Delay Memory Module (DMM)

Introduction

The functions of the DMM are to receive channel data from the CRM, transmit channel data in correlator frames as set by the DCM, to delay or advance data on a per channel basis, to insert correlator frame header data on the validity signal of each channel. Operation is described with reference to the block diagram.

Connectors

There are two connectors which interface to the back plane of the station unit rack. They are designated PL1 and PL2. PL1 is a standard 96 pin VME bus connector but only the +5V and GND pins are used. PL2 is a 185 pin connector which carries all other relevant station unit signals. There is a 48 pin connector on the front panel which carries some test signals. The pin connections can be found on the circuit diagrams.

Re-Time

Clocks and control signals are routed from the connector to buffers. Channel clock and station clock can be selected as normal or inverted by jumper links. Both have been selected as normal.

Channel clock is delayed by 12.5 ns and used to clock eight channels of data input from the CRM into buffers. Channel clock is delayed by 22.5 ns and used as a receive clock to clock data into the gate arrays. These delays have been hard wired.

Station clock is delayed by 10 ns and used as a transmit clock to clock eight channels of data out of the gate arrays, eight channels of data into the output buffers, and to generate the FIFO memory read signal. This delay has been selected by jumper link.

DMM FPGA 0 to 7

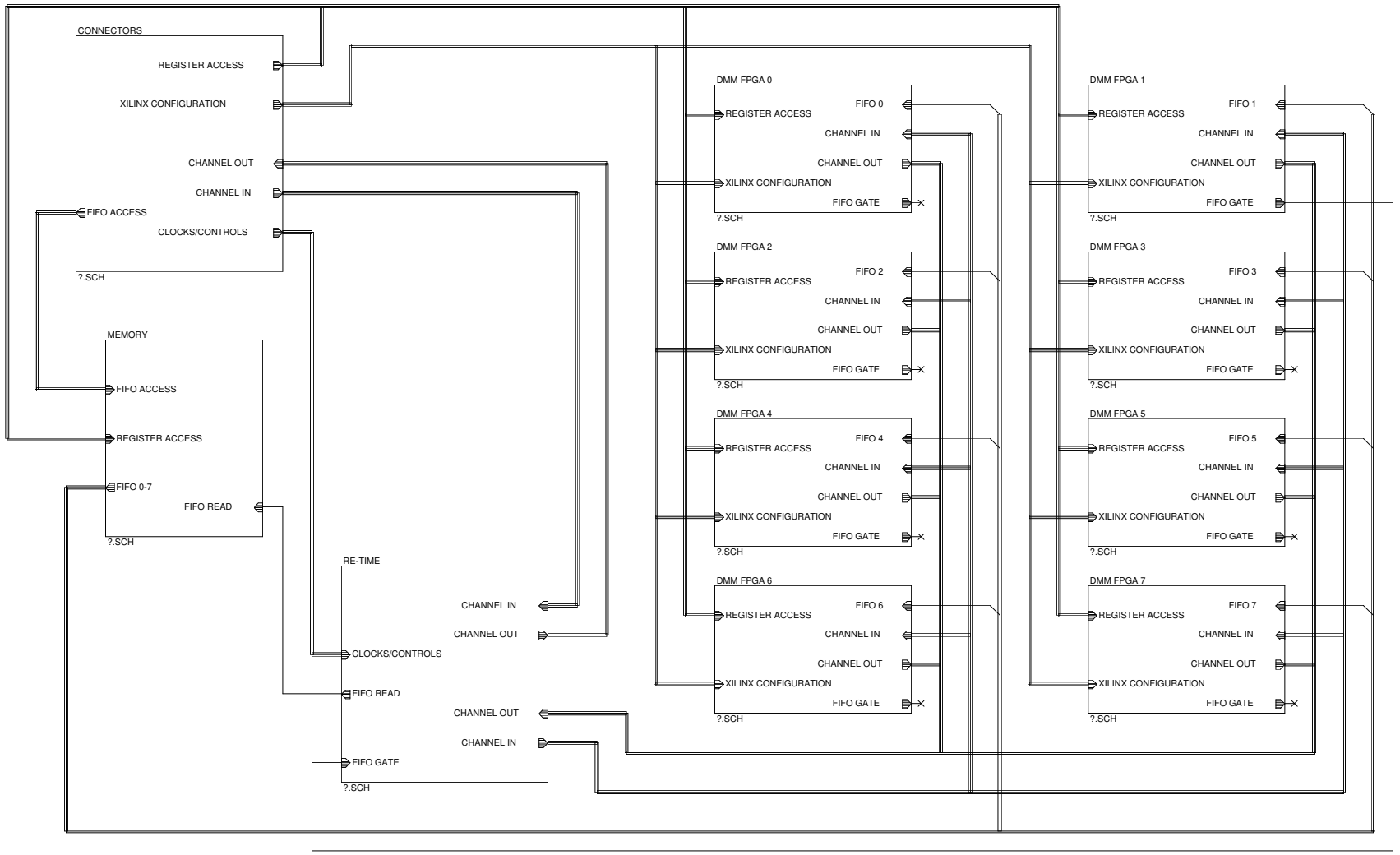
All these blocks are the same and consist of a XILINX gate array and three Dynamic RAMs (DRAMs). There is one gate array per channel and one DRAM per channel signal. All the DRAMs of a channel have common address lines so that the three signals of a channel are processed identically. Each gate array is accessed individually for configuration and register read/write operations. Clocks and control signals to the gate arrays are not shown in the block diagram.

Channel In data is received serially by the gate array and written as parallel eight bit bytes to the DRAMs. All control, address, and data lines to the DRAMs originate from the gate array. Channel Out data is read as parallel eight bit bytes from the DRAMs and transmitted serially. Reception of the SOT signal is a marker for the received data which enables the DCM microcontroller to read, from gate array registers, the DRAM address where that data has been written. Reception of the BOCF_DM signal synchronises the start of transmission of data for the next correlator frame and the insertion of correlator frame header data. The DRAM address for transmission is written to gate array registers by the DCM microcontroller.

Memory

This block consists of read only registers, a FIFO memory, and access control logic. The registers hold board identification data. The logic uses buffers and a PAL device to control access. When accessing the FIFO the DCM microcontroller resets it and then writes all the bytes of header data to it. This is performed as a whole operation and is the only access that the DCM microcontroller has to the FIFO. The FIFO read signal is generated from the transmit clock and a version of the BOCF signal received by the DMM.

After a reset the FIFO memory has a latency of clock cycle on a read access (see the data sheet for the device). Because the FIFO is always reset for each write/read cycle this latency always occurs during the read part of the cycle. This means that the first byte of data is not read out until the second read clock. Therefore to present all of the header data at the correct time a delayed and stretched version of the BOCF signal is required to gate the transmit clock. This FIFO gate signal is generated in the XILINX gate arrays which means that it is available from eight sources. The signal which is used is routed from DMM FPGA 1.



DELAY MEMORY MODULE

BLOCK DIAGRAM

DMM Gate Array 1660010016

This description should be read in conjunction with the circuit diagrams for the gate array. Signal names which are used are those found in the diagrams. General principles of operation are described and no detail of implementation to gate level is given.

There are three clock inputs to the FPGA which are all asynchronous to each other; receive clock, transmit clock , and 32 MHz general clock. Not all sections of the circuit need to run at 32 MHz so this is divided to 16 MHz for use in these sections.

The FPGA is set-up and controlled by the DCM microcontroller over the station unit back plane. This communicates via sixteen bi-directional data lines and decoded address and read/write control signals. The decoded signals select registers for writing and multtiplexers for reading(sheets 5,15).

SIGN_IN, MAG_IN, VAL_IN, and CFRAME are clocked in by RX_BIT_CLK (sheet 3). At this point data still has bit positions from which the validity data has been extracted. CFRAME is used to gate an eight bit serial to parallel shift register so that an eight bit parallel data byte is generated from the serial data. The parallel data is transferred to the first stage of a receive FIFO parallel buffer under the control of RX_BYTE_CLK. Waveform diagrams for multiplexed, non multiplexed, and de-multiplexed data are shown in Figures 1, 2, 3, and 4.

The receive FIFO buffer is three bytes wide so that sign, magnitude, and validity are processed identically. There are a further six stages of the receive FIFO buffer which are clocked at 32 MHz. In order to ensure correct transfer from the first stage, the control signal DAT_AV is synchronised from RX_BYTE_CLK. DAT_AV only occurs once for each occurrence of RX_BYTE_CLK and is always one clock period wide at 32 MHz (sheet 2). Waveform diagrams for the extremes of difference between RX_BIT_CLK and M32_CLK are shown in Figures 5 and 6.

Bytes of data are shuffled along the FIFO to the other end or until they catch up with previous data. Previous data is never over-written. When a byte is extracted at the last stage all bytes in the FIFO move along one position (sheet 4). When four bytes are present at the end of the FIFO, RX_RAM_REQ is generated. This will be accepted by the dynamic ram (DRAM) access circuits (sheets 6,7,8,9,10) and four bytes will be written to the DRAM's (sign, magnitude, validity).

There are three types of access to the DRAM's; four byte write, four byte read, and refresh. A write access is initiated by RX_RAM_REQ, a read access is initiated by TX_RAM_REQ. These requests for access are prioritised with write access having the higher priority. If neither a write access nor a read access is requested DRAM refreshes are performed automatically. A request for write or read access will not always be accepted immediately so the request is latched and the latched request is only cleared when it has been accepted. The number of clock cycles required for a DRAM access varies with the type of access. Waveform diagrams for the three types of access are shown in Figures 7,8,9. Write and read accesses are always four byte accesses. The refresh cycle is a CAS before RAS cycle (see the DRAM data sheet for further details). Refresh does not require an address.

The two low order address lines for the DRAM are generated within the access control circuits. The remainder of the address is generated by an accumulator. These higher order write and read addresses are stored in separate registers but use a common adder. They are incremented automatically for each write or read. They are not initially constrained to start at any particular address except from first configuration of the FPGA they will be zero because of the way the devices start up.

The write address will increment until the top of the memory is reached and then wrap round to zero. When an SOT_IN signal is received the current write address is latched into further registers which can be read by the DCM microcontroller.

The read address will increment and wrap round in a similar way to the write address. However this sequence will be changed every time a BOCF_DM signal is received. An address from which the DRAM should be read next can be written to registers by the micrcontroller. On reception of the next BOCF_DM signal this address will be used for a read access and incremented from there in the normal way.

When a DRAM read access is initiated, four bytes of data loaded into a transmit FIFO buffer (sheets 11,12). There will always be four vacant positions in the FIFO because this condition generates the TX_RAM_REQ signal. The transmit FIFO buffer is three bytes wide so that sign, magnitude, and validity are processed identically. Bytes of data are shuffled along the FIFO to the other end or until they catch up with previous data. Previous data is never over-written. Data at the end of the FIFO are available to a parallel to serial shift register which is where the serial transmission takes place (sheet 13). TX_PIPE_CLK enables the transfer of data from the last buffer to the shift register and this occurs every eight bits transmitted.

This transfer also causes GET_NEXT to shift data down the FIFO by one position so that the next byte is ready. To ensure that this occurs correctly, GET_NEXT is synchronised by one of TX_PIPE_1, TX_PIPE_2 or TX_PIPE_3. GET_NEXT only occurs once per synchronising event and is only one clock period wide at 32 MHz. Waveform diagrams for the extremes of difference between TX_BIT_CLK and M32_CLK are shown in Figures 10 and 11.

During the BOCF header period sign and magnitude data are not valid and BOCF header data is inserted into the validity signal. This data is retrieved from a FIFO memory device external to the FPGA. BOCF_DM is delayed so that the insertion of this header data is synchronised with the BOCF signal which has been transmitted from DCM to the SUIM. In order to achieve this with the speed and fitting constraints of the FPGA this has been achieved as follows:-
input from BOCF_DM
delayed to BOCF
delayed to DDBOCF
output to BOCF_FB
input from BOCF_FIFO
delayed to DBOCF_FIFO
This has required an external link from BOCF_FB to BOCF_FIFO. The header data from the external memory also has to be synchronised with DBOCF_FIFO. This is achieved by the output signal B_F_OUT which is a delayed, stretched and phase shifted version of BOCF_DM which enables data BOCF_H_IN to be input at the correct time.

The DRAM read address generation permits data to be transmitted from any four byte boundary. The transmit control circuits permit transmission to start at any bit within that four bytes. The number of the bit in the four bytes (in the range 0 to 31), BIT_CNT, is pre-set into a five bit binary down counter. At the end of a BOCF frame signalled by the rising edge of BOCF_DM, generation of TX_RAM_REQ is inhibited and the transmit FIFO buffer is allowed to completely empty. The generation of TX_RAM_REQ is then re-enabled and transmission is resumed while counting down from BIT_CNT. When the terminal count is reached transmission is inhibited until the end the BOCF header period with the selected bit lined up ready for transmission.

The transmission of data can be advanced or delayed by one bit position at a time. When this occurs is signalled by SLIP_BIT which is one period of TX_BIT_CLK wide. Whether data is advanced or delayed is determined by SKIP_BIT. This is set by the DCM microcontroller writing to a register.

When SKIP_BIT is set low, the occurrence of SLIP_BIT inhibits clocking of the parallel to serial shift registers and the counter which generates TX_PIPE_CLK. This causes the bit at this time to be transmitted again introducing the delay. It is also marked as invalid by forcing the validity line low for this bit period. This event can occur at any point in a byte transfer. Waveform diagrams are shown in Figure 12.

When SKIP_BIT is set high, the occurrence of SLIP_BIT latches the validity line low. At the end of the current cycle of the counter which generates TX_PIPE_CLK it is set to count seven instead of eight for one cycle. This cycle skips one bit introducing the advance. At the end of this cycle the validity line is unlatched. This event can occur at any point in a byte transfer. Depending on where in the byte the event occurs the validity line will be forced low for between eight and fifteen bit positions. Waveform diagrams are shown in Figure 13.

The generation of SLIP_BIT is performed in the delay accumulator (sheet 14). The accumulator is thirty two bits wide. In order to achieve operation at 32 MHz this is split into two sections of eighteen bits and fourteen bits. The carry out of the eighteen bit section is pipelined to the carry in of the fourteen bit section. The carry out of the fourteen bit section is pipelined to become SLIP_BIT. An 18:14 split was chosen to match the way that the accumulator is used. It will be primed with a thirty two bit initial value and when running an eighteen bit value will be added in on every TX_BIT_CLK.

At the end of transmission of the current correlator frame, the initial value is loaded into the accumulator. The add-in value is the presented to the accumulator inputs ready for when transmission of the next correlator frame starts. This is achieved by pipeline registers. The DCM microcontroller first writes the initial value as two 16 bit words to two registers. It the writes the add-in value to the same registers ensuring that the high order fourteen bits are zero. This second write shuffles the data first written down the pipeline. The initialisation of the accumulator starts when TX-SMC3 goes low. There are circuits to synchronise this signal to FPGA_CLK (16 MHz) and generate pipeline control signals which are then resynchronised to TX_BIT_CLK. The data is extracted from the end of the pipeline, as determined by these control signals, which is then available to receive new values. Waveform diagrams are shown in Figure 14.

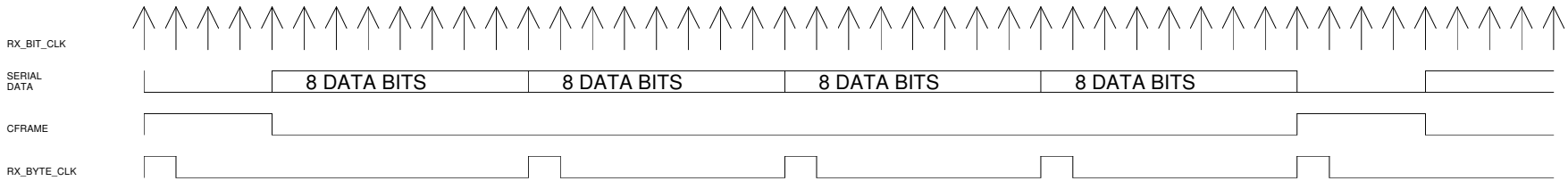


Figure 1. 9 BIT to 8 BIT TRANSFER 4:1 MULTIPLEXED (pipeline delays not shown)

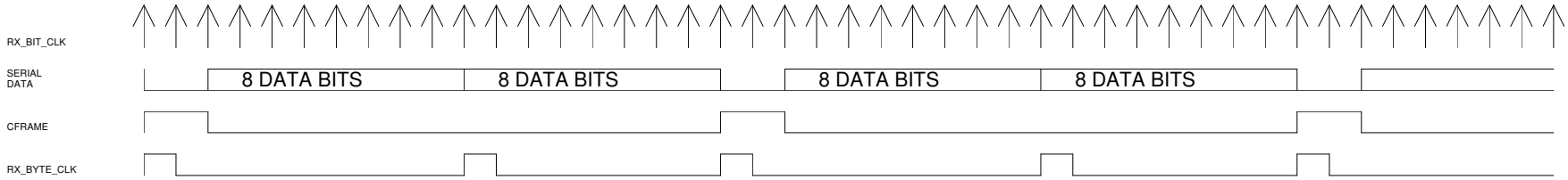


Figure 2. 9 BIT to 8 BIT TRANSFER 2:1 MULTIPLEXED (pipeline delays not shown)

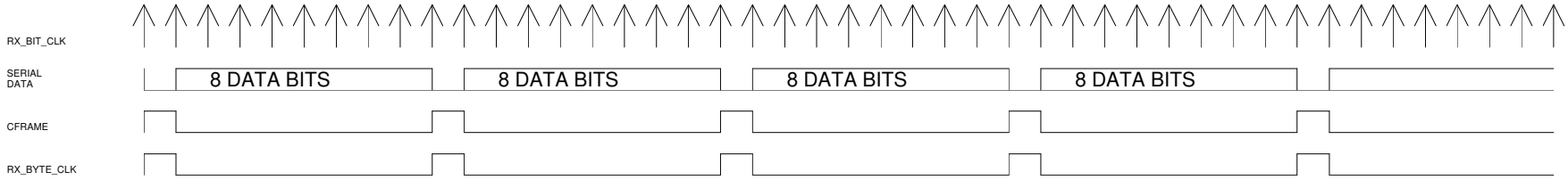


Figure 3. 9 BIT to 8 BIT TRANSFER 1:1 NON MULTIPLEXED (pipeline delays not shown)

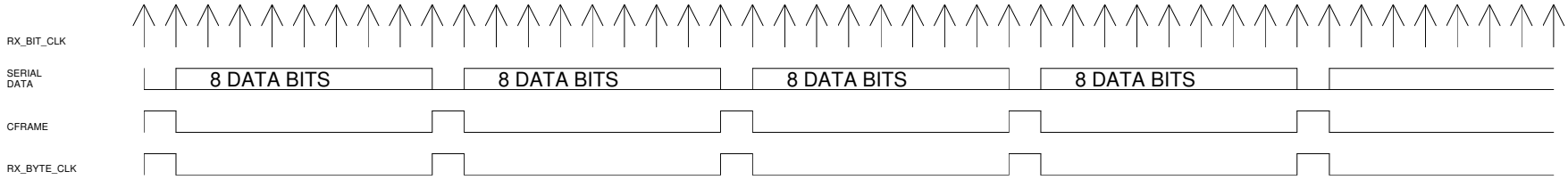


Figure 4. 9 BIT to 8 BIT TRANSFER 1:2 AND 1:4 DE-MULTIPLEXED
(pipeline delays not shown)

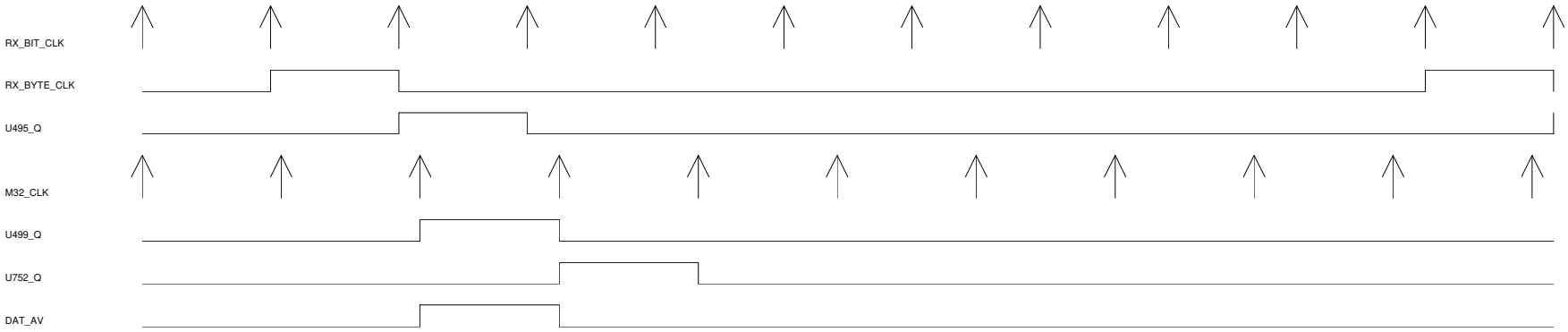


Figure 5. RX_BYTE_CLK to DAT_AV INTERFACE (RX_BIT_CLK > M32_CLK)

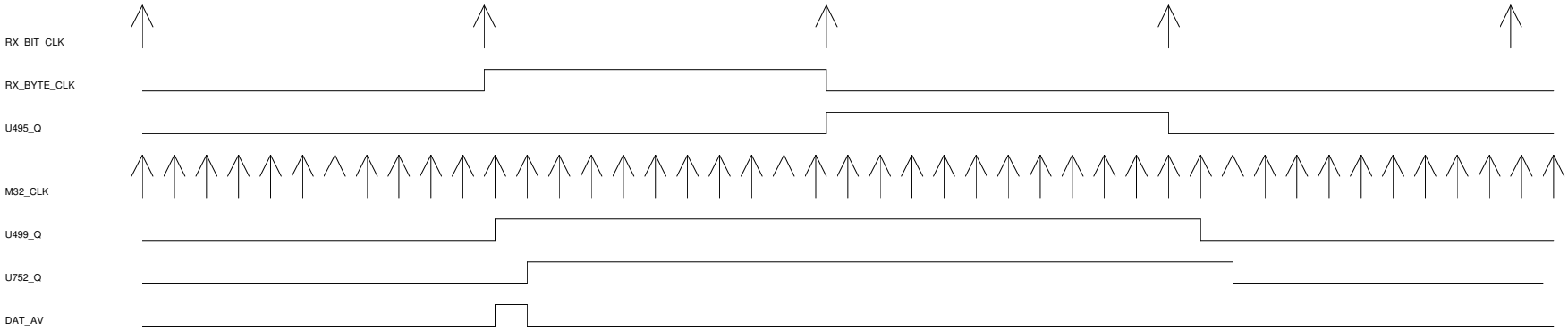


Figure 6. RX_BYTE_CLK to DAT_AV INTERFACE (RX_BIT_CLK << M32_CLK)

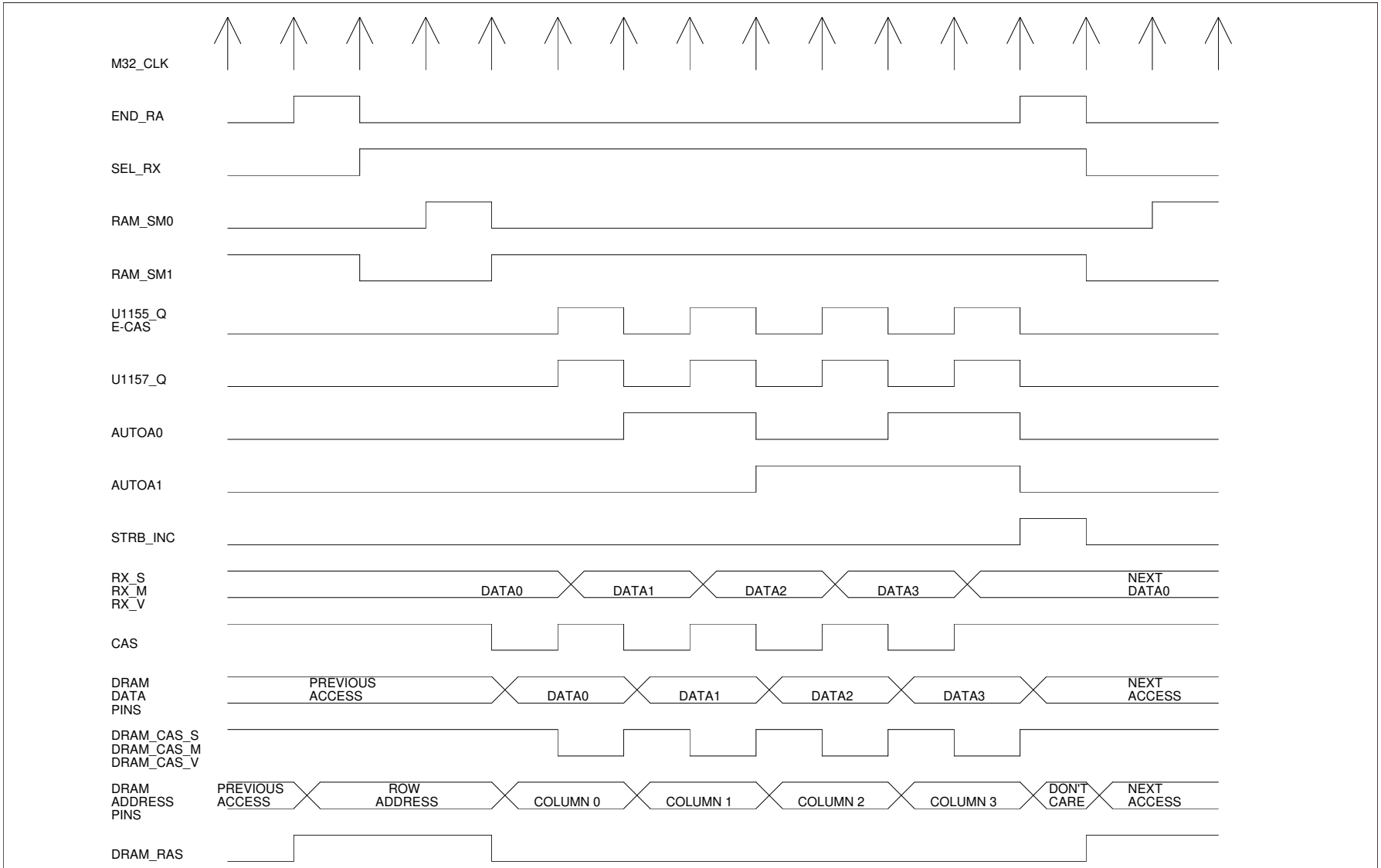


Figure 7. WRITE ACCESS TO DRAM's

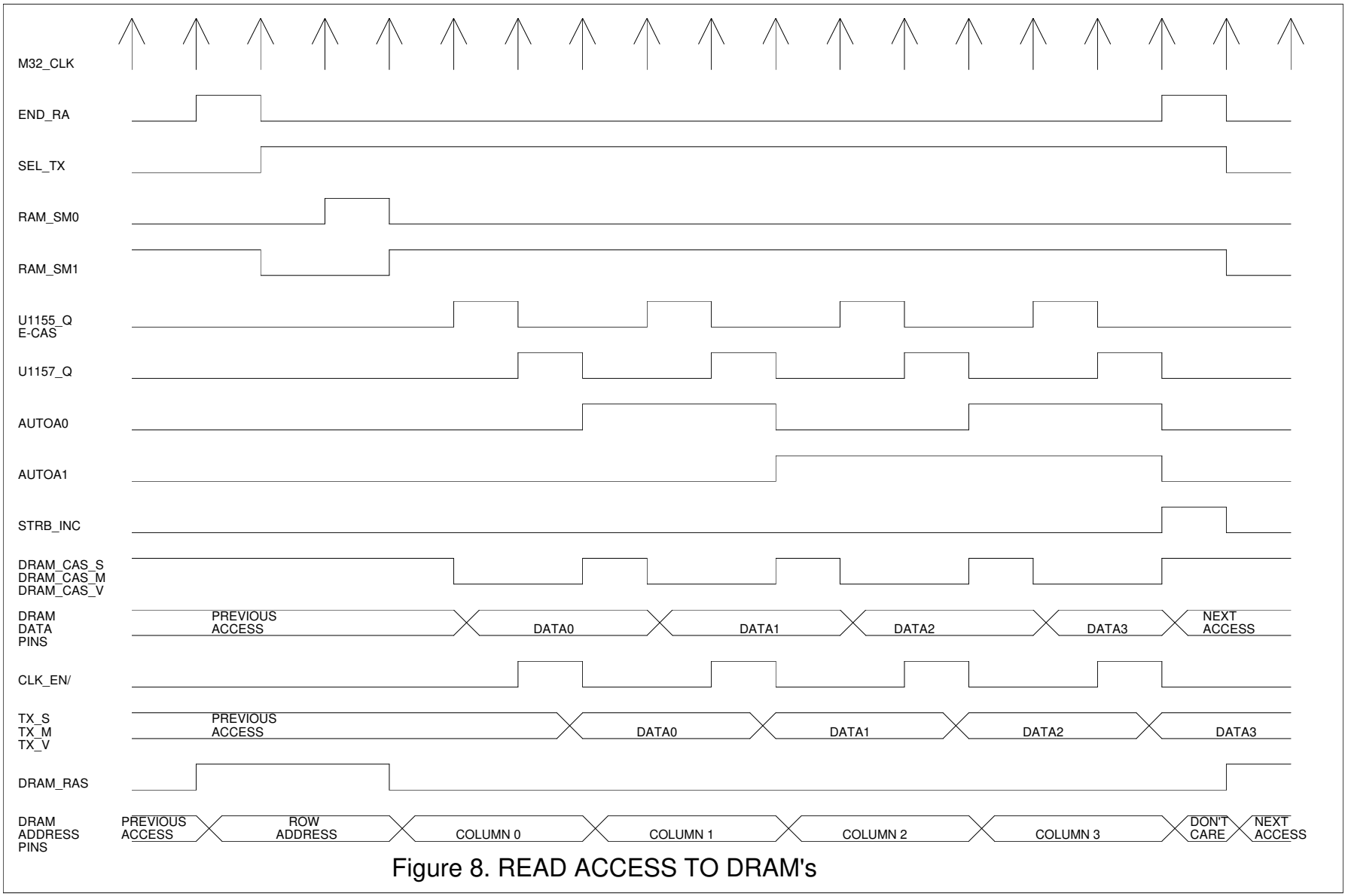


Figure 8. READ ACCESS TO DRAM's

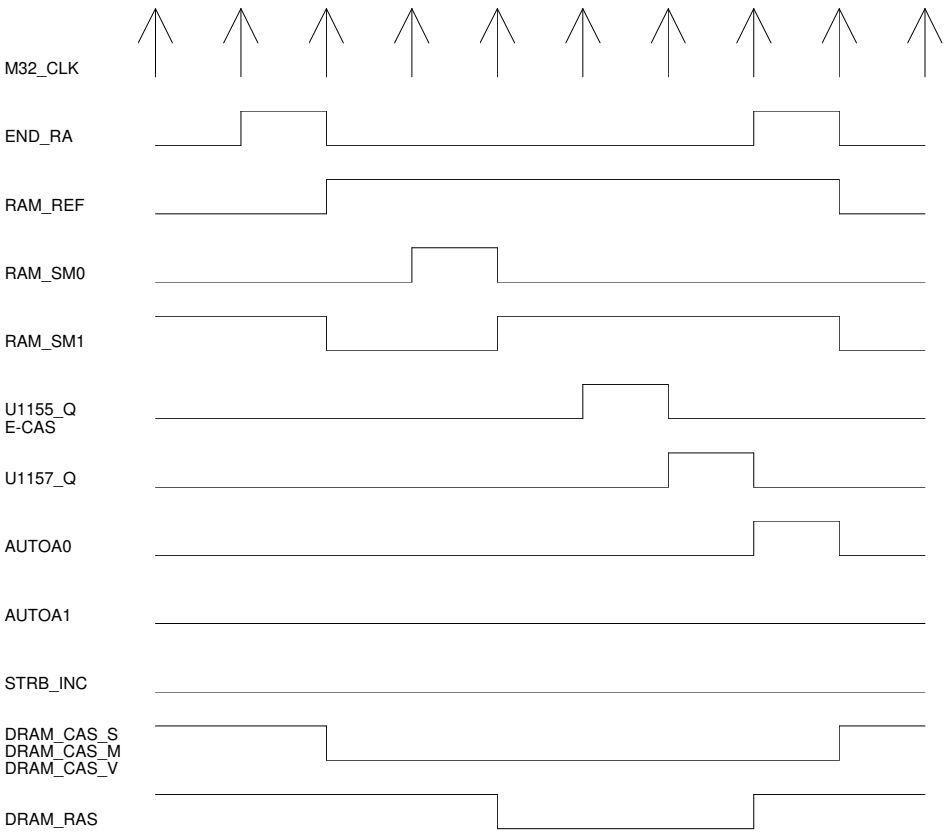


Figure 9. REFRESH ACCESS TO DRAM's

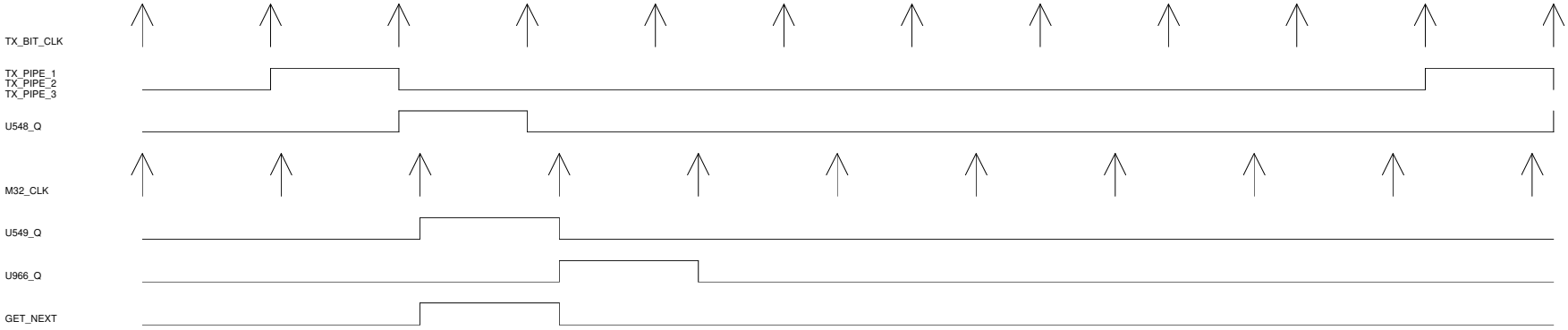


Figure 10. TX_PIPE_(1,2,3) to GET_NEXT INTERFACE (TX_BIT_CLK > M32_CLK)

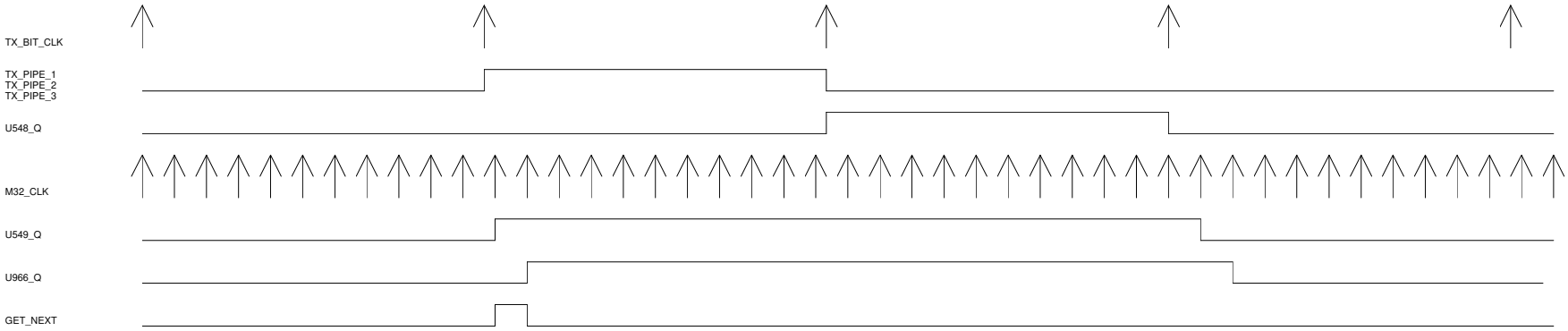


Figure 11. TX_PIPE_(1,2,3) to GET_NEXT INTERFACE (TX_BIT_CLK << M32_CLK)

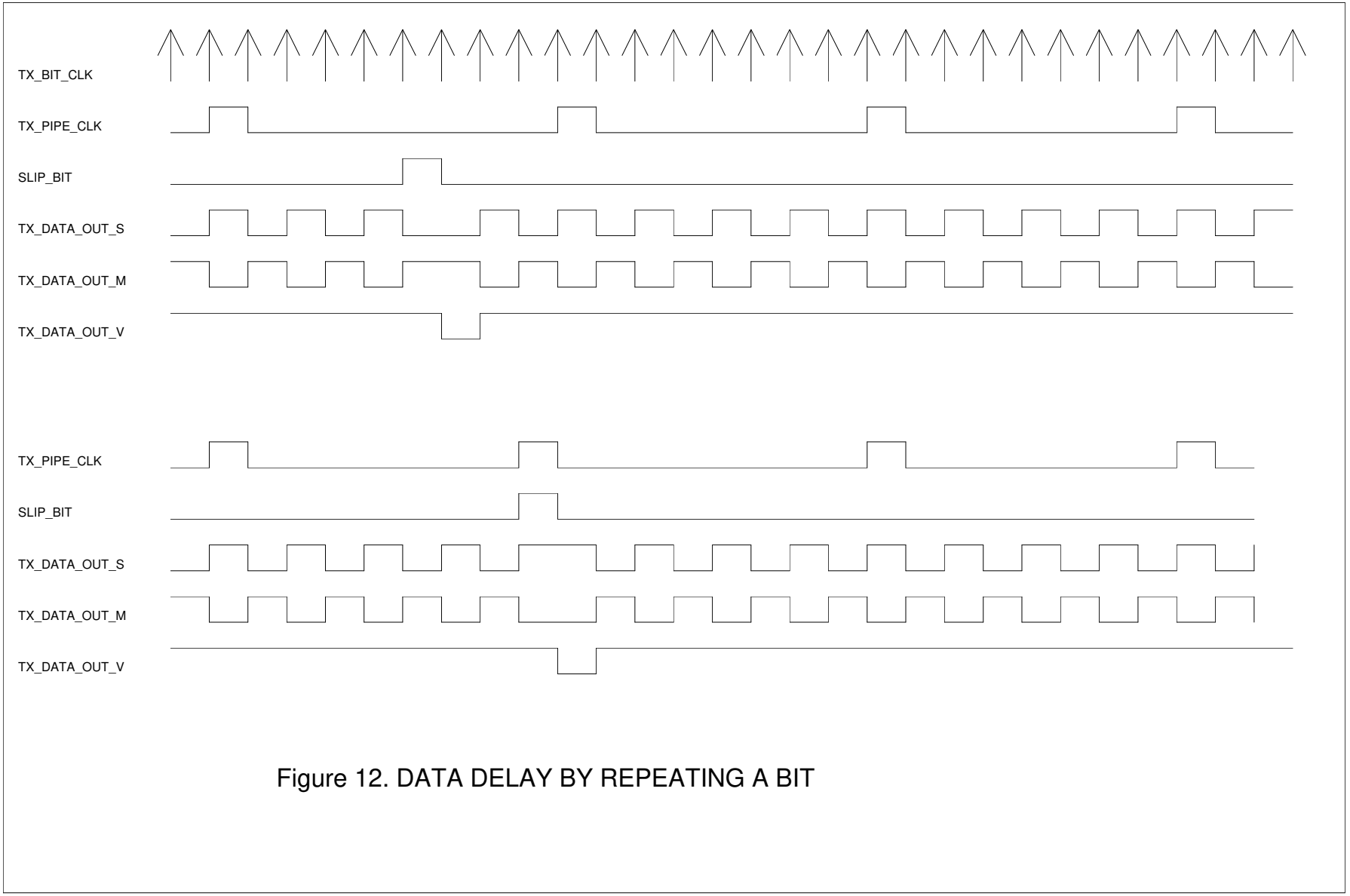


Figure 12. DATA DELAY BY REPEATING A BIT

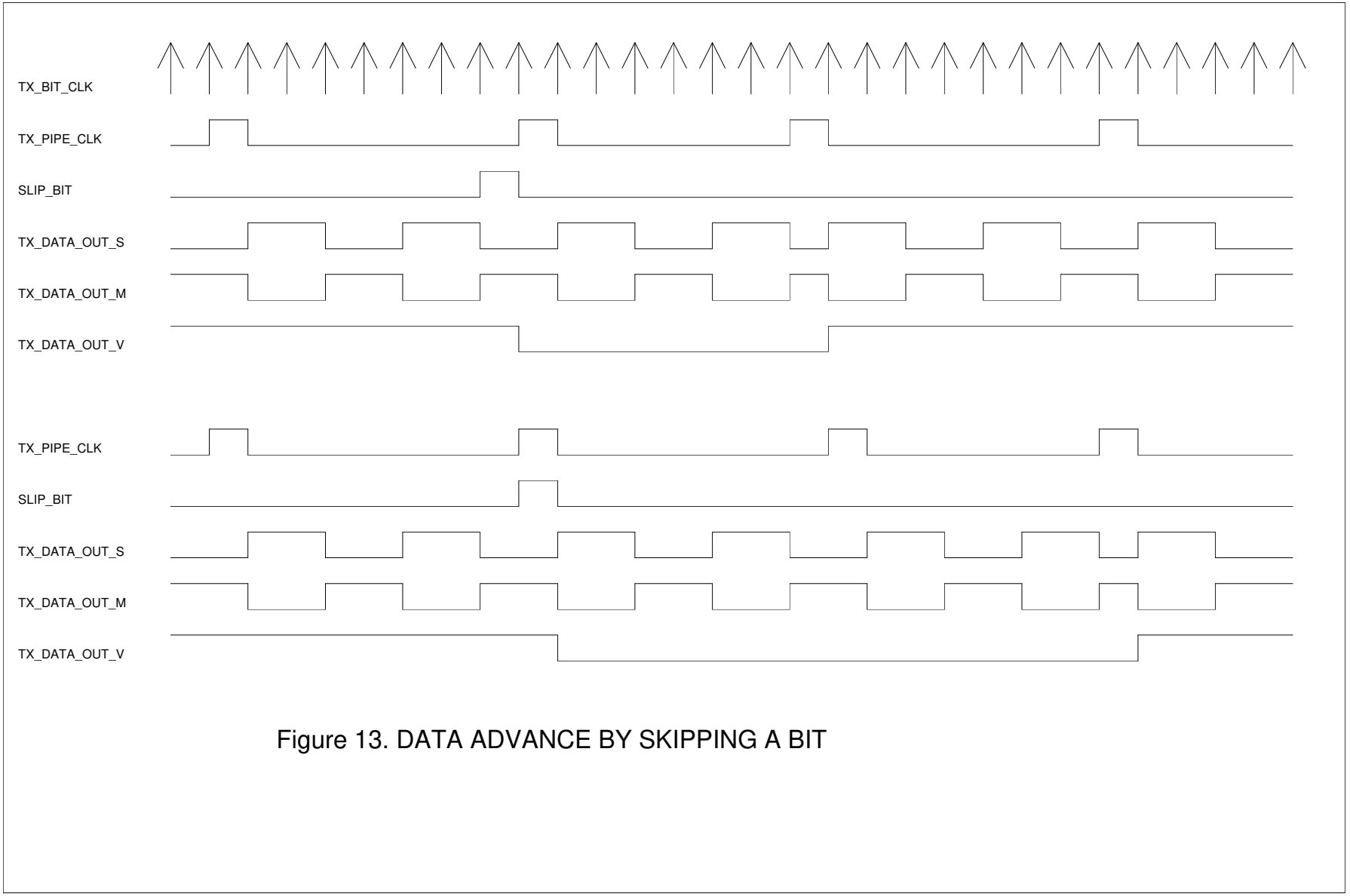


Figure 13. DATA ADVANCE BY SKIPPING A BIT

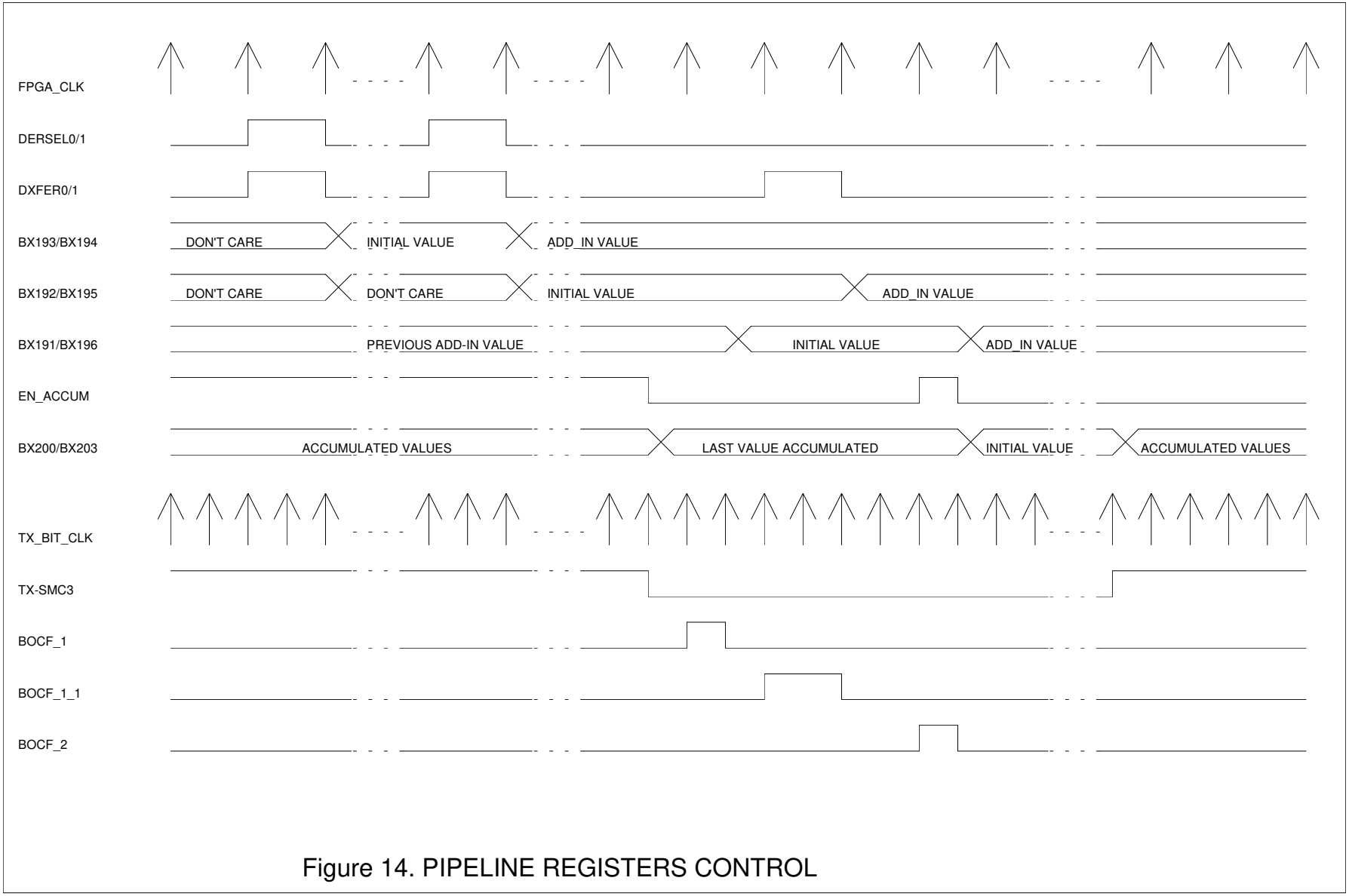


Figure 14. PIPELINE REGISTERS CONTROL