Lecture 04
AXI Traffic Generation (ATG)

Electrical Engineering Department
Pontificia Universidad Católica de Chile
peclab.ing.uc.cl

AXI Traffic Generator

What is a ATG?

- It is a IP developed by vivado to test AXI protocol in all its vesions.
- The IP-Core is fully synthesizable AXI4 compliant. That means that you can generate bitstream and introduce it into the PL of your Zynq.
- Supports dependent/independent transaction between read/write master port with configurable delays.
- External start/stop signals to generate traffic without processor intervention.
- Generates IP-specific traffic on AXI interface for pre-defined protocols.

The architecture of the core is broadly separated into a Master and Slave block, and each contains the Write and Read blocks.

AXI Traffic Generator

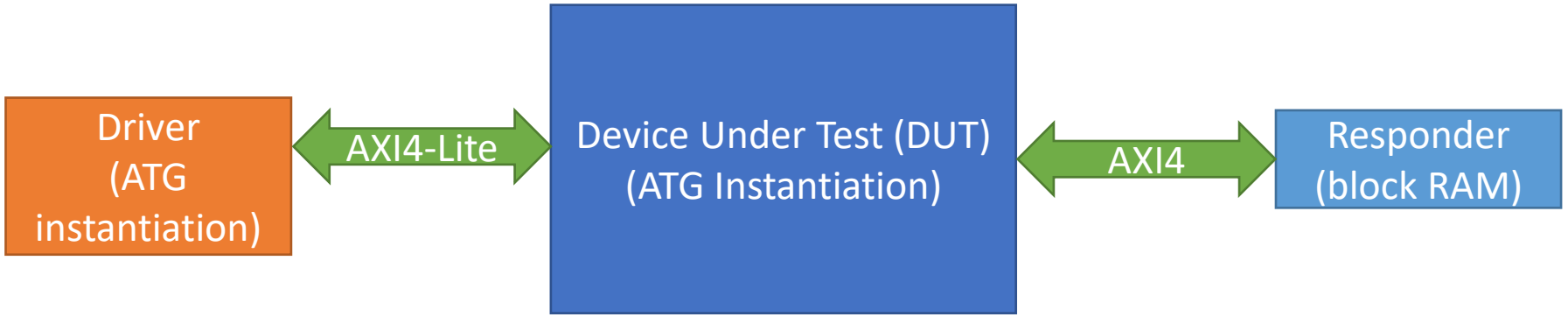
What is a ATG?

- To understand this laboratory you must read **pg125-axi-traffic-gen.pdf** available on github document, complementary to this lecture.
- The ATG IP-Core has 2 profile selection modes:
  - **Custom:** This mode allows you to select different AXI4 interface traffic generation. The available options are: **AXI4, AXI4-Stream, and AXI4-Lite.** The following modes are possible:
    - **Advance mode**
    - **Basic Mode**
    - **Static Mode**
    - **System Init/Test Mode**
    - **Streaming Mode**
  - **High level traffic:** This mode allows you to generate IP specific traffic on the AXI interface for pre-defined protocols. The currently supported traffic profiles include:
    - Video Mode; PCIe® Mode; Ethernet Mode; USB Mode; Data Mode

AXI Traffic Generator

The ATG Example

- The following example Will be used to exaplain the difference between Advance mode and Testing Mode configurations of ATG.
- First ATG is named driver and it is a custom profile Test Mode ATG.
- Second ATG is named DUT and it is a custom profile Advanced Mode ATG



Driver: An instantiation of the ATG in AXI4-Lite mode. This module is used to generate AXI4 Lite transactions to program the DUT module.
DUT: Device under test. A second instantiation of an ATG, this time in AXI4 mode. This module is used to generate AXI4 transaction to the Responder.
Responder: An instantiation of a block RAM controller that will accept the generated traffic from the ATG. This differs from the block RAM transaction buffers that are internal to the two ATGs.

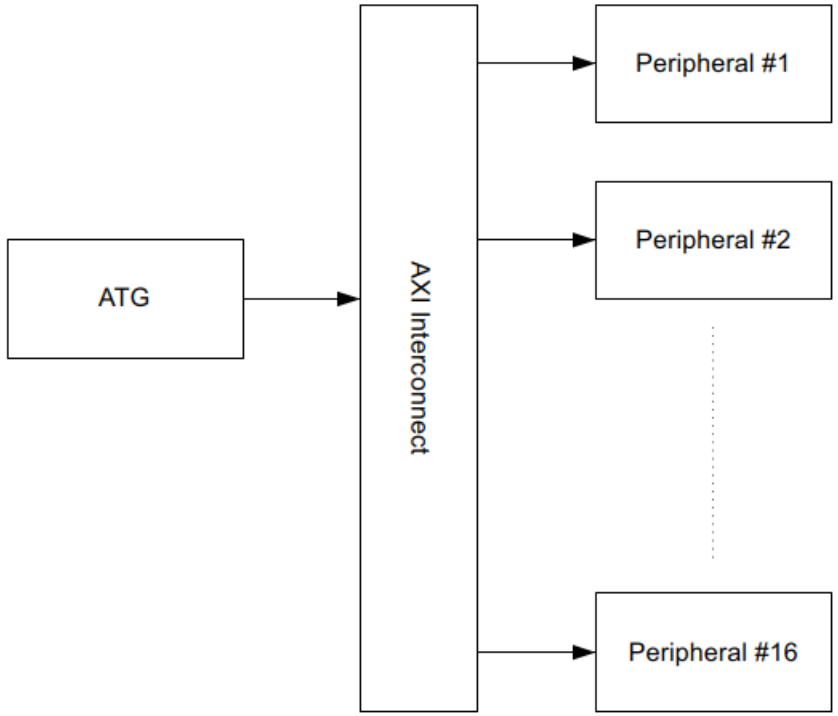
Custom Profile
System Init/Test Mode

XILINX®

AXI Traffic Generator

ATG IP-Core with Custom Profile and System Init/Test Mode

The System Test mode is just an extention of the System Init mode.



- This mode provides an **AXI4-Lite Master interface.**
- It is used to **initialize the system peripherals with preconfigured values** (coefficient (COE) files).
- After the core comes out of reset, it reads the coefficient (COE) files (address and data) from the ROM and generates AXI4-Lite transactions.
- You must provide two COE files for this mode. Entries in all the COE files are 32-bit.
- **Address COE File** – Provides the sequence of addresses to be issued
- **Data COE File** – Provides the sequence of data corresponding to the address specified in Address COE File

AXI Traffic Generator

ATG IP-Core with Custom Profile and System Init/Test Mode

POWER & ENERGY CONVERSION LABORATORY

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

OPERATION System Init Mode:

1. After AXI Traffic Generator (ATG) comes out of reset, it reads the ADDR and DATA ROMs.
2. It initiates AXI4-Lite write transactions to a specified address and data in the COE files.
3. The core goes to idle state after AXI4-Lite transactions are issued.

Note1: The number of entries in the COE file can be: 16, 32, 64, 128 or a maximum of 256.

Note2: Introducing the address FFFFFFFF in an address Coe file the core stops generating further transactions. At least you need one NOP within address COE File.

ATG IP-Core with Custom Profile and System Init/Test Mode

System Test Mode:

1. It is an improvement to the System Init mode. Enabling read transactions.

2. Also allow the incorporation of instructions for the transactions, included in a new file, named Control COE file. We can only set reading or writing instructions. This is limited, but good enough for most of simple purpose applications.

3. It also incorporate a MASK COE File.

AXI Traffic Generator

ATG IP-Core with Custom Profile and System Init/Test Mode

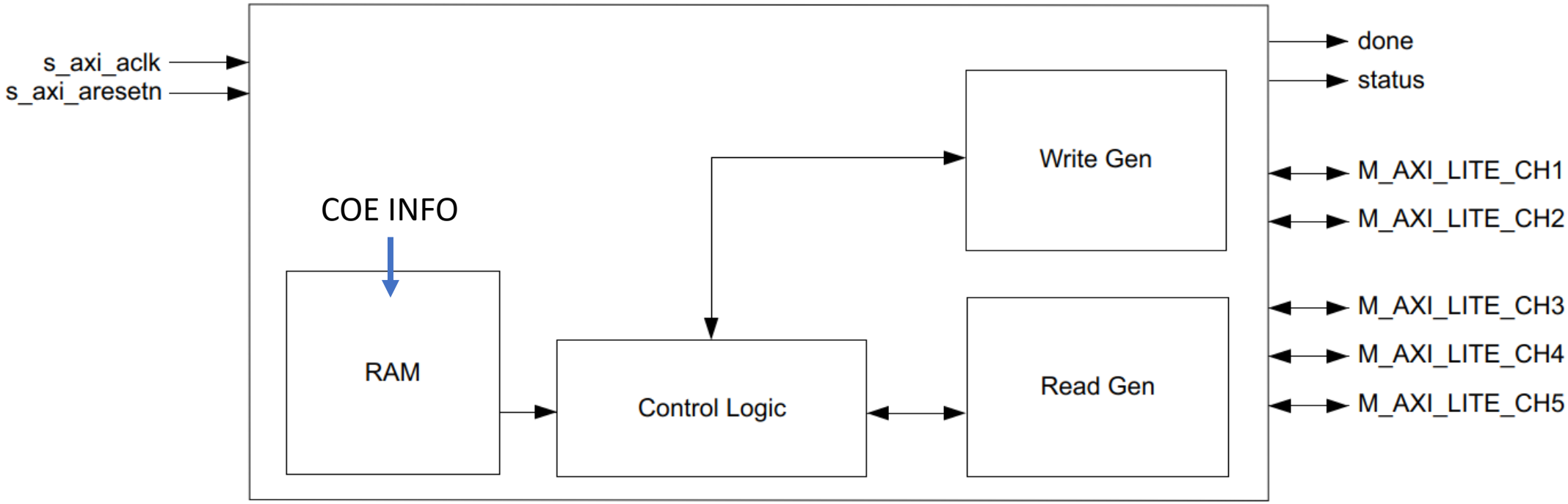


Figure 1-2: **AXI4-Lite Traffic Generator Block Diagram**

ATG IP-Core with Custom Profile and System Init/Test Mode

Control COE File

Table 1-14: **Control COE File – 32-bit Control information**

Bits	Description
31:18	Reserved. Must be filled to zeros.
17	Count as Error Checks the status of the transaction. For Write: BRESP is monitored to be OKAY. For Read: RDATA compared against the entry in Data COE File. 0 = check the BRESP/RDATA and do not increment error counter 1 = check the BRESP/RDATA and increment error counter
16	0 = read transaction 1 = write transaction
15:8	Next COE entry to be fetched upon successful completion of the current transaction.
7:0	Next COE entry to be fetched if the current transaction failed.

ATG IP-Core with Custom Profile and System Init/Test Mode

MASK COE FILE:

It mask a data before comparing read data with expected data.

- Mask bit value of 1 indicates the corresponding bit is used for comparing incoming read data with expected data.
- Mask bit value of 0 indicates the corresponding bit is not used for comparing incoming read data with expected data.

Note 3: If more than 256 transaction are needed, then you can cascade several ATG IP-Core with custom profile and System Init/Test Mode.

ATG IP-Core with Custom Profile and System Init/Test Mode

OPERATION Test Mode:

1. After AXI Traffic Generator (ATG) comes out of reset, it reads the ADDR and DATA ROMs.
2. It initiates AXI4-Lite transactions defined in CONTROL COE FILE to a specified address and data in the COE files.
3. The core goes to idle state after AXI4-Lite transactions are issued.

Note4: If read transactions are issued, value is saved in data ROM and masked as MASK COE file.

AXI Traffic Generator

Example Test Mode ATG Configuration

First four lines in Control COE File

```
memory_initialization_radix = 16;
memory_initialization_vector =
00020100
00010201
00010302
00010403
00010504
```

First four lines in Address COE File

```
memory_initialization_radix = 16;
memory_initialization_vector =
00000000
00008000
00008004
00008008
0000800C
```

First four lines in Data COE File

```
memory_initialization_radix=16;
memory_initialization_vector=
20000000
00000000
80002402
00006400
00000000
```

'00000000000000 1 0 00000001 00000000'
'00000000000000 0 1 00000010 00000001'
'00000000000000 0 1 00000011 00000010'
'00000000000000 0 1 00000100 00000011'

This means:

- First transaction made by the Master AXI Lite driver is a read transaction. (control COE bit 16 is 0)
- First read transaction reads the address 0x0, located in the slave peripheral.
- First (expected) value read is 0x20000000
- Second instruction is a write transaction. Data 0x0 will be written in address 0x8000 at the slave peripheral through AXI-Lite.
- Third instruction is a write transaction. Data 0x80002402 will be written in address 0x8004 at the slave peripheral through AXI-Lite.

POWER & ENERGY CONVERSION LABORATORY

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Custom Profile
Advanced Mode

XILINX®

ATG IP-Core with Custom Profile and Advanced Mode

Advanced mode allows full control over the traffic generation.

Control register are provided to program the core and generate different AXI4 transactions.

It possesses three internal RAMs:

- Command RAM (CMDRAM)
- Parameter RM (PARAMRAM)
- Master RAM (MSTRAM)
- Address RAM (ADDRRAM)

Note1: No COE files are associated in this mode.

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

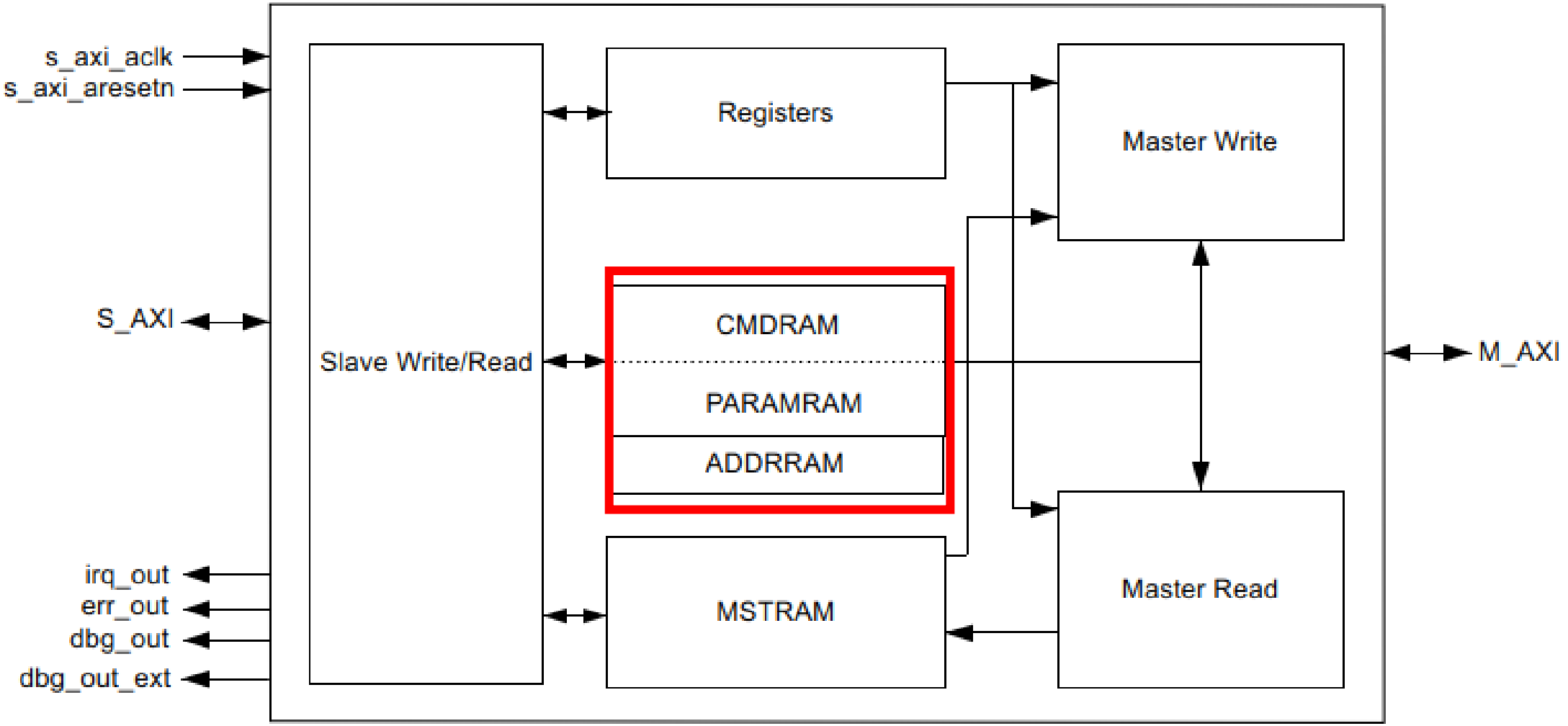


Figure 1-1: **AXI4 Traffic Generator Block Diagram**

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

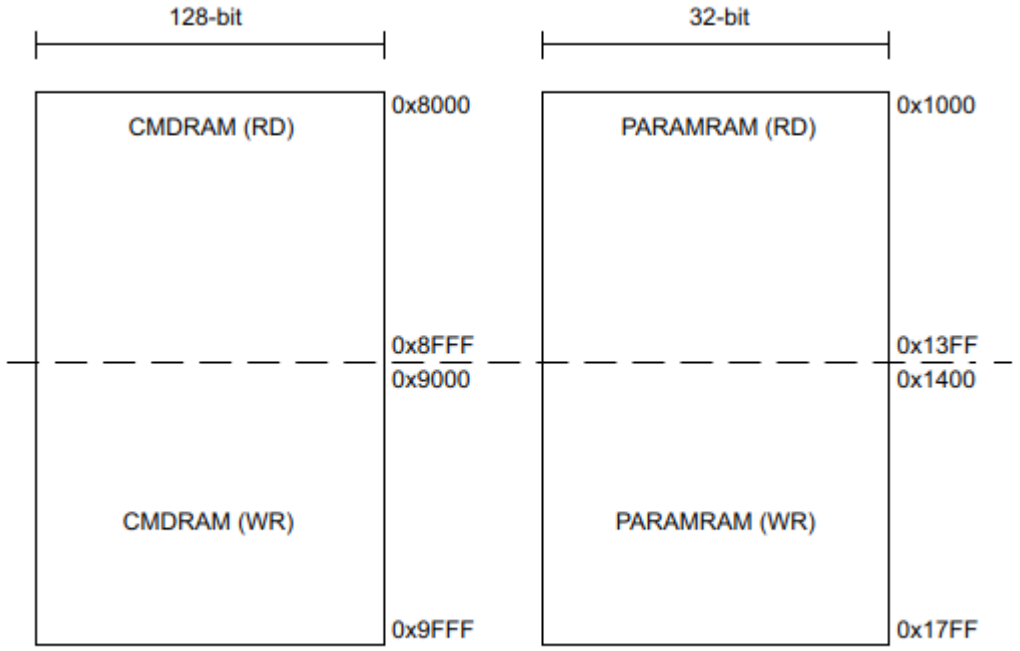


Figure 1-4: **PARAMRAM vs. CMDRAM**

Command RAM:

- Divided into two 4kB regions.
- One region for reads commands (0x8000-0x8FFF)
- One region for write commands (0x9000-0x9FFF)
- Each command is composed of 128bits (16Bytes).
- Each region has 1024 positions. Each position save a 32bits word. Then each region can hold 256 commands.
- Read and write commands are executed independently.
- Access to CMDRAM is prohibited after master logic core is enabled (Bit[20] of Master Control register).
- The details of the 128bits of command are described in next slide.

For **full detail** of command bits see p.8-9 here.

Memoria CDAM	Dirección de Memoria CDAM
Position 1024 (32bits; 4 bytes)	0x8000 a 0x800C
Position 1023 (32bits; 4 bytes)	0x800D a 0x8010
⋮	⋮
Position 1 (32bits; 4 bytes)	0x8FFC a 0x8FFF

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Word Offset	Bits	Description
+00	31:0	**AXI_Address[31:0]**: Address to drive on ar_addr or aw_addr (a*_addr[31:0])
+01	31	**Valid_cmd**(1): When set, this is a valid command. When clear, halt the master logic for this request type (read or write).
	30:28	**last_addr[2:0]**: Should be set to 0 for C_M_AXI_DATA_WIDTH > 64. For writes, indicates the valid bytes in the last data cycle. **64-bit mode**: 000 = All bytes valid 001 = Only Byte 0 is valid 010 = Only Bytes 0 and 1 are valid ... **32-bit mode**: 000 = All bytes valid 100 = Only Byte 0 is valid 101 = Only Bytes 0 and 1 are valid 110 = Only Bytes 0, 1, and 2 are valid
	27:24	Reserved
	23:21	**Prot[2:0]**: Driven to a*_prot[2:0]
	20:15	**Id[5:0]**: Driven to a*_id[5:0]
	14:12	**Size[2:0]**: Driven to a*_size[2:0]
	11:10	**Burst[1:0]**: Driven to a*_burst[1:0]
	9	Reserved
	8	**Lock**: Driven to a*_lock
	7:0	**Len[7:0]**: Driven to a*_len[7:0].

Address where AXI is going to read or write.

Size define the Bytes involved in the transaction
Typically 32 bits.

Table A3-2 Burst size encoding

AxSIZE[2:0]	Bytes in transfer
0b000	1
0b001	2
0b010	4
0b011	8
0b100	16
0b101	32
0b110	64
0b111	128

For **full detail** of command bits see p.8-9 here.
For **full detail** of AX4 Signals Chapter A.2. here.

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Word Offset	Bits	Description
+00	31:0	**AXI_Address[31:0]**: Address to drive on ar_addr or aw_addr (a*_addr[31:0])
+01	31	**Valid_cmd**(1): When set, this is a valid command. When clear, halt the master logic for this request type (read or write).
	30:28	**last_addr[2:0]**: Should be set to 0 for C_M_AXI_DATA_WIDTH > 64. For writes, indicates the valid bytes in the last data cycle. **64-bit mode**: 000 = All bytes valid 001 = Only Byte 0 is valid 010 = Only Bytes 0 and 1 are valid ... **32-bit mode**: 000 = All bytes valid 100 = Only Byte 0 is valid 101 = Only Bytes 0 and 1 are valid 110 = Only Bytes 0, 1, and 2 are valid
	27:24	Reserved
	23:21	**Prot[2:0]**: Driven to a*_prot[2:0]
	20:15	**Id[5:0]**: Driven to a*_id[5:0]
	14:12	**Size[2:0]**: Driven to a*_size[2:0]
	11:10	**Burst[1:0]**: Driven to a*_burst[1:0]
	9	Reserved
	8	**Lock**: Driven to a*_lock
	7:0	**Len[7:0]**: Driven to a*_len[7:0]

Burst define the type of Burst Implemented. This define how the following address is calculated.
Most common is Incremental (INCR)

Table A3-3 Burst type encoding

AxBURST[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	Reserved

More info here

Lenght of the burst. Len[7:0]+1

For **full detail** of command bits see p.8-9 here.
For **full detail** of AX4 Signals Chapter A.2. here.

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Table 1-2: **CMDRAM Memory Format *(Cont'd)***

Word Offset	Bits	Description
+02	31	Reserved
	30:22	**My_depend[8:0]**: This command does not begin until this master logic has at least completed up to this command number. A value of zero in this field means do not wait. This allows a command to wait until previous commands have completed for ordering.
	21:13	**Other_depend[8:0]**: This command does not begin until the other master logic has completed up to this command number. For example, if a write command had 0x04 in this field, the write would not begin until the read logic had at least completed its commands (CMDs) 0x00 through 0x03. A value of 0 in this field means do not wait, but commands can only be started in order for each master type. For example, if Write CMD[0x05] waits for Read 0x03, then Write CMD[0x06] cannot start until Read 0x03 completes as well. A read completes when it receives the last cycle of data, and a write completes when it receives BRESP.
	12:0	**Mstram_index[12:0]**(2): Index into MSTRAM for this transaction (reads will write to this MSTRAM address, writes take data from this address)
+03	31:20	Reserved
	19:16	**qos[3:0]**: Driven to a*_qos[3:0]
	15:8	**user[7:0]**: Driven to a*_user[7:0]
	7:4	**cache[3:0]**: Driven to a*_cache[3:0]
	3	Reserved
	2:0	**Expected_resp**: 0x0 to 0x1 = Only OKAY is allowed 0x2 = Only EX_OK is allowed 0x3 = EX_OK or OKAY is allowed 0x4 = Only DECERR or SLVERR is allowed 0x7 = Any response is allowed

For **full detail** of command bits see p.8-9 here.
For **full detail** of AX4 Signals Chapter A.2. here.

Index define the MSTRAM position of the saved data.

It has 8kB of internal RAM (0xC000 – 0xCFFF)

Index is the position of the data.

For example if we set data lenght as 32bits. Then **Index 4** means the data saved on the fifth position counting in 32bits steps. (Index=0 is first position)

Thus, considering that memory is byte-adressable, it would be memory address:

0xC000 to 0xC003 (first data; Index=0)
0xC004 to 0xC007 (second data ; Index=1)
0xC008 to 0xC00B (third data ; Index=2)
0xC00C to 0xC00F (fourth data ; Index=3)
0xC010 to 0xC013 (fifth data ; Index=4)

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Example of Data in CMDRAM:

```
% Data in address 0x9000 to 0x902F (write operation):        0x00000000 0x80002402 0x00000000 0x00000000 (address in CMDRAM 0x9000-0x900F) (16 bytes, i.e. 16 addresses
                                                             0x00000040 0x80002403 0x00000010 0x00000000 (address in CMDRAM 0x9010-0x901F)
                                                             0x00000080 0x80002403 0x00000020 0x00000000 (address in CMDRAM 0x9020-0x902F)

%(+00) AXI_Address%                          %(+01)AXI Size (010=4Bytes), Burst=(Incrementing), Len=X+1=3%                %(+02) INDEX [12:0]%
'00000000000000000000000000000000' '1000000000000000 (010) (01) 0 0 (00000010)' '0000000000000000 0000000000000' '00000000000000000000000000000000'
'00000000000000000000000001000000' '1000000000000000 (010) (01) 0 0 (00000011)' '0000000000000000 0000000010000' '00000000000000000000000000000000'
'00000000000000000000000010000000' '1000000000000000 (010) (01) 0 0 (00000011)' '0000000000000000 0000000100000' '00000000000000000000000000000000'
```

First write Command most important information is:

- Command a write operation to AXI4. (word written is read from MSTRAM). **Write data to RAM address 0x0**
- **SIZE:** AXI word size set to 4bytes or 32 bits.
  BURST: AXI burst type set as incremental.
- **Len:** length of the burst is set to 2+1=3 words per transaction.
- **Index:** MSTRAM data position 0^{th}. As words are defined as 32bits. This represents address (C0000+ Hex(4*0=0)) 0xC0000 to 0xC0003 of MSTRAM.

Second write Command most important information is:

- Command a write operation to AXI4. (word written is read from MSTRAM). **Write data to RAM address 0x40**
- **SIZE:** AXI word size set to 4bytes or 32 bits.
  BURST: AXI burst type set as incremental.
- **Len:** length of the burst is set to 3+1=4 words per transaction.
- **Index:** MSTRAM data position 16^{th}. As words are defined as 32bits. This represents address (C0000+ Hex(4*16=64)) 0xC0040 to 0xC0043 of MSTRAM.

Third write Command ???-→ you can understand it know!

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Example of Data in CMDRAM:

```
% Data in address 0x8000 to 0x801F (read operation):      0x00000000 0x80002402 0x00006400 0x00000000  (address in CMDRAM 0x8000-0x800F)
                                                          0x00000040 0x80002403 0x00000010 0x00000000  (address in CMDRAM 0x8010-0x801F)

%(+00) AXI_Address%                    %(+01)AXI Size (010=4Bytes), Burst=(Incrementing), Len=X+1=3%        %(+02) INDEX [12:0]%
'00000000000000000000000000000000' '1 000 0000 000 000000 (010) (01) 0 0 (00000010)' '0000000000000000 0010000000000' '00000000000000000000000000000000'
'00000000000000000000000001000000' '1 000 0000 000 000000 (010) (01) 0 0 (00000011)' '0000000000000000 0000000010000' '00000000000000000000000000000000'
```

First read Command most important information is:

- Command a **read operation** to AXI4. (word read is written into MSTRAM). **Read data from RAM address 0x0**
- **SIZE:** AXI word size set to 4bytes or 32 bits.
  BURST: AXI burst type set as incremental.
- **Len:** length of the burst is set to 2+1=3 words per transaction.
- **Index:** MSTRAM data position 1024th. As words are defined as 32bits. This represents address (C0000+ Hex(4*1024=4096)) 0xC1000 to 0xC1003 of MSTRAM.

Second read Command most important information is:

- Command a **read transaction** to AXI4. (word read is written into MSTRAM). **Read data from RAM address 0x40.**
- **SIZE:** AXI word size set to 4bytes or 32 bits.
  BURST: AXI burst type set as incremental.
- **Len:** length of the burst is set to 3+1=4 words per transaction.
- **Index:** MSTRAM data position 16th. As words are defined as 32bits. This represents address (C0000+ Hex(4*16=64)) 0xC0040 to 0xC0043 of MSTRAM.

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

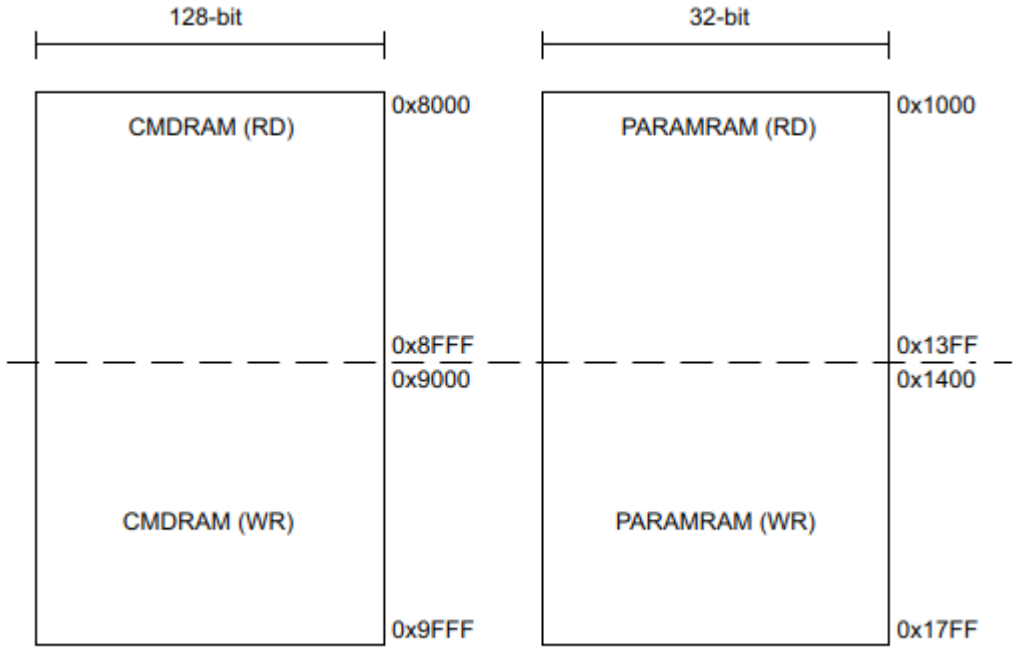


Figure 1-4: **PARAMRAM vs. CMDRAM**

For **full detail** of command bits see p.8-9 here.

PARAM RAM:

- It extends the command programmability by 32bits.
- Only write access is allowed to PARAMRAM
- Some features allow to repeat a command several times or delays before execution.
- For **full detail** of PARAM bits see p.10-11 here.

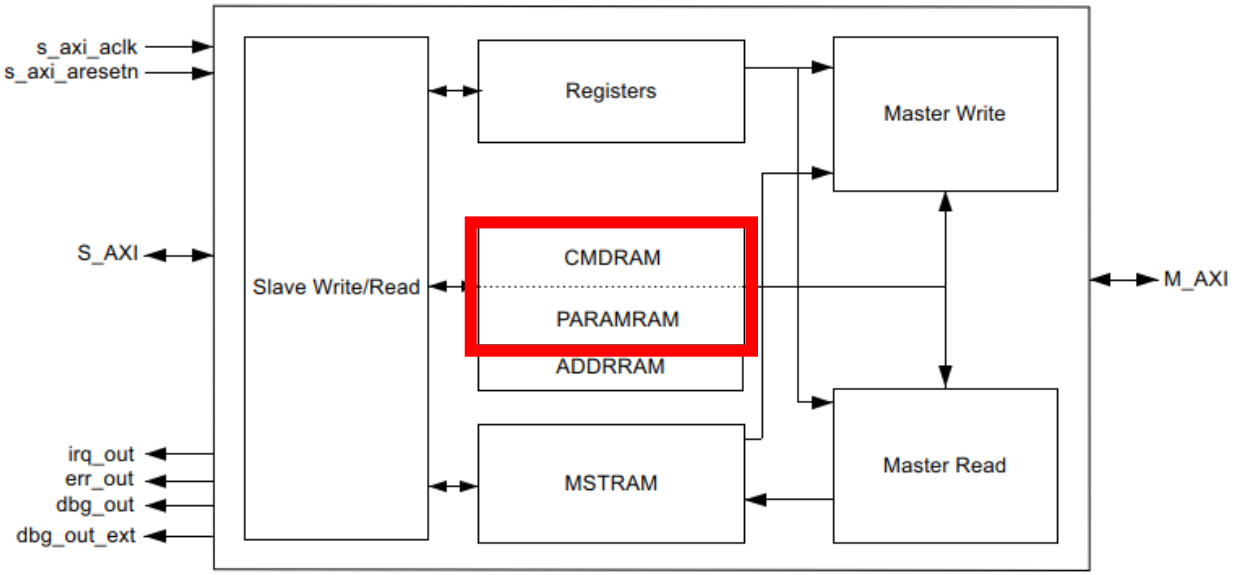


Figure 1-1: **AXI4 Traffic Generator Block Diagram**

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

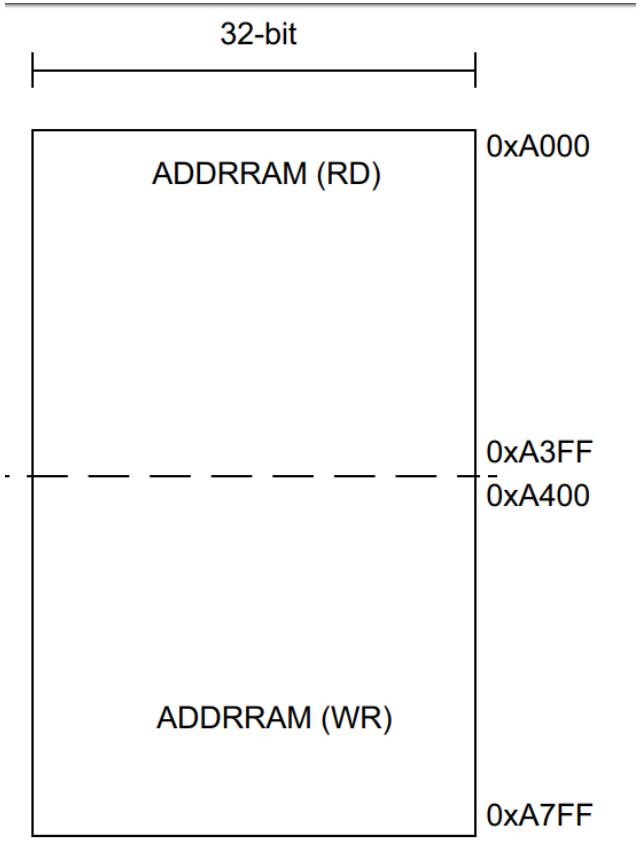


Figure 1-5: **Address RAM**

Address RAM:

- When address width is configured to be >32bits, the address RAM saves the rest of address bits.
- The Address RAM entries correspond to the MSB bits of address and are concatenated to Bits[31:0] in CMDRAM.
- In cases when address width is configured to 32, the Address RAM is not present and cannot be accessed.

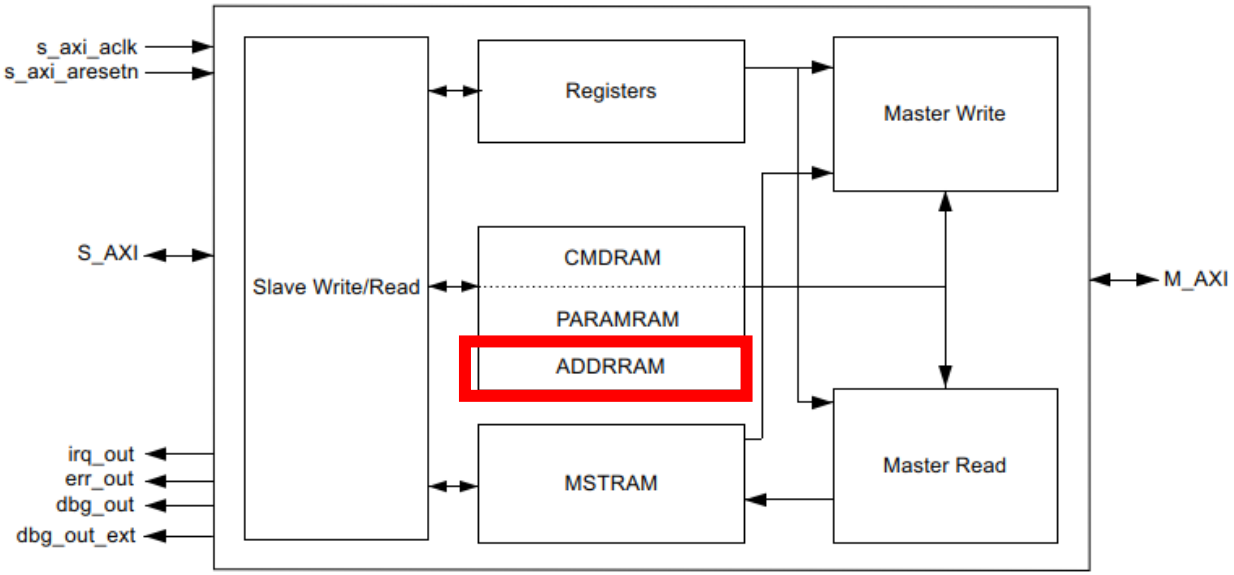
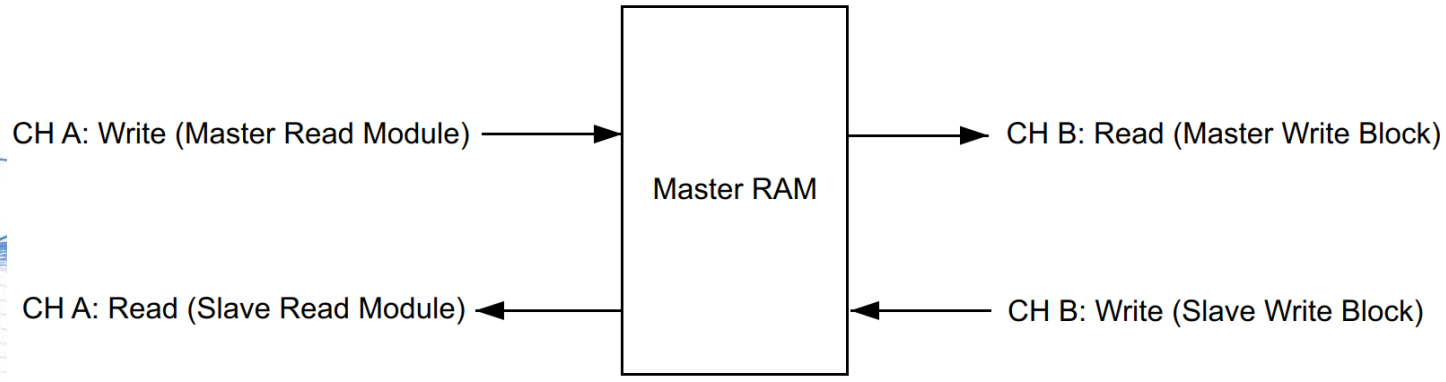


Figure 1-1: **AXI4 Traffic Generator Block Diagram**

ATG IP-Core with Custom Profile and Advanced Mode

Master RAM (MSTRAM):

- It has 8kB of internal RAM (0xC000 – 0xCFFF)
- Take data from this ram for write transactions
- Store data to this RAM for read transactions
- MSTRAM has two channels.
  - A: Master read / slave read (from MSTRAM point of view, these operations are write/read)
  - B: Master write / slave write (from MSTRAM point of view, these operations are read/write)



AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Master RAM (MSTRAM):

- The **index** of MSTRAM separate the different words of data.
- For instance, from address 0xC000 to 0xC007 there are 8 positions of addresss, i.e. 8 bytes of data are saved. A data of length 64bits is saved.
- The index entered in CMDRAM is the position in MSTRAM where the ATG will look for a data in a AXI write operation or where it will write a DATA in a AXI read operation.

Table 1-8: **Write**

Address	Data	MSTRAM Entry Number (Index Entered in CMDRAM Programming)
0xC000	0x11111111	0
0xC004	0x22222222	
0xC008	0x33333333	1
0xC00C	0x44444444	
0xC010	0xABCD1234	2
0xC014	0xFAAB1234	
...		

Table 1-9: **Read**

Address	Data	MSTRAM Entry Number (Index Entered in CMDRAM Programming)
0xC000	0x11111111	0
0xC004	0x22222222	
0xC008	0x33333333	1
0xC00C	0x44444444	
0xC010	0xABCD1234	2
0xC014	0xFAAB1234	
...		

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Master RAM (MSTRAM):

- The **index** of MSTRAM separate the different words of data.
- For instance, from address 0xC000 to 0xC007 there are 8 positions of addresss, i.e. 8 bytes of data are saved. A data of length 64bits is saved.
- The index entered in CMDRAM find read a data in a write operation or to write a DATE in MSTRAM for a read operation of the Master AXI4.

Table 1-8: **Write**

Address	Data	MSTRAM Entry Number (Index Entered in CMDRAM Programming)
0xC000	0x11111111	0
0xC004	0x22222222	
0xC008	0x33333333	1
0xC00C	0x44444444	
0xC010	0xABCD1234	2
0xC014	0xFAAB1234	
...		

Table 1-9: **Read**

Address	Data	MSTRAM Entry Number (Index Entered in CMDRAM Programming)
0xC000	0x11111111	0
0xC004	0x22222222	
0xC008	0x33333333	1
0xC00C	0x44444444	
0xC010	0xABCD1234	2
0xC014	0xFAAB1234	
...		

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

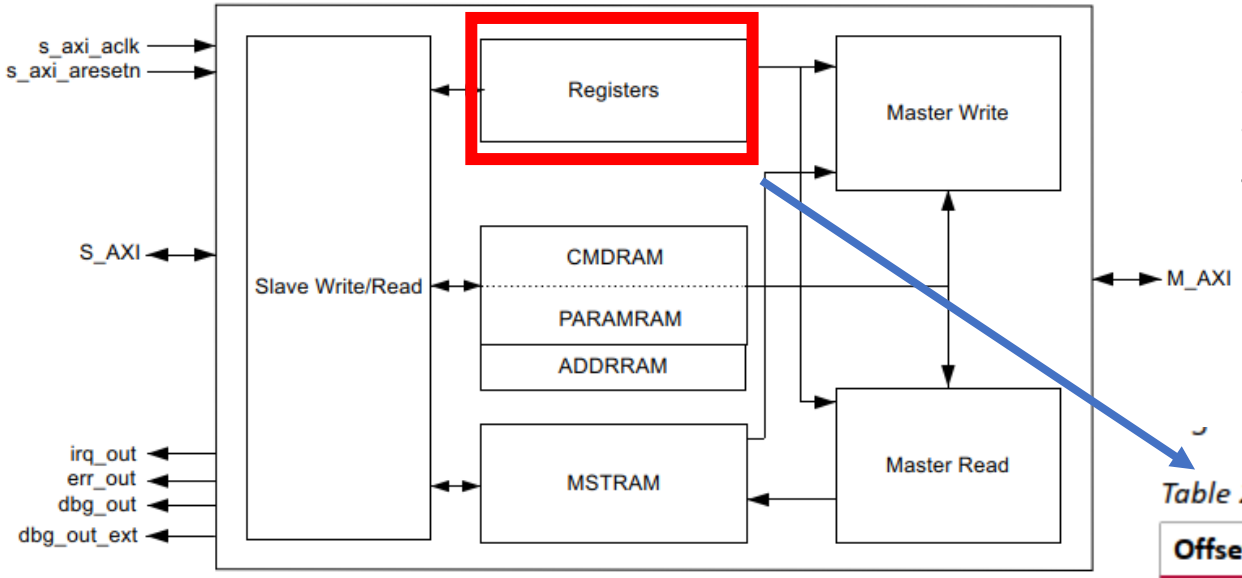


Figure 1-1: **AXI4 Traffic Generator Block Diagram**

A set of Registers help us to check error and operate the AXI4 transactions.
The most important is the **Master Control and Slave Control**.
Master_control[20]: Enable/starts Master transactions.

Table 2-2: **Advanced/Basic Mode Register Map**

Offset	Register Name	Description
0x00	Master Control	To control master logic.
0x04	Slave Control	To control slave logic.
0x08	Error Status	Different errors reported during core operation.
0x0C	Error Enable	Enable register to report intended error.
0x10	Master Error Interrupt Enable	To generate/mask external error interrupt.
0x14	Config Status	Stores the current configuration of the core.
0x18 to 0x2C	Reserved	Reserved
0xB4	Slave Error	Access to this register returns the SLVERR response.

AXI Traffic Generator

ATG IP-Core with Custom Profile and Advanced Mode

Table 2-3: **Master Control (0x00)**

Bits	Name	Reset Value	Access Type	Description
31:24	REV	0x20	R	Revision of the core.
23:21	MSTID	0x0	R	M_ID_WIDTH, where: 0x0 = Indicates 0 or 1-bit width 0x1 = Indicates 2-bit width ... 0x7 = Indicates 8-bit width
20	MSTEN	0x0	R/W	Master Enable When set, the master logic begins. When both the Read and Write state machines complete, this bit is automatically cleared to indicate to software that the AXI Traffic Generator is done.
19	Loop Enable(1)	0x0	R/W	Loop Enable • Loops through the command set created using CMDRAM and PARAMRAM (as applicable) indefinitely when set to 1. • When this bit is reset to 0, core stops looping after the current command set of transactions are completed. • Dependency (if any, both mydepend and otherdepend) is ignored when loop enable is set. Dependency gets honored after the loop enable is reset to 0. • Both channels loopback to their first command independently without waiting for the outstanding transactions to get completed. • If interrupt is enabled, core generates `irq_out` after completing the command set following the reset of loop enable to 0. ***Note:*** Dependency for the last command set run is based on the point at which the loop enable is reset to 0. For example, a command set with 12 writes and 16 reads are present with the 13th read is dependent on sixth write. Now if the loop enable is reset to 0 before sixth write and 13th read of command run, you see the dependency in the last run else the dependency is not seen even after loop enable is reset. For bullet point 4, consider a case of a command set with 50 write commands and two read commands. In such a case, the read command should get repeated more than once before one set of write commands are completed.
18:0	Reserved	N/A	N/A	Reserved

Table 2-4: **Slave Control (0x04)**

Bits	Name	Reset Value	Access Type	Description
31:20	Reserved	N/A	N/A	Reserved
19	BLKRD	0x0	R/W	Enable Block Read When set, slave reads are not processed if there are any pending writes. On completing each write, at least one read data is returned to prevent starvation.
18	DISEXCL	0x0	R/W	Disable Exclusive Access When set, disables exclusive access support and error response ability for reads on Slave Error register.
17	WORDR	0x0	R/W	Enable in Order Write Response When set, forces all BRESPs to be issued in the order the requests were received.
16	RORDR	0x0	R/W	Enable in Order Read Response When set, forces all slave reads to be done in the order received.
15	ERREN	0x0	R/W	Enable Error Generation When set, if any bit in Error Status register Bits[15:0] is set, then err_out is asserted.
14:0	Reserved	N/A	N/A	Reserved

Back to Top

For any furhter information you can consult these files:

1. ATG IP-Core here.
2. AXI3 ; AXI4 Protocol here.

Electrical Engineering Department
Pontificia Universidad Católica de Chile
peclab.ing.uc.cl