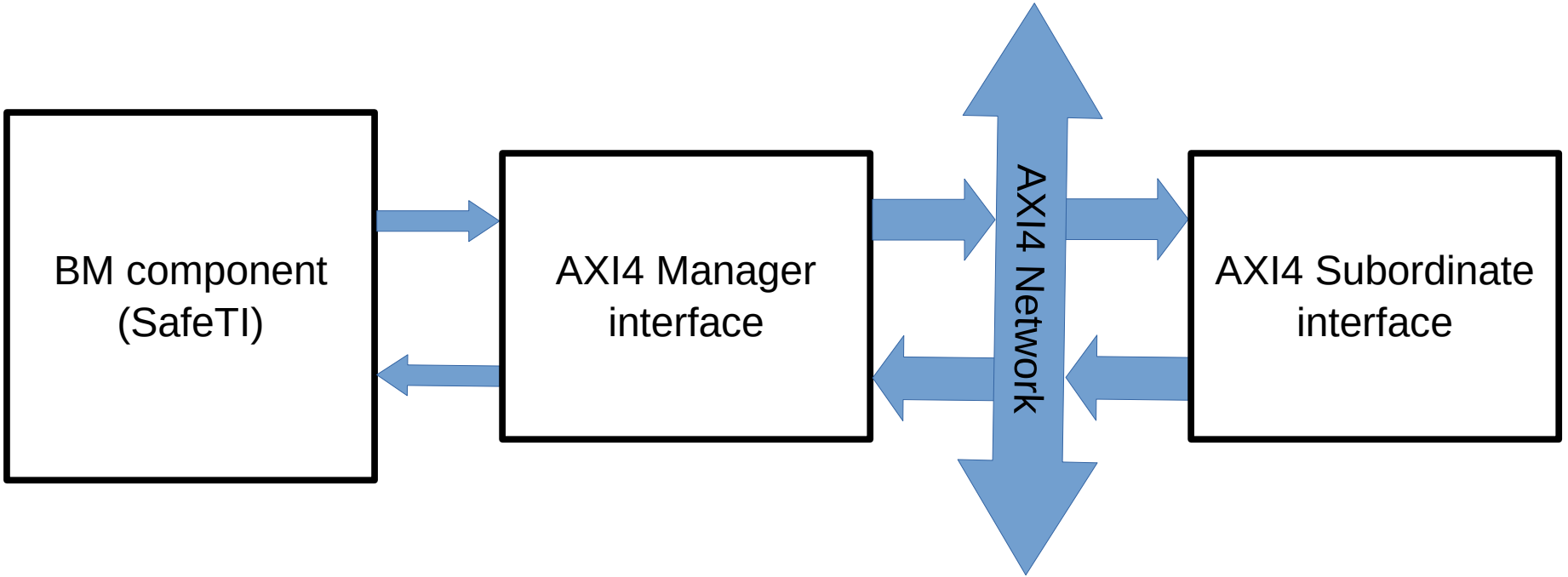
AXI4 Manager Interface



AXI4 Protocol features

- Every transaction is executed through a **burst**, where many **beats** transfers are executed until the completion of the burst.
- The subordinates allocate 4kB of memory (4096 address spaces). This sets the **4kB boundary rule** where any burst must not surpass the 4kB address space, since every burst is specific to a subordinate.
- The protocol features **5 channels** (group of busses); AW, AR, W, R and B.
- The read and write data busses are extensible from 8 upto 1024 bits in power of two's steps. **The width of the data bus fixes what addresses each byte lane can access**. For instance, 64 and 128 bus width:

Addresses ended in 0xXX &	7/F	6/E	5/D	4/C	3/B	2/A	1/9	0/8

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

AXI4 Bus use

Handshake

Use AX* channel:

- M -> S
  - > **Burst** mode.
  - > Starting **addr**ess.
  - > **Size** mode.
  - > Burst **len**gth.
  - > **Valid** control.
  - - **ID**.
  - - Memory **region**.
  - - **Lock**.
  - - **Cache**.
  - - **Prot**ected.
  - - **QOL**.
- S -> M
  - > Control **ready**.

Beats

Read transactions:

- M -> S, R channel:
  - > **Ready** data.
- S -> M, R signals:
  - > Read **data**.
  - > **Valid** data.
  - > **Last** beat.

Write transactions:

- M -> S, W channel:
  - > Write **data**.
  - > **Valid** data.
  - > **Last** beat.
  - - Write **strobe**.
- S -> W, W channel:
  - > **Ready** data.

Response

Read transactions:

- S -> M, R channel:
  - > Burst **resp**onse.

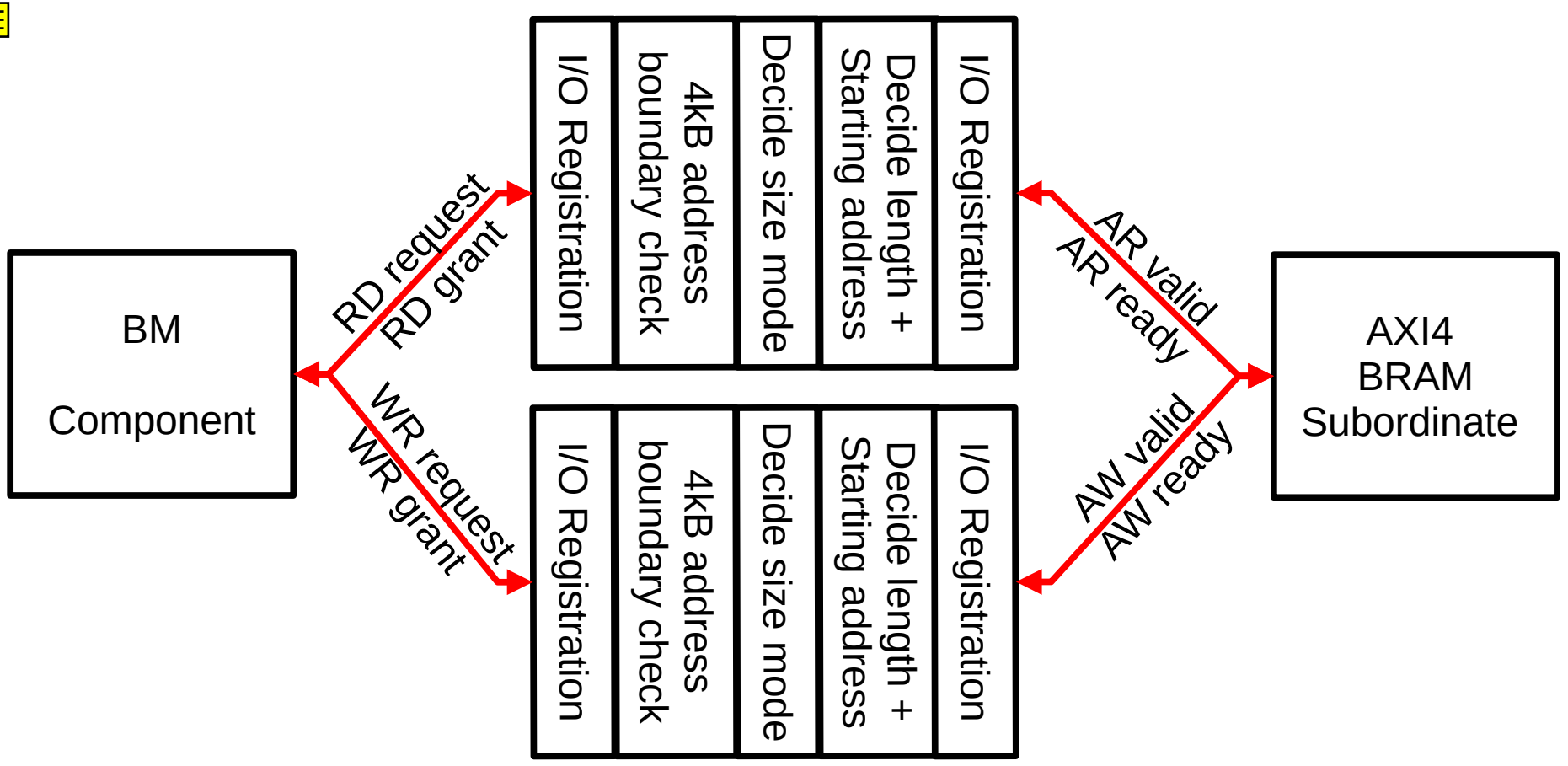
Write transactions:

- S -> M, B channel:
  - > Beat **resp**onse.

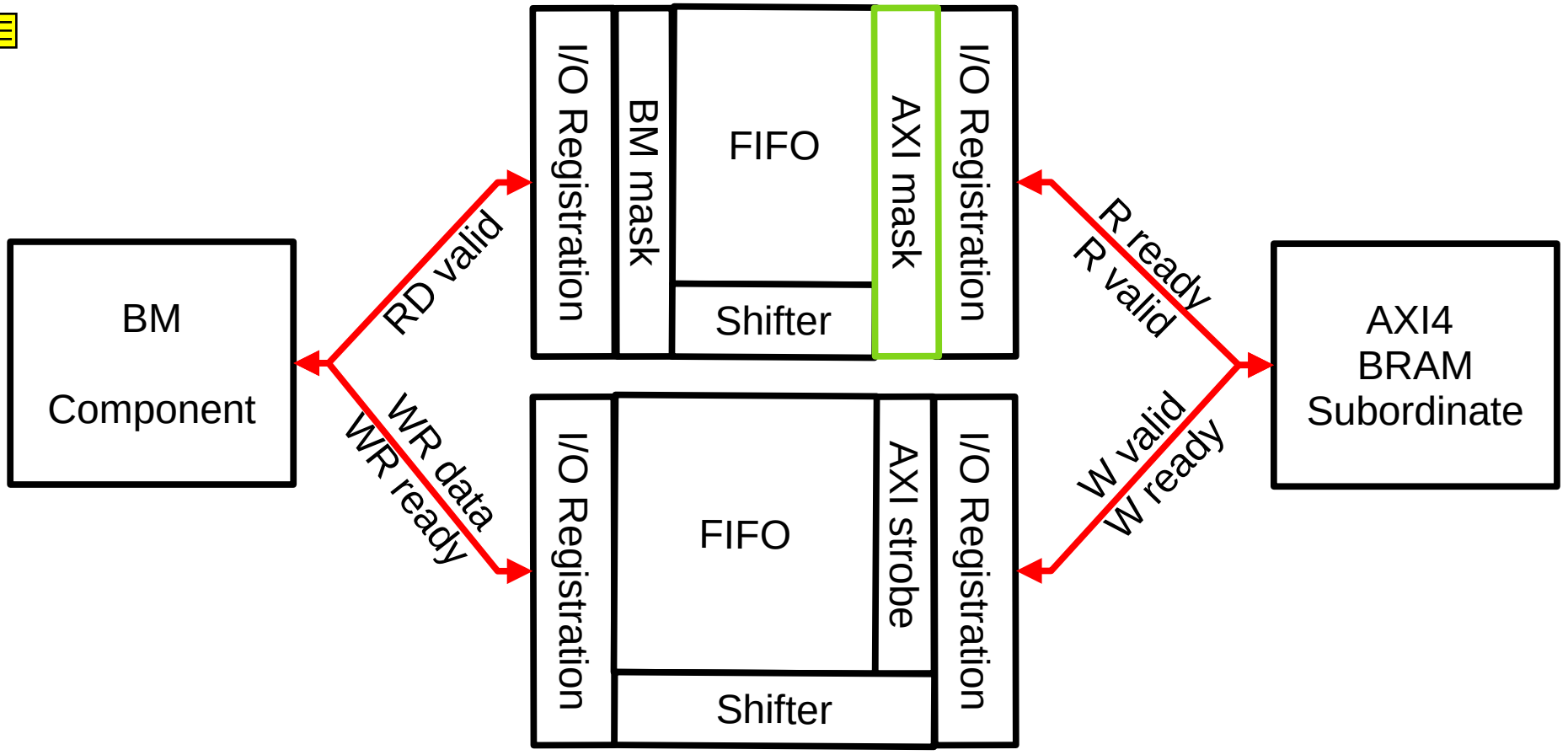
S: Subordinate.
M: Manager.

*: X is W for write and R for read transactions.

AXI4 Manager interface structure for Handshake



AXI4 Manager interface structure for data transfer



AXI4 Transaction example

Let’s suppose a BM data bus width of 32 bits (dbits) and an AXI data bus width of 128 bits (AXI4_DATA_WIDTH).

BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

AXI4 Transaction example

BM read transaction request:
Starting address = 0x0FFB
Total size transfer = 23 bytes
(codification) = 0x16

Let’s suppose a BM data bus width of 32 bits (dbits) and an AXI data bus width of 128 bits (AXI4_DATA_WIDTH).

0x0FFB + 23 data bytes – 1 = 0x1011 <- Last read address to BM transfer.
^----------------------------------^- Acces to two different subordinates is required.

AXI4 Transaction example

BM read transaction request:
Starting address = 0x0FFB
Total size transfer = 23 bytes
(codification) = 0x16

Let’s suppose a BM data bus width of 32 bits (dbits) and an AXI data bus width of 128 bits (AXI4_DATA_WIDTH).

0x0FFB + 23 data bytes – 1 = 0x1011 <- Last read address to BM transfer.
^----------^- Acces to two different subordinates is required.

First burst:

0x0FFF – 0x0FFB = 5 data bytes -> Size mode is 8 bytes/beat.

Only one beat is required with a starting address of 0x0FF0.

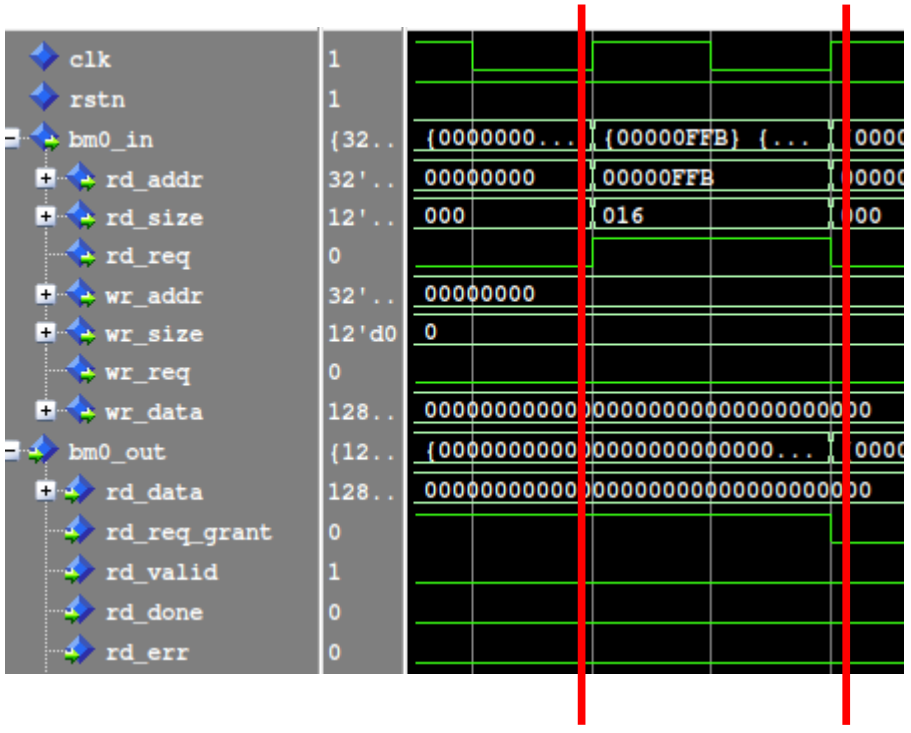
Second burst:

23 data bytes – 5 data bytes = 18 data bytes -> Size mode is 16 bytes/beat.

(Size mode limited by AXI4_DATA_WIDTH)

Two beats are required with a starting address of 0x1000.

AXI4 Transaction example



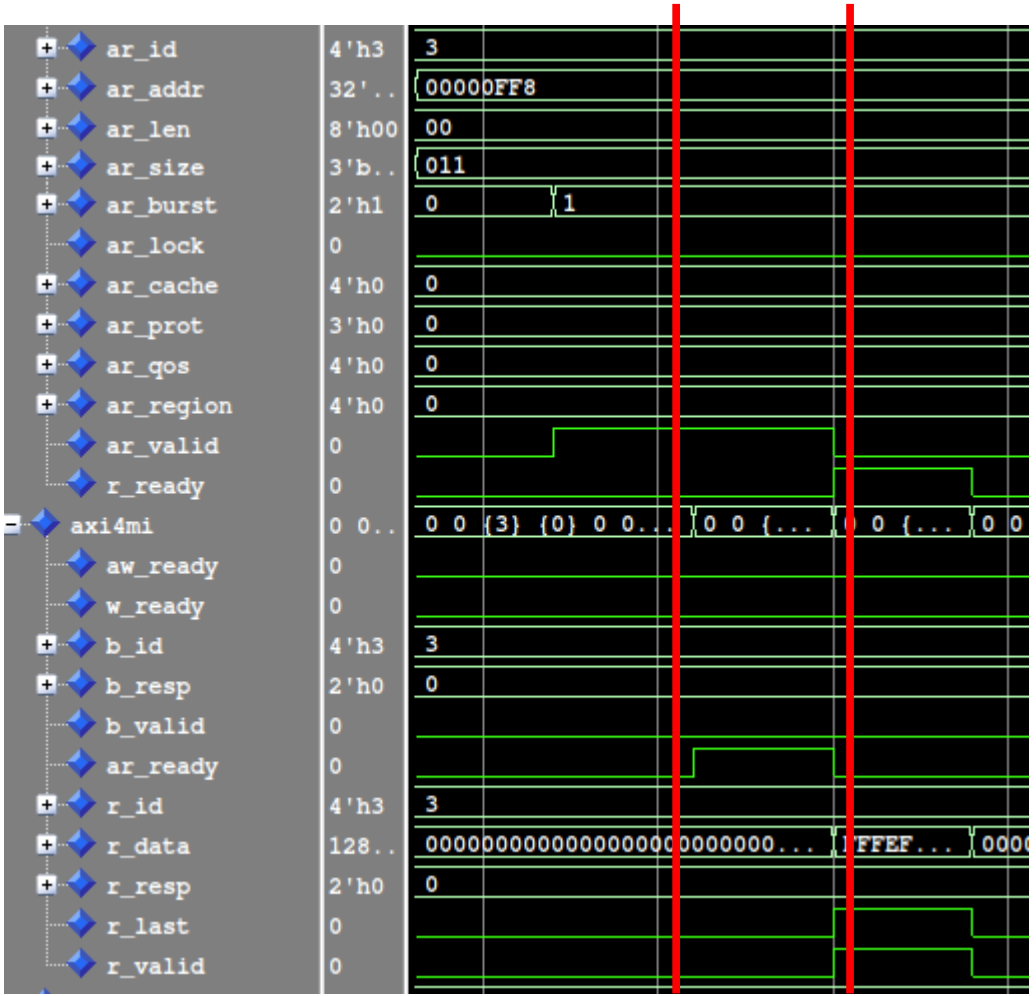
BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01

AXI4 Transaction example



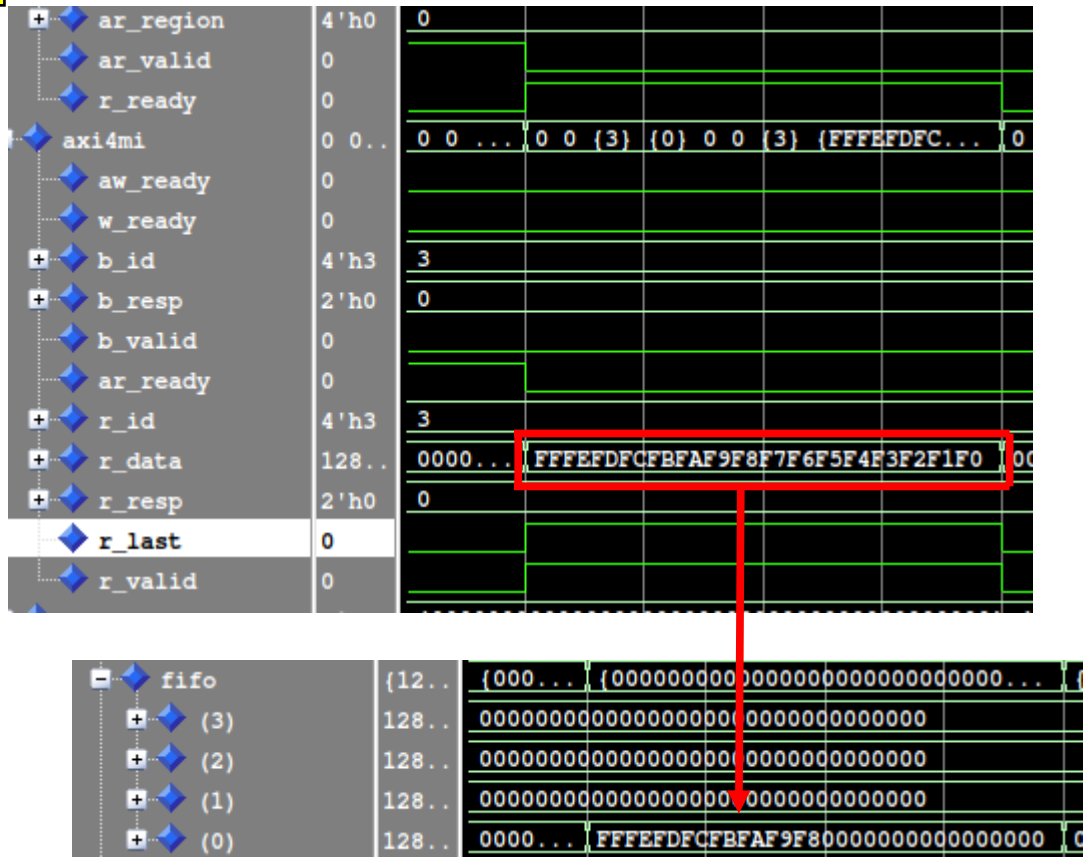
BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01

AXI4 Transaction example



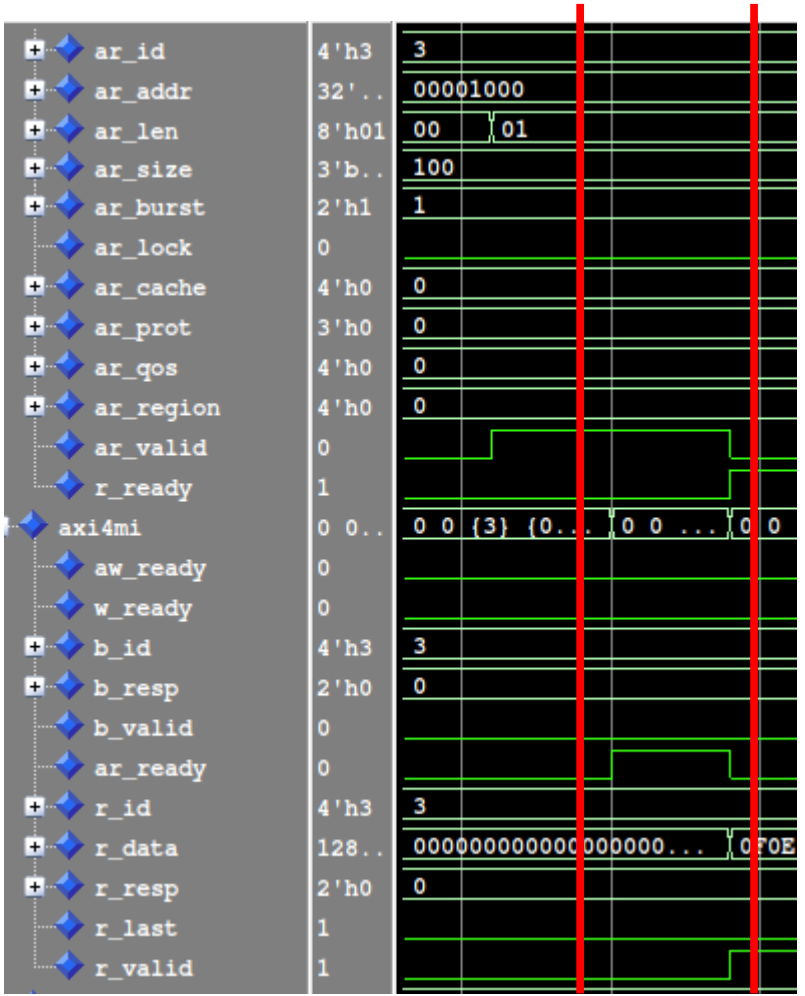
BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01

AXI4 Transaction example



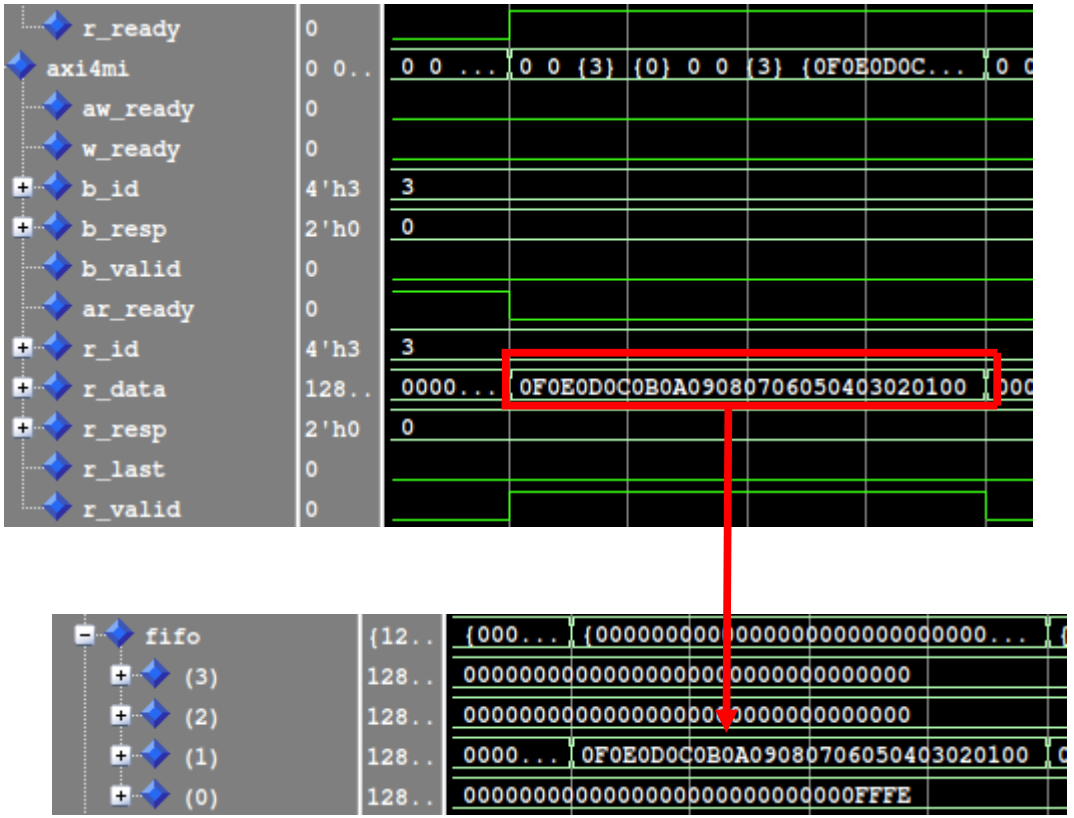
BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01

AXI4 Transaction example



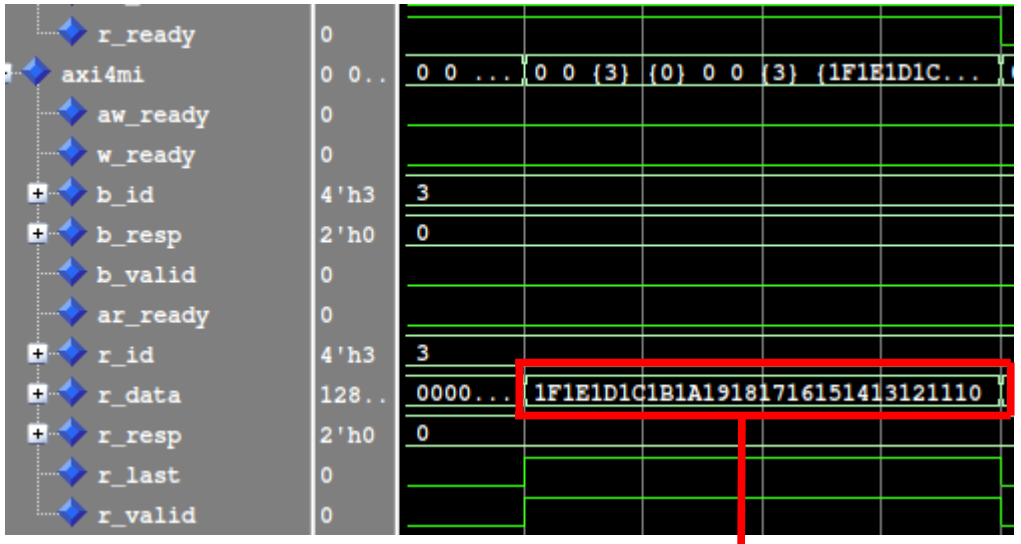
BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01

AXI4 Transaction example

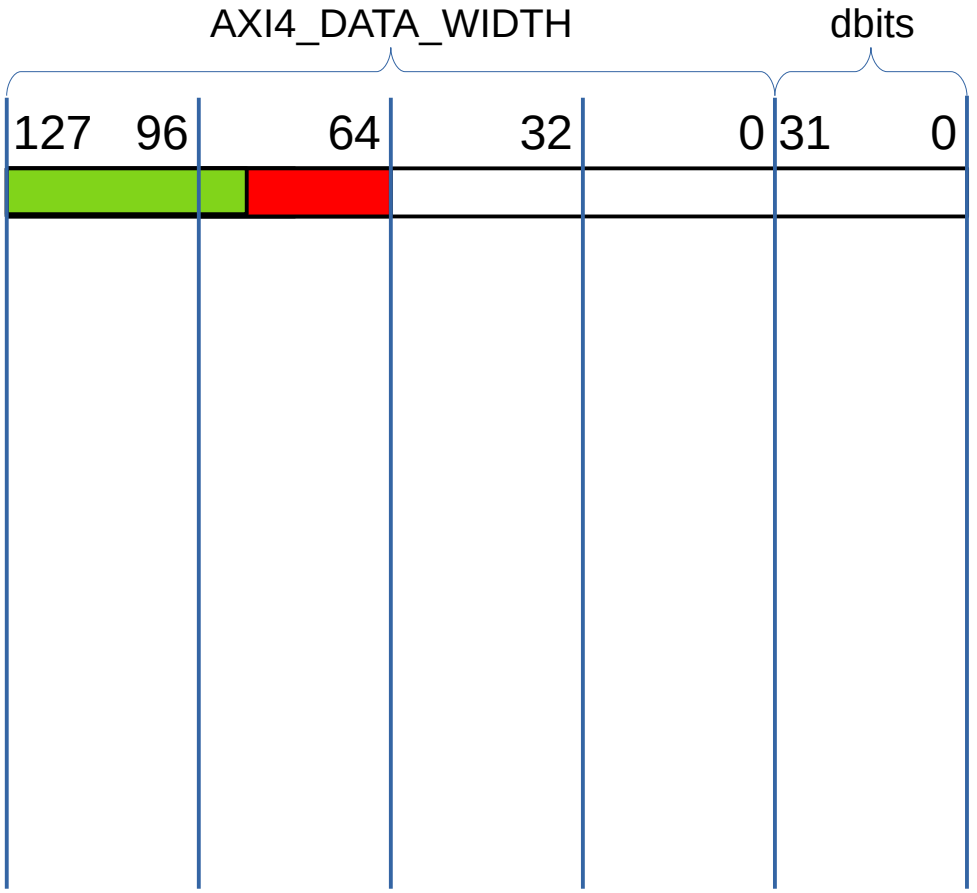


BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01



BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

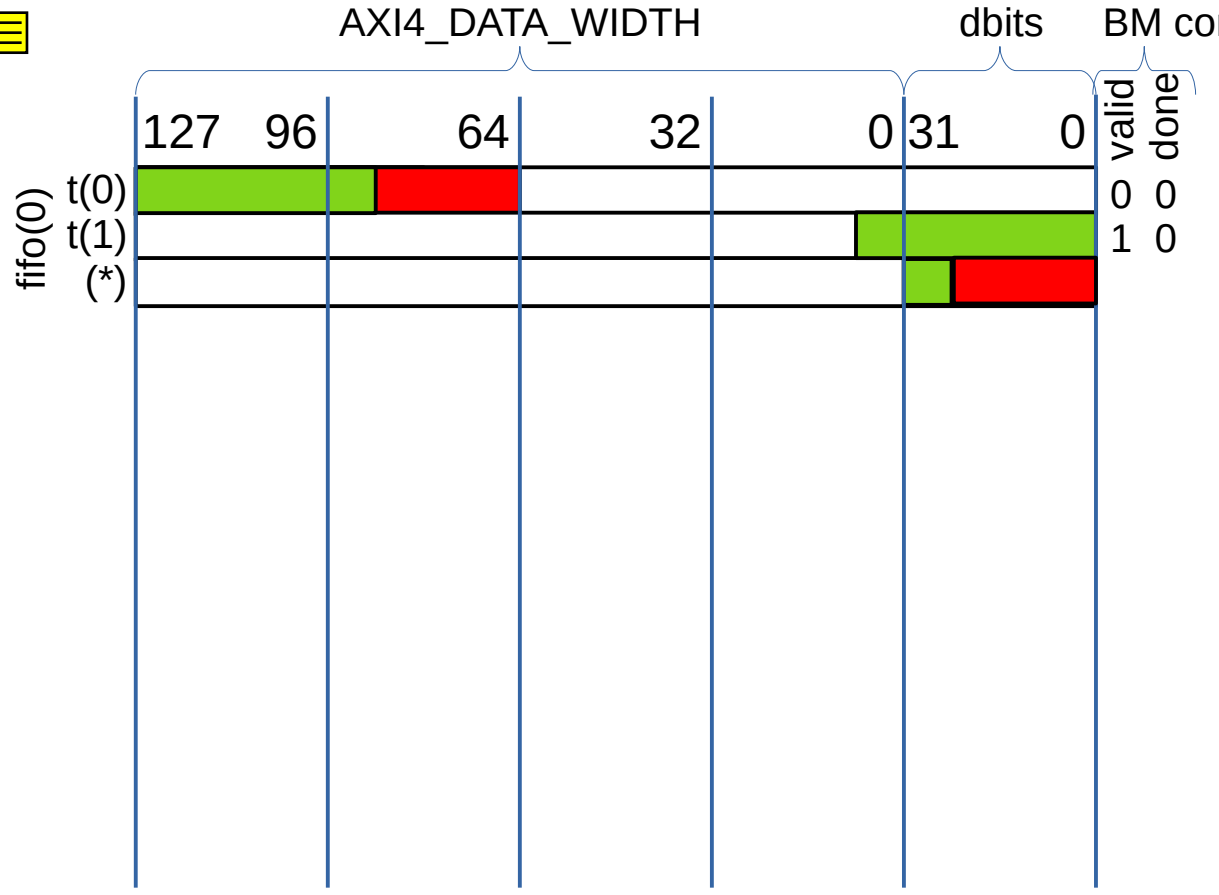
Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF0
- Size mode = 8 bytes = 011’b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100’b
- Burst length = 2 beats = 0x01

: AXI read data, but it’s unrequested by BM. : BM requested data from first, second and third beats.

(*): Skipped time slots. Represented to better conceptualize the state of the different FIFO regs with the dbits slot.



BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

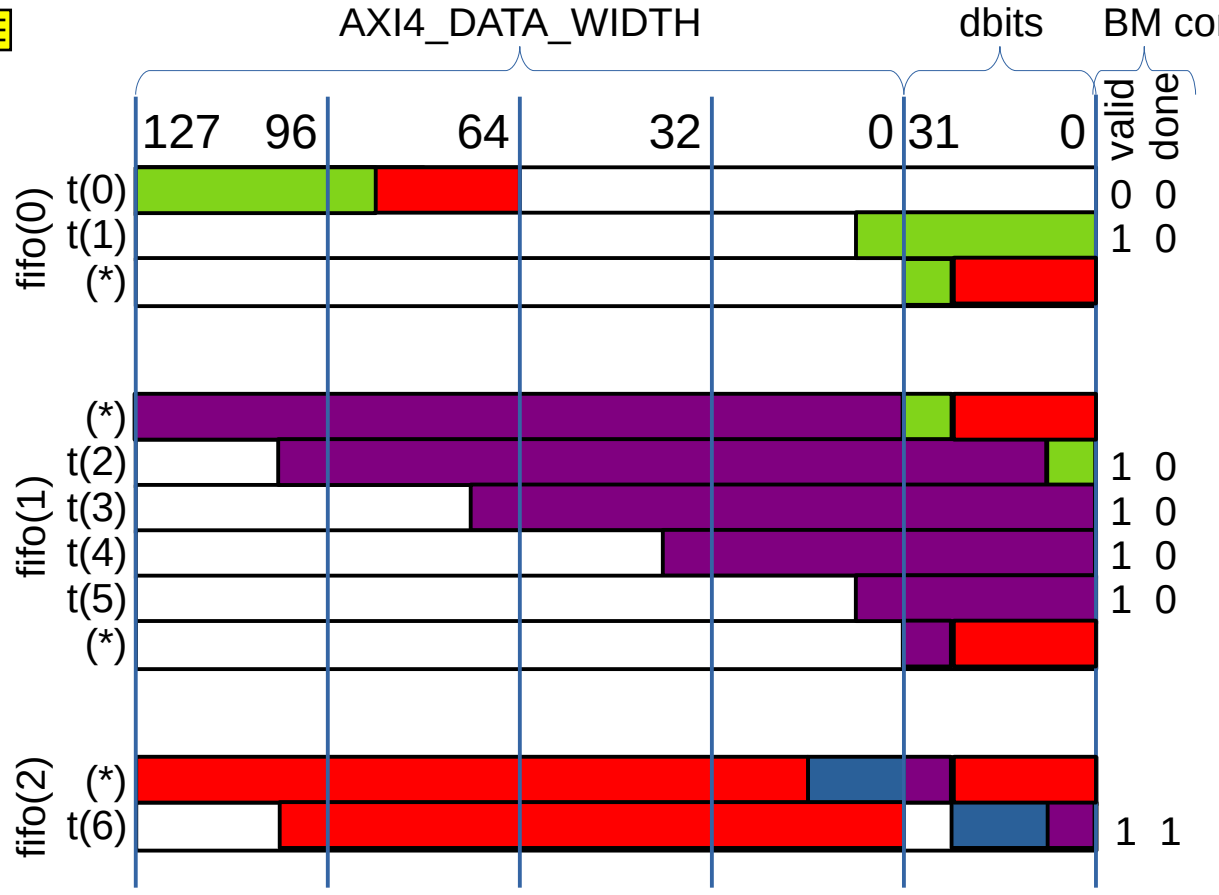
Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011'b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100'b
- Burst length = 2 beats = 0x01

: AXI read data, but it's unrequested by BM. : BM requested data from first, second and third beats.

(*): Skipped time slots. Represented to better conceptualize the state of the different FIFO regs with the dbits slot.



BM read transaction request:
- Starting address = 0x0FFB
- Total size transfer = 23 bytes
- (codification) = 0x16

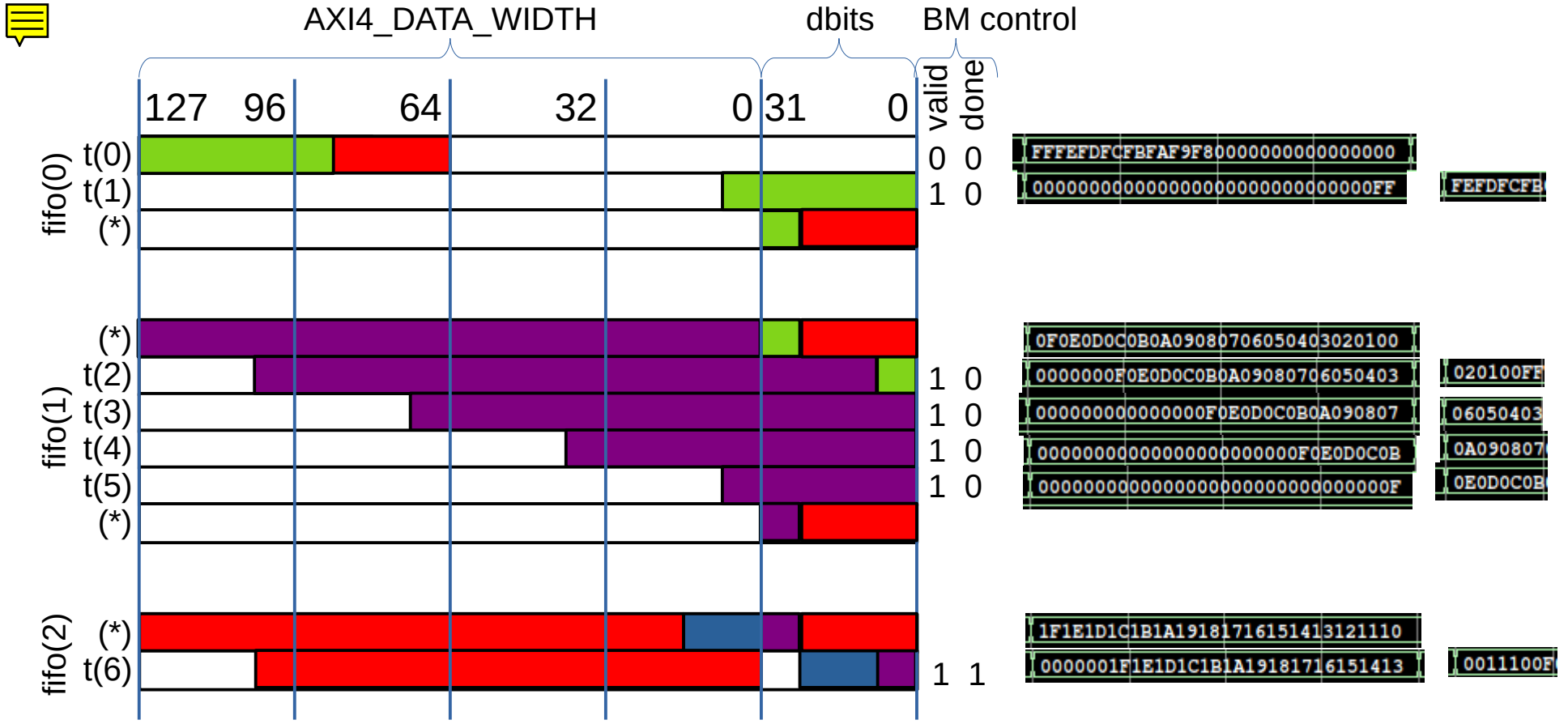
Since the 4kB boundary space is expected to be surpassed by the request, two AXI bursts are generated.

First AXI burst:
- Starting address = 0x0FF8
- Size mode = 8 bytes = 011'b
- Burst length = 1 beat = 0x00

Second AXI burst:
- Starting address = 0x1000
- Size mode = 16 bytes = 100'b
- Burst length = 2 beats = 0x01

: AXI read data, but it's unrequested by BM. : BM requested data from first, second and third beats.

(*): Skipped time slots. Represented to better conceptualize the state of the different FIFO regs with the dbits slot.



: AXI read data, but it's unrequested by BM. : BM requested data from first, second and third beats.

(*): Skipped time slots. Represented to better conceptualize the state of the different FIFO regs with the dbits slot.

AXI4 Narrow transfer example

Address: 0x07
Transfer size: 32-bits
Burst type: incrementing
Burst length: 5 transfers

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0	
0x07	0x06	0x05	0x04	0x03	0x02	0x01	0x00	1st transfer
0x0F	0x0E	0x0D	0x0C	0x0B	0x0A	0x09	0x08	2nd transfer
0x0F	0x0E	0x0D	0x0C	0x0B	0x0A	0x09	0x08	3rd transfer
0x17	0x16	0x15	0x14	0x13	0x12	0x11	0x10	4th transfer
0x17	0x16	0x15	0x14	0x13	0x12	0x11	0x10	5th transfer

WDATA[63:0]

Source: “Figure A3-14 Aligned and unaligned transfers on a 64-bit bus”, page A3-58, from ARM IHI 0022H.c "AMBA AXI and ACE Protocol Specification”.

AXI4 Narrow transfer example on faulty subordinate

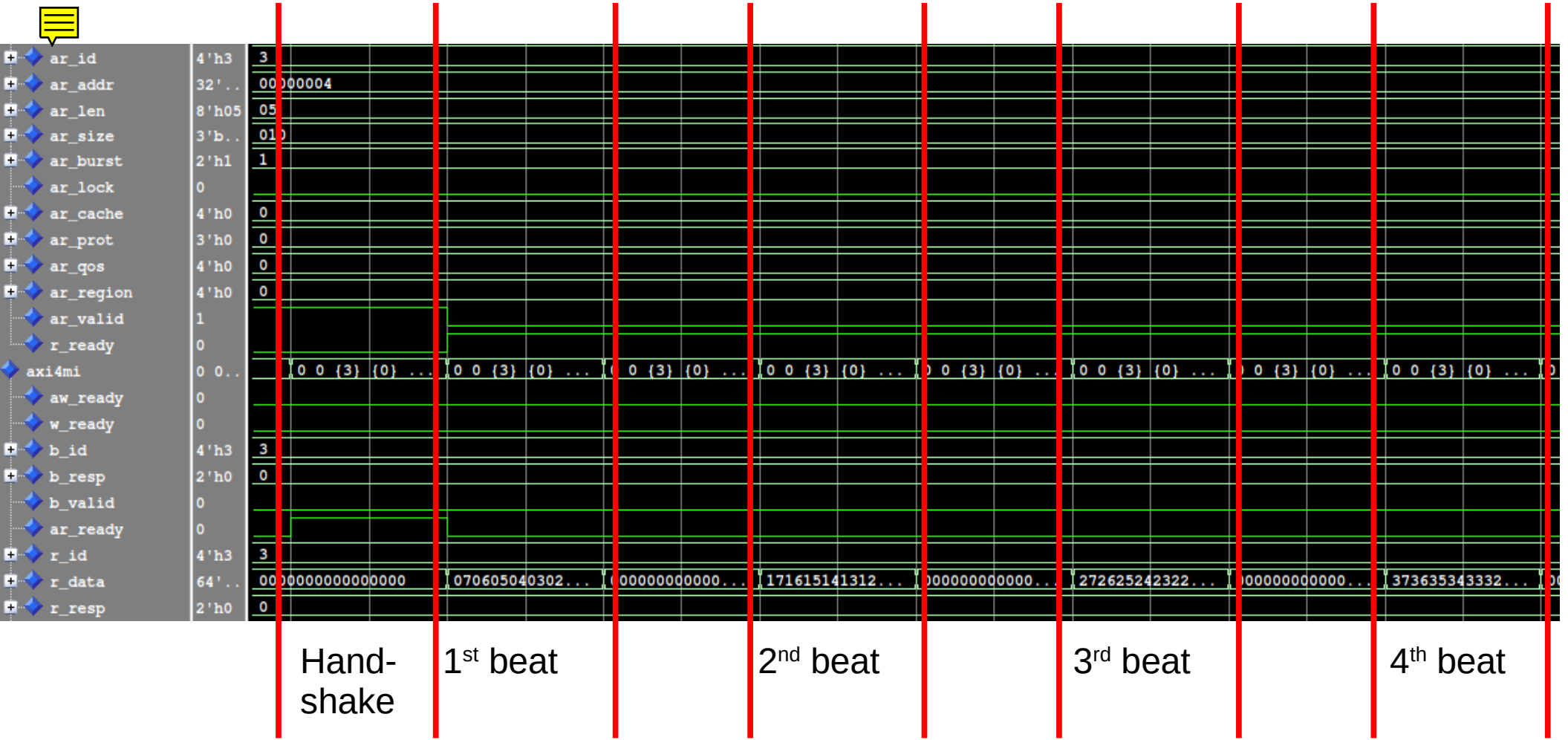


Table 4-1: **Xilinx AXI4 and AXI4-Lite Feature Adoption and Support**

AXI Feature	Xilinx IP Support
READY/VALID Handshake	Full forward and reverse direction flow control of AXI protocol-defined READY/VALID handshake.
Transfer Length	AXI4 memory-mapped burst lengths of: · 1 to 256 beats for incrementing bursts · 1 to 16 beats for wrap bursts Fixed bursts should not be used with Xilinx IP. Conversions of FIXED bursts through AXI Interconnect infrastructure could have sub-optimal performance.
Transfer Size / Data Width	IP can be defined with native data widths of 32, 64, 128, 256, 512, and 1024 bits wide. For AXI4-Lite, the supported data width is 32 or 64 bits only, but 32-bits is recommended to minimize resource utilization. The use of AXI4 narrow bursts is supported but is not recommended. Use of narrow bursts can decrease system performance and increase system size. Where Xilinx IP of different widths need to communicate with each other, the AXI Interconnect provides data width conversion features.
Read/Write only	The use of read/write, read-only, or write-only interfaces. Many IP, including the AXI Interconnect, perform logic optimizations when an interface is configured to be Read-only or Write-only.

Features

- AXI4 (memory mapped) slave interface
- Low latency memory controller
- Separate read and write channel interfaces to utilize dual port FPGA BRAM technology
- Option to arbitrate read and write data for use with a single port of block RAM (in AXI4 and AXI4LITE modes)
- Configurable BRAM data width (32-, 64-, 128-, 256-, 512-, and 1024-bit) (equals AXI slave port data width size)
- Supports memory sizes up to a maximum of 2 MBytes (byte size 8 or 9)
- Supports INCR burst sizes up to 256 data transfers, and WRAP bursts of 2, 4, 8, and 16 data beats
- Supports AXI narrow and unaligned write burst transfers
- Compatible with Xilinx AXI Interconnect
- Supports two-deep address pipelining on each read and write channel (order must be maintained)
- Reduced footprint option for AXI4-Lite
- Supports of both Internal and External modes of BRAM block instances
- Optional ECC support with 32-, 64- or 128-bit BRAM data widths
    - AXI4-Lite register interface for status and control of ECC operations

AXI Block RAM (BRAM) Controller v4.0

LogiCORE IP Product Guide

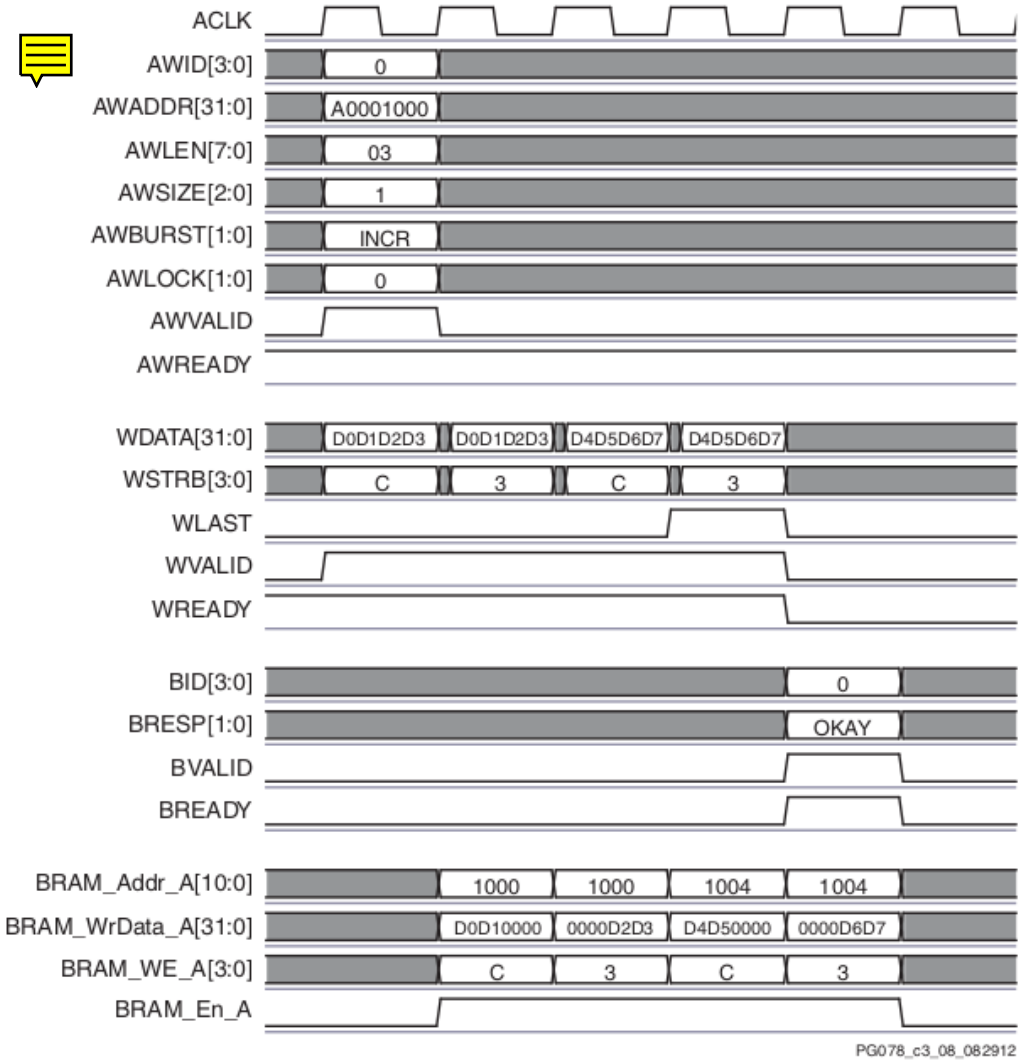


Figure 3-8: **AXI Narrow Burst Write Diagram**

AXI Block RAM (BRAM) Controller v4.0

LogiCORE IP Product Guide

Conclusions

- Started the design of the interface executing narrow bursts always. But, due to the characteristics of the subordinate used for debugging, the interface has been modified to only execute narrow bursts if they occur in different AXI4_DATA_WIDTH addresses (more efficient).
- The modifications have made the interface to be incapable of managing multiple narrow bursts that occur on the same AXI4_DATA_WIDTH slot. However, it also has given a step to simplify the interface logic, that it could be simplified even further. This would be by setting an AXI size mode adjusted to AXI4_DATA_WIDTH, saving resources and even a clock cycle.
- At the moment, the AXI4 Manager interface only executes incremental bursts.