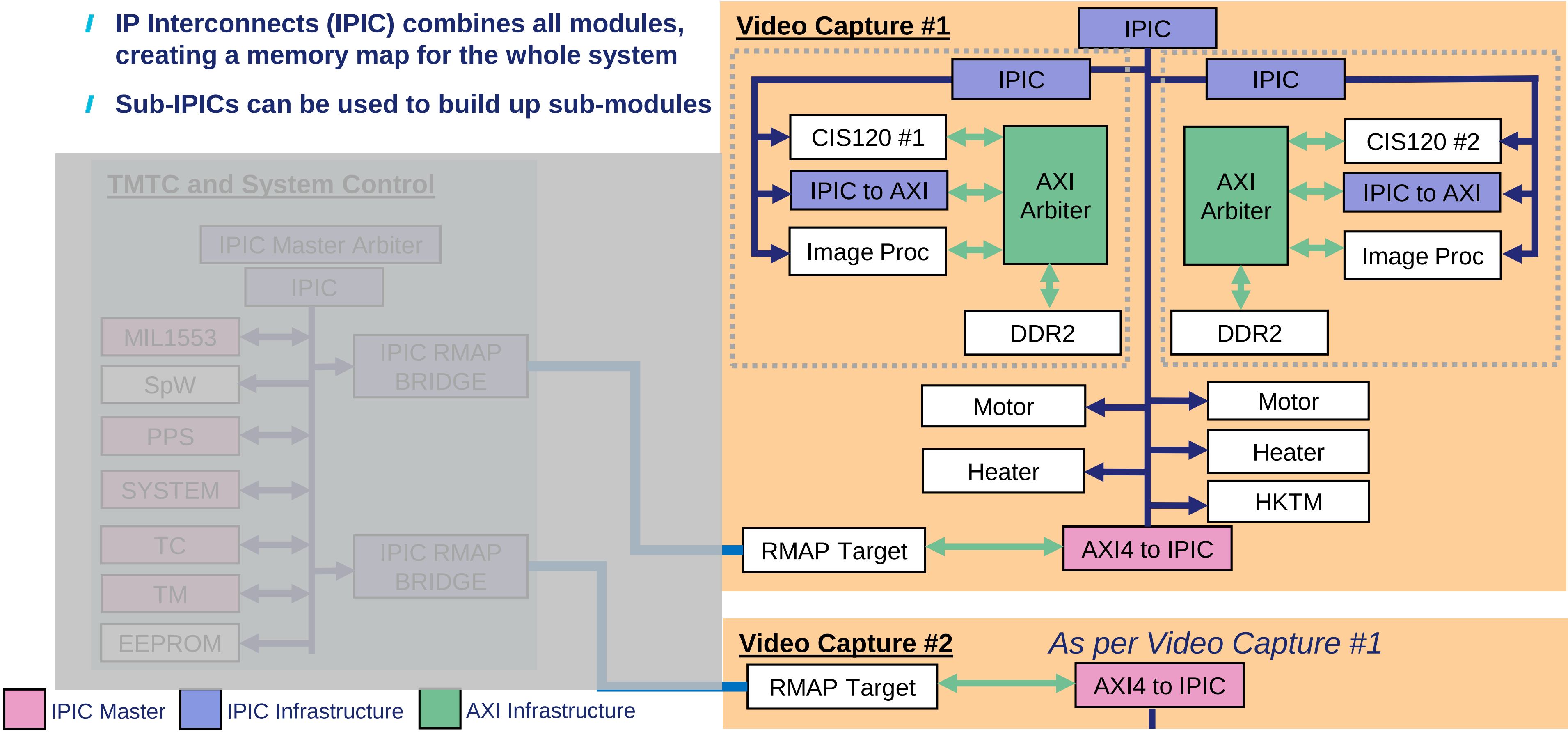
CO2M MAP FPGA ARCHITECTURE #3

- **IP Interconnects (IPIC) combines all modules, creating a memory map for the whole system**
- **Sub-IPICs can be used to build up sub-modules**



CO2M MAP FPGA ARCHITECTURE #3

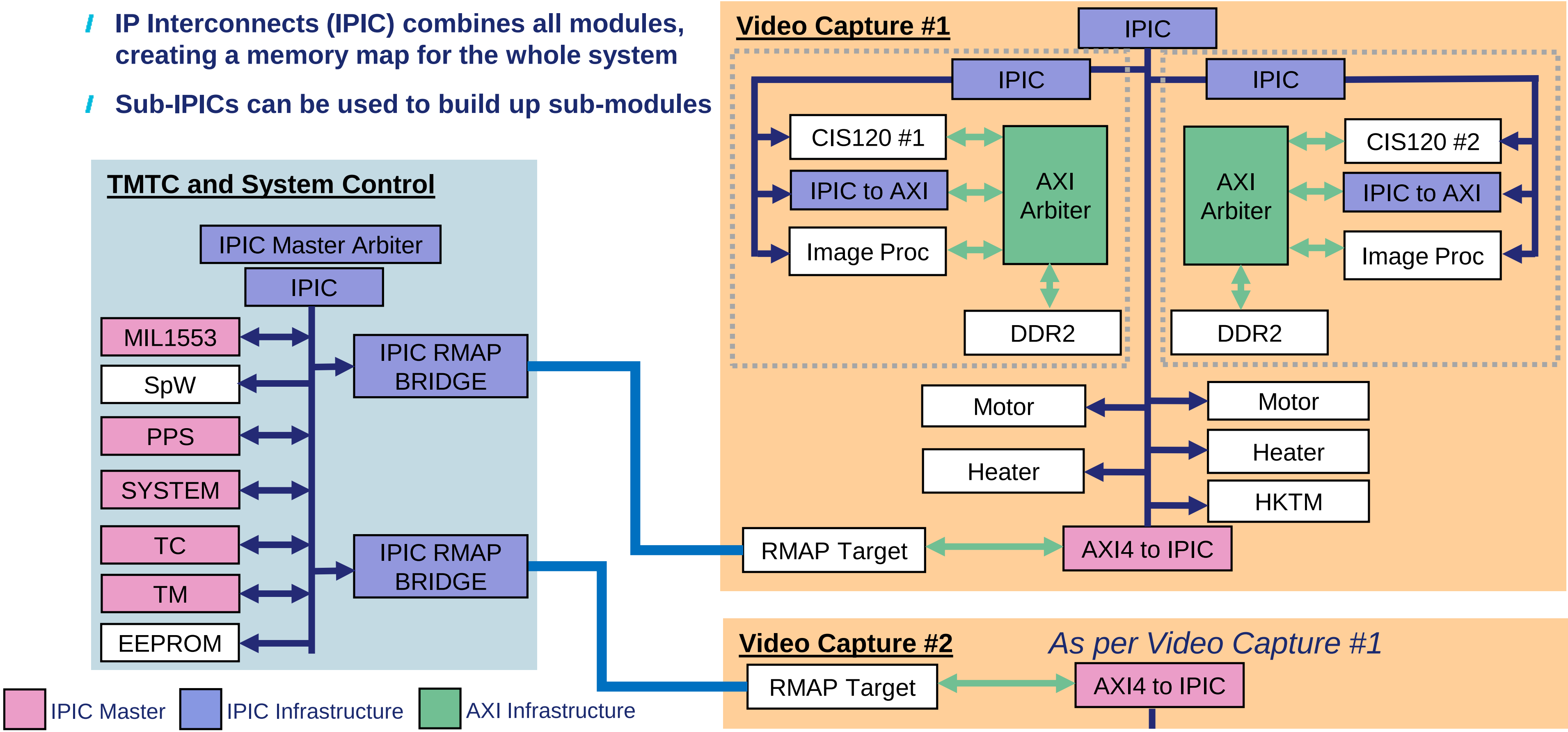
- **IP Interconnects (IPIC) combines all modules, creating a memory map for the whole system**
- **Sub-IPICs can be used to build up sub-modules**



PROPRIETARY INFORMATION

CO2M MAP FPGA ARCHITECTURE #3

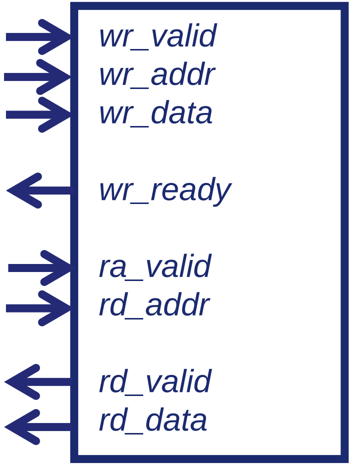
- IP Interconnects (IPIC) combines all modules, creating a memory map for the whole system
- Sub-IPICs can be used to build up sub-modules



IP INTERCONNECT CONCEPT

IP INTERCONNECT CONCEPT

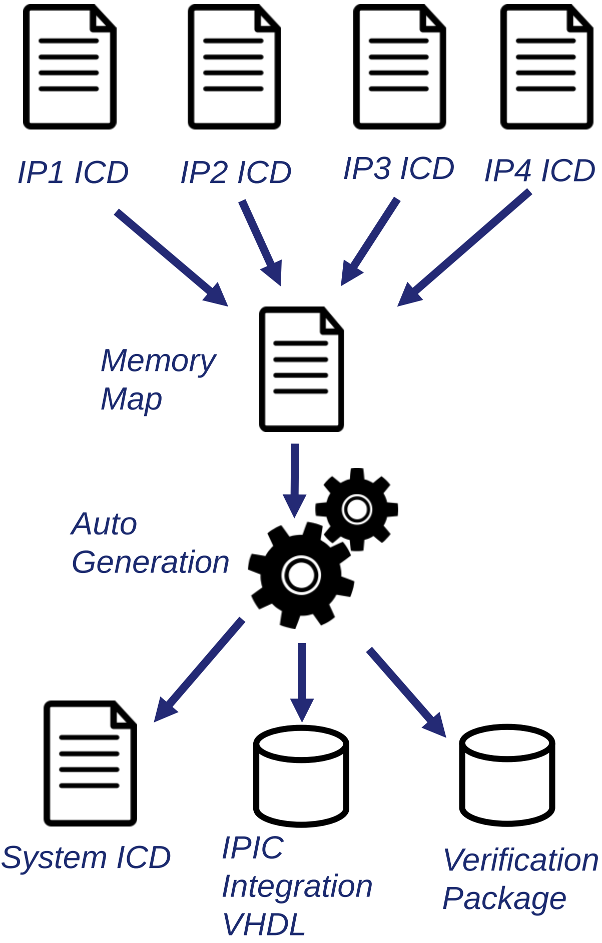
- **An IPIC interface is simply a data word (width configurable) with some simple handshaking signals**
  - The aim is to create a minimal memory-mapped interface that can still provide flow control
  - Have an interface and interconnect that is cheap enough that it can be added to as many modules as possible with minimal resource overhead
  - Allow the designer to optimise the actual interface as required on a module-by-module basis (e.g. adding in CDC, custom flow-control mechanisms)
  - Simple Bus Masters that are resource efficient and easily integrated into FSMs are required
  - Every module with an IPIC interface is a potential reusable module in the future, so extends to small modules such as PPS handlers, ADC acquisition



- **Why not AXI or AXI-Lite?**
  - AXI interfaces are designed specifically for connecting IP cores together on FPGA/ASIC
  - Burst capabilities of AXI too complex to warrant full AXI support at each interface, master design becomes complex, reducing its attractiveness for use with simple modules
  - AXI-Lite is a good fit, but has a minimal data width of 32-bits. Transaction response and protection signals are still required in AXI-Lite
  - Half-supporting a standard isn’t an option, so we created a new one…
- **Vendor AXI Interconnects offer a huge amount of flexibility**
  - But are highly complex and also opaque to designers, this is difficult to verify
  - Present a design timing risk, limited options if a path fails within the interconnect late in the design process...

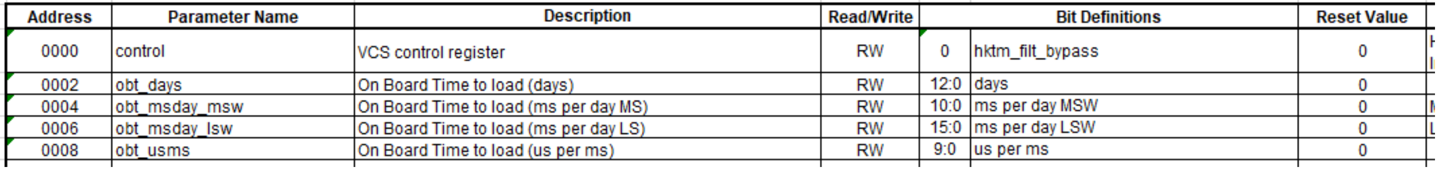
IPIC GENERATOR TOOL

- **For modular design, the configuration of the interconnect is as important as the interface itself**
  - The IPIC framework includes infrastructure generation tools for specifying both the registers of the individual IPs, and also the structure of the interconnects.
- **Each IPIC IP core has an ICD defining the register interface of the IP core**
  - This forms standard documentation of the IP core
- **The designer imports the base ICDs for the IP cores that they are using and creates a memory map**
  - ICDs output from the tool can be imported to create hierarchical IPIC structures
- **Outputs from the tool are then auto-generated, creating an overall system ICD, VHDL for integration and testbench packages for verification**
- **This provides traceability and configuration control across the whole IPIC design**
  - An updated ICD for an IP core can easily be re-imported and the outputs regenerated with a single mouse click.
  - Reduces designer effort and risk of a stale configuration being used in the design or documentation
  - Single point of truth, VHDL, ICDs and testbenches cannot get out of date with each other



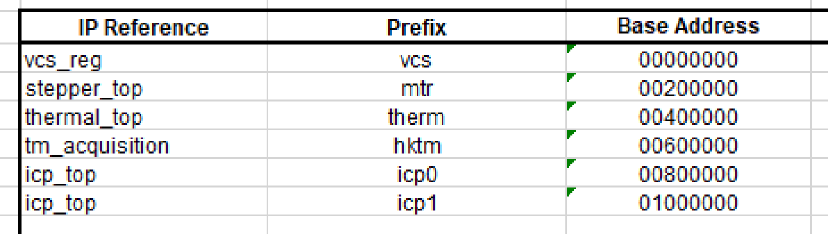
IP INTERCONNECT EXAMPLE

- A ICD template for each IP core is created and the register definition created:

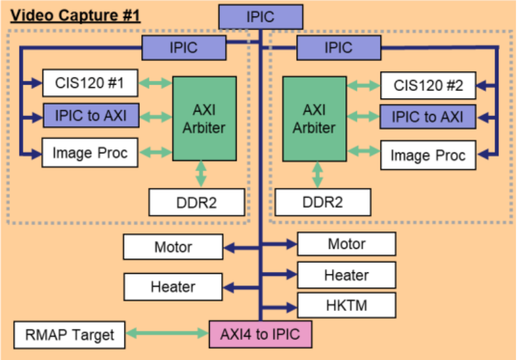


Address	Parameter Name	Description	Read/Write	Bit Definitions		Reset Value
0000	control	VCS control register	RW	0	hktm_filt_bypass	0
0002	obt_days	On Board Time to load (days)	RW	12:0	days	0
0004	obt_msday_msw	On Board Time to load (ms per day MS)	RW	10:0	ms per day MSW	0
0006	obt_msday_lsw	On Board Time to load (ms per day LS)	RW	15:0	ms per day LSW	0
0008	obt_usms	On Board Time to load (us per ms)	RW	9:0	us per ms	0

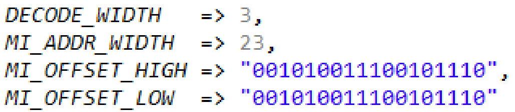
- These ICDs are then imported by the designer to create the memory maps:



IP Reference	Prefix	Base Address
vcs_reg	vcs	00000000
stepper_top	mtr	00200000
thermal_top	therm	00400000
tm_acquisition	hktm	00600000
icp_top	icp0	00800000
icp_top	icp1	01000000



- The tool can then auto generate the VHDL parameters required for configuring the IP Interconnect module:



```
DECODE_WIDTH   => 3,
MI_ADDR_WIDTH  => 23,
MI_OFFSET_HIGH => "001010011100101110",
MI_OFFSET_LOW  => "001010011100101110"
```

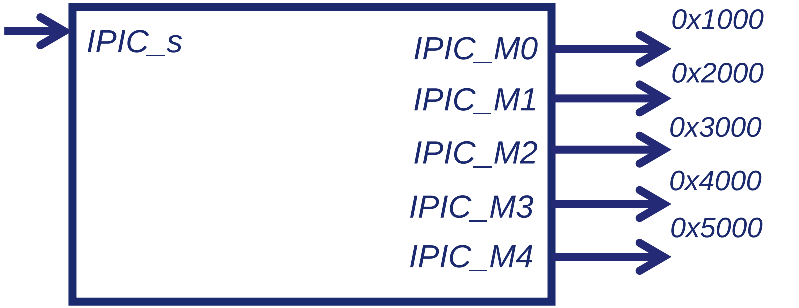
- The tool can be rerun when any updates to the system memory map or the IPs occur, these changes then are automatically propagated to the design, documentation and to the verification package

IP INTERCONNECT TOOLBOX

IPIC CORE MODULES

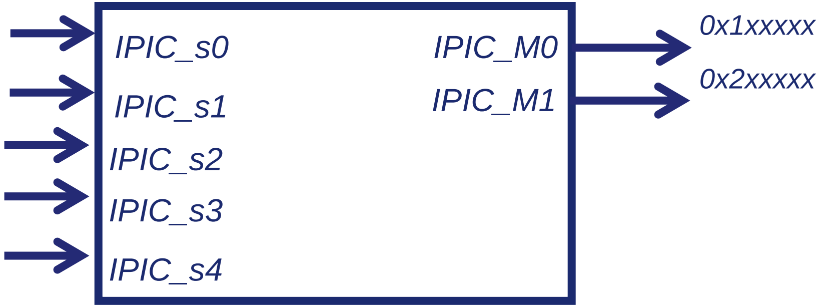
IP Interconnect

- Provides the IP interconnect, connects on IPIC slave interface to a configurable number of masters
- Single cycle pipeline
- 300 LUTs, 130 FFs for a 6 port IPIC.



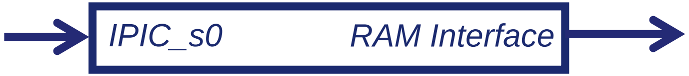
IPIC Master Arbiter

- Connects a number of masters to a number of slaves
- Combined with IP Interconnect to create a multi-master interconnect
- Each master interface is assigned a memory space, addresses are decoded to that memory space
- Simple priority arbitration between masters
- Each master interface is an independent channel, allows multiple transactions to different address spaces in parallel



IPIC RAM interface

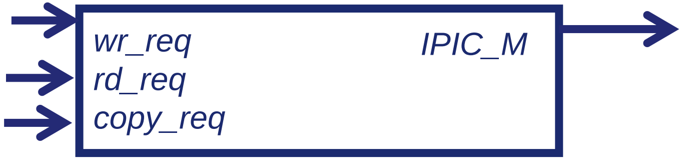
- Decodes between an IPIC slave and a generic embedded-RAM interface with a configurable read delay
- Optional inline scrubber, which will continuously scrub the RAM in the background when the IPIC is not active



IPIC MASTER AND TMTC SEQUENCER

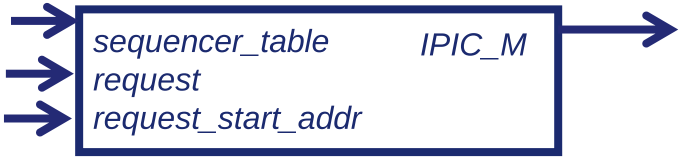
IPIC Master

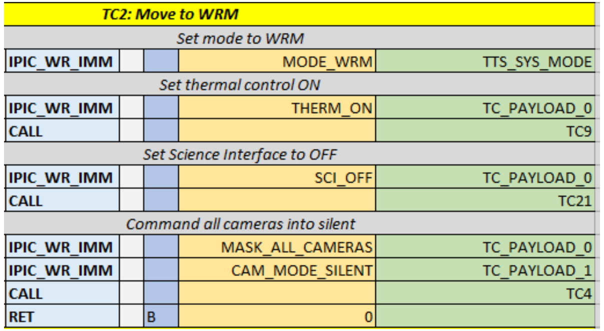
- Performs a single write or read transaction over the IPIC
- Can also copy a number of bits between two addresses
- Provides a simple interface that is easy to integrate into state machines



IPIC Sequencer

- Extended IPIC master that fetches the list of IPIC transactions to perform from a table
- Request address input allows different sequences to be executed depending on location in the table
- Provides IPIC read and write instructions, along with a handful of basic operations useful to TMTC collection and execution (e.g. OR, SHIFT, AND)
- Single-level call stack allows the sequencer to execute from a different region of the table and return at the end of sequence
- Used on CO2M MAP for execution of TCs (sequence of IPIC read/writes per TC) and collection of the TM set
- Sequencer table is loaded from EEPROM on boot, allows patching of TC execution or updates to the TM set collected late in AIT (or even in Flight) if required without reflashing FPGA.



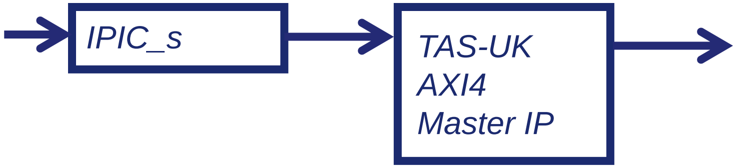


TC2: Move to WRM				
Set mode to WRM				
IPIC_WR_IMM			MODE_WRM	TTS_SYS_MODE
Set thermal control ON				
IPIC_WR_IMM			THERM_ON	TC_PAYLOAD_0
CALL				TC9
Set Science Interface to OFF				
IPIC_WR_IMM			SCI_OFF	TC_PAYLOAD_0
CALL				TC21
Command all cameras into silent				
IPIC_WR_IMM			MASK_ALL_CAMERAS	TC_PAYLOAD_0
IPIC_WR_IMM			CAM_MODE_SILENT	TC_PAYLOAD_1
CALL				TC4
RET		B	0	

IPIC BRIDGES

IPIC to AXI4

- Interfaces the IPIC data and control signals to the TAS-UK AXI4 master IP core
- Address and data width conversion may be required within the bridge

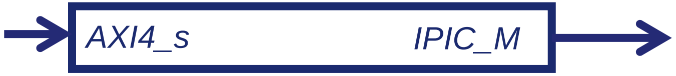


IPIC_s

TAS-UK AXI4 Master IP

AXI4 to IPIC Master

- Maps the AXI data and control signals to their IPIC equivalents
- Address and data width conversion may be required

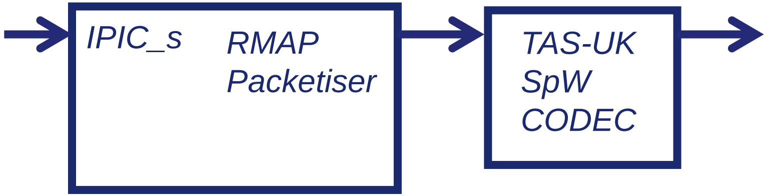


AXI4_s

IPIC_M

IPIC to RMAP

- This allows the IPIC to be extended off-device through RMAP
- This includes control of devices with a memory map, but without an IPIC at the remote end
- Allows IPIC masters (e.g. MIL1553 handler) to control devices via RMAP
- When an IPIC transaction to slave is received, a RMAP packet is created to access to remote address. The RMAP reply is awaited before closing down the IPIC transaction.
- The TAS-UK SpW CODEC is coupled with this bridge to send the generated RMAP packet over SpW.



IPIC_s

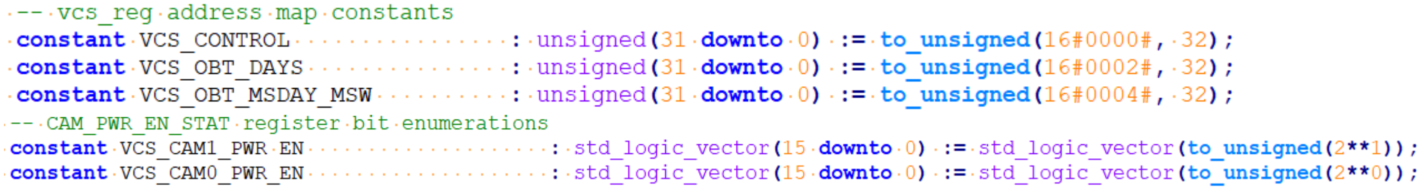
RMAP Packetiser

TAS-UK SpW CODEC

IPIC VERIFICATION IP

Register Map

- The ICIP generation tools will generate a package with all of the addresses and bit defniitions in the design:

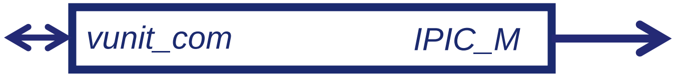


```
 -- vcs_reg address map constants
 constant VCS_CONTROL                : unsigned(31 downto 0) := to_unsigned(16#0000#, 32);
 constant VCS_OBT_DAYS               : unsigned(31 downto 0) := to_unsigned(16#0002#, 32);
 constant VCS_OBT_MSDAY_MSW          : unsigned(31 downto 0) := to_unsigned(16#0004#, 32);
-- CAM_PWR_EN_STAT register bit enumerations
constant VCS_CAM1_PWR_EN                  : std_logic_vector(15 downto 0) := std_logic_vector(to_unsigned(2**1));
constant VCS_CAM0_PWR_EN                  : std_logic_vector(15 downto 0) := std_logic_vector(to_unsigned(2**0));
```

- Test benches can access parts of the design directly using these constants
- If the memory map is changed or an IP core updated, then the package is auto-regenerated with the updated addresses
- No need to update memory address in verification environment, saves a large amount of time and potential errors!

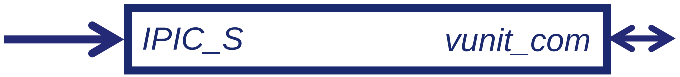
VUnit IPIC_M BFM

- An IPIC master compatible with VUnit message passing API



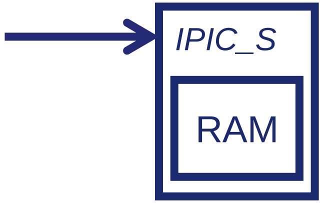
VUnit IPIC_S BFM

- An IPIC slave compatible with Vunit message passing API



IPIC_S Sparse Memory Model

- A generic memory space to which any address can be read/written to
- Stand in for other memory-mapped slaves for unit testing
- Test fake for internal/external RAMs, EEPROMs etc.



SUMMARY

SUMMARY

- **Low overhead IP Interconnects allow a highly modular design process to be followed**
  - Every module in the design can have a register interface – ease the transition from Visio to HDL!
  - Interconnects and interfaces are simple enough that they can be tweaked on a module-by-module basis
  - Performance is limited by lack of bursts, max timing is also limited by simple the interconnect scheme. But the role of the IPIC is for configuration not data transfer.
  - High-performance interconnects are still used where performance or low-latency is required and can be bridged into IPIC interconnects

- **TAS-UK framework provides a complete "toolbox" for modular design using IPIC**
  - Starting at module specification using the ICD creator
  - Then at system architecture using the memory map generator (using the outputs from the ICDs)
  - Through to verification, providing guaranteed correct memory maps that are easily updated as the design evolves
  - Allows easy unit-level development and testing
  - Easy learning curve for new engineers, due to simplicity of interface and direct correlation between configuration files and outputs

- **Proven use on the CO2M MAP instrument**
  - Has allowed very quick design turnaround of about 18 months for a complex instrument
  - Builds a library of small reusable IP modules with flight heritage