TẬP ĐOÀN CÔNG NGHIỆP
VIỄN THÔNG-QUÂN ĐỘI
VIETTEL

DESIGN SPECIFICATION

MODEL

DMAC (Direct memory access controller)

Author: Huan Nguyen-Duy
Date: 7/31/2024

CATALOG

FIGURE

TABLE

Error! No table of figures entries found.

1. OVERVIEW

The document illustrates the design specification for DMAC (direct memory access controller). The DMAC contain 256 channels for peripherals, the lower channel will have higher priority compared to others. On the other hand, the user can test operation of DMAC model by using register instead of signal request. In particular, DMAC mainly support 4 features: memory to peripheral, peripheral to peripheral, peripheral to memory, memory to memory.

2. DMAC ARCHITECTURE

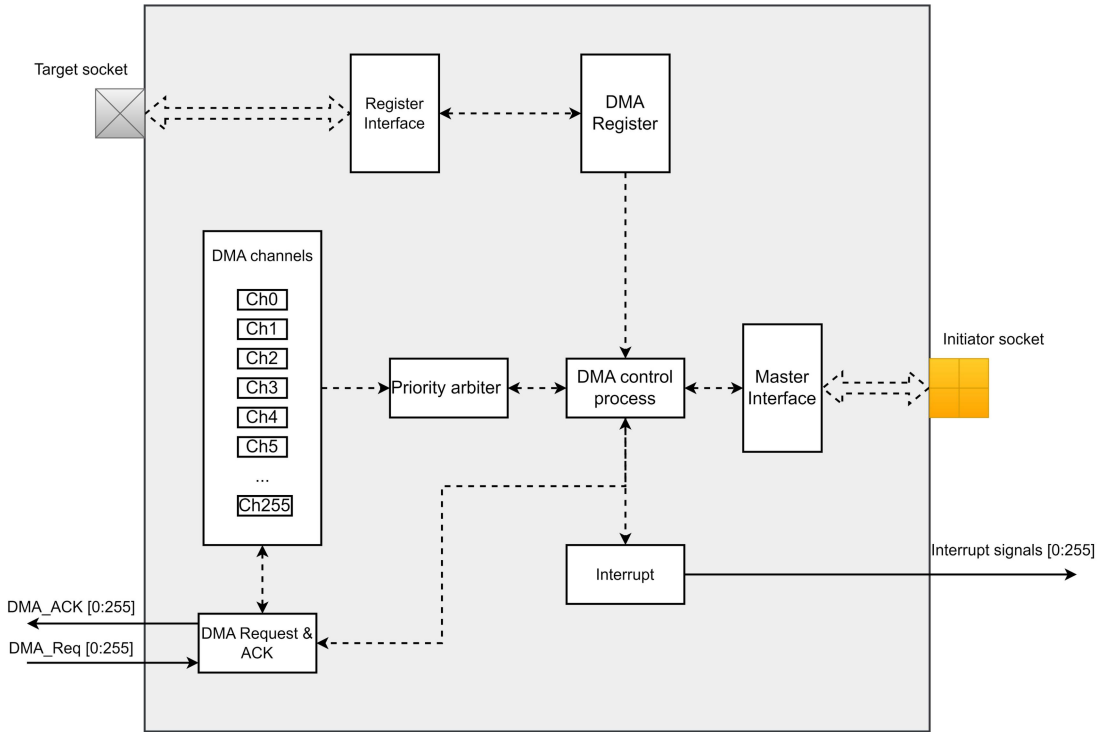

Figure 1: DMAC architecture

Figure 1 show DMAC architecture, including 256 DMA channels and register interface, lower channels have higher priority than others. Specially, DMAC registers can be accessed by register interface.

In particular, the priority arbiter will decide the channel that is used by DMA request signals (the channel 0 is the highest priority). After that, the DMA controller process gets the source and destination address from registers, starting DMA operation and returning ACK to peripherals and interrupts to CPU. During DMA operation, other requests will be ignored.

3. Register

Overall, DMAC model supports 256 channels DMA register to store source address, destination address, and data length that corresponds with 256 DMA channels. Moreover, it offers 32 channel registers to enable or disable DMA channels.

In test mode operation, DMAC model offers 32 channels registers for DMA request operation, interrupt, and and acknowledge operation.

Register name	Base address
DMADESADD(i) (i = [0,255])	0x00 + 0x04*(i)
DMASRCADD(i) (i = [0,255])	0x400 + 0x04*(i)
DMADATALENGTH(i) (i = [0,255])	0x800 + 0x04*(i)
DMAREQ(i) (i = [0,31])	0xC00 + 0x04*(i)
DMAACK(i) (i = [0,31])	0xC20 + 0x04*(i)
DMAINT(i) (i = [0,31])	0xC40 + 0x04*(i)
DMACHEN(i) (i = [0,31])	0xC60 + 0x04*(i)

3.1. DMADESADDRi (0 - 255)

31	24	16
DESADDR		
DESADDR		
14	8	0

The DMADESADDR register contain 256 channels corresponding with 256 DMA channels

Bit name	Range	Permission	Description
DESADDR	[0,31]	R/W	Containing destination address information

3.2. DMASRCADDRi (0 - 255)

31	24	16
SRCADDR		
SRCADDR		
14	8	0

The DMASRCADDR register contain 256 channels corresponding with 256 DMA channels

Bit name	Range	Permission	Description
SRCADDR	[0,31]	R/W	Containing source address information

3.3. DMADATALENGTHi (0 - 255)

31	24	16
DATALENGTH		
DATALENGTH		
14	8	0

The DMASRCADDR register contain 256 channels corresponding with 256 DMA channels

Bit name	Range	Permission	Description
DATALENGTH	[0,31]	R/W	Containing data length information

3.4. DMACHENi (0 - 31)

31							24								16
En 31*i	En 30*i	En 29*i	En 28*i	En 27*i	En 26*i	En 25*i	En 24*i	En 23*i	En 22*i	En 21*i	En 20*i	En 19*i	En 18*i	En 17*i	En 16*i
En 15*i	En 14*i	En 13*i	En 12*i	En 11*i	En 10*i	En 9*i	En 8*i	En 7*i	En 6*i	En 5*i	En 4*i	En 3*i	En 2*i	En 1*i	En 0*i
14							8								0

The DMACHEN register contain 32 channels corresponding with 256 DMA channels. The channel will be enable if the corresponding En bit is high.

Bit name	Range	Permission	Description
En n*I (n = [0,31], I = [0,31])	[n,n]	R/W	Enable or disable the corresponding DMA channels. 0x01: Enable DMA operation 0x00: Disable DMA operation

3.5. DMAREQi (0 - 31)

31							24								16
Req 31*i	Req 30*i	Req 29*i	Req 28*i	Req 27*i	Req 26*i	Req 25*i	Req 24*i	Req 23*i	Req 22*i	Req 21*i	Req 20*i	Req 19*i	Req 18*i	Req 17*i	Req 16*i
Req 15*i	Req 14*i	Req 13*i	Req 12*i	Req 11*i	Req 10*i	Req 9*i	Req 8*i	Req 7*i	Req 6*i	Req 5*i	Req 4*i	Req 3*i	Req 2*i	Req 1*i	Req 0*i
14							8								0

The DMACREQ register contain 32 channels corresponding with 256 DMA channels. The channel will be triggered DMA operation if the corresponding Req bit is high.

Bit name	Range	Permission	Description
Ack n*I (n = [0,31], I = [0,31])	[n,n]	R/W	Trigger or disable DMA operation of the corresponding DMA channels. 0x01: Trigger DMA operation 0x00: Not trigger DMA operation

3.6. DMAACKi (0 - 31)

31							24								16
Ack 31*i	Ack 30*i	Ack 29*i	Ack 28*i	Ack 27*i	Ack 26*i	Ack 25*i	Ack 24*i	Ack 23*i	Ack 22*i	Ack 21*i	Ack 20*i	Ack 19*i	Ack 18*i	Ack 17*i	Ack 16*i
Ack 15*i	Ack 14*i	Ack 13*i	Ack 12*i	Ack 11*i	Ack 10*i	Ack 9*i	Ack 8*i	Ack 7*i	Ack 6*i	Ack 5*i	Ack 4*i	Ack 3*i	Ack 2*i	Ack 1*i	Ack 0*i
14							8								0

The DMACACK register contain 32 channels corresponding with 256 DMA channels. It is used to store the state of DMA operation.

Bit name	Range	Permission	Description
Ack n*I (n = [0,31], I = [0,31])	[n,n]	R/W	Using to store the state of DMA operation 0x01: DMA operation is done. 0x00: DMA operation is not done.

3.7. DMAINTi (0 - 31)

31							24								16
Int 31*i	Int 30*i	Int 29*i	Int 28*i	Int 27*i	Int 26*i	Int 25*i	Int 24*i	Int 23*i	Int 22*i	Int 21*i	Int 20*i	Int 19*i	Int 18*i	Int 17*i	Int 16*i
Int 15*i	Int 14*i	Int 13*i	Int 12*i	Int 11*i	Int 10*i	Int 9*i	Int 8*i	Int 7*i	Int 6*i	Int 5*i	Int 4*i	Int 3*i	Int 2*i	Int 1*i	Int 0*i
14							8								0

The DMACINT register contain 32 channels corresponding with 256 DMA channels. It is used to store the state of interrupt that corresponds with each channel.

Bit name	Range	Permission	Description
Int n*I (n = [0,31], I = [0,31])	[n,n]	R/W	Using to store the state of interrupt that corresponds with each channel 0x01: Interrupt is high 0x00: Interrupt is low

4. SEQUENCE DIAGRAM

The DMAC mainly supports 4 features, including transferring data from memory to peripheral, memory to memory, peripheral to peripheral, and peripheral to memory. All operations is processed by a initiator socket that have role as master in bus MMIO.

4.1. Memory to peripheral

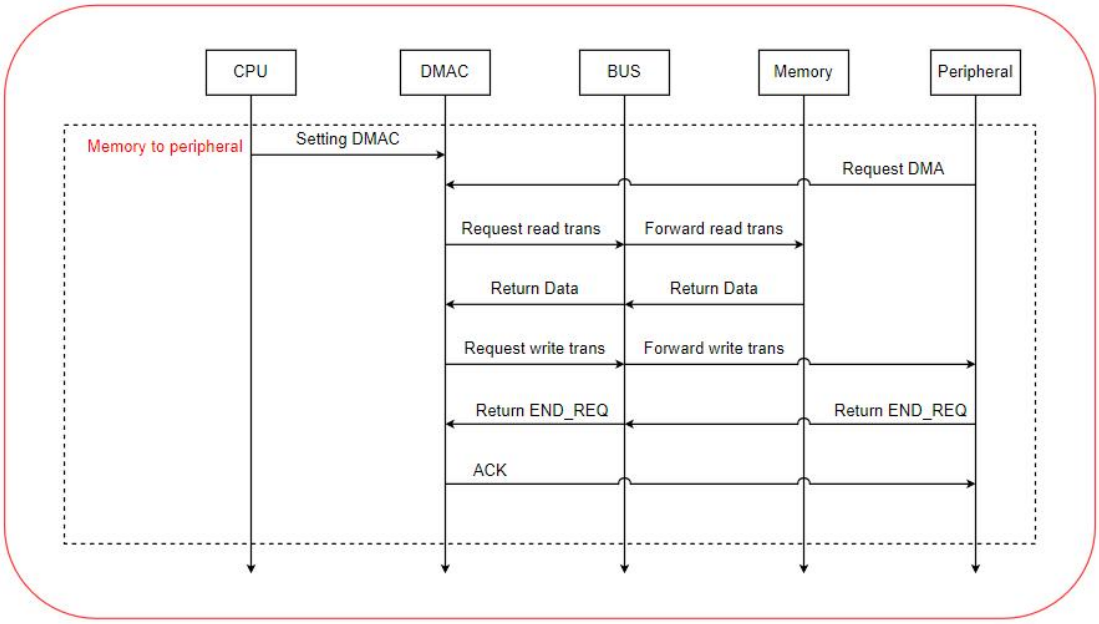

Figure 2: memory to peripheral sequence diagram

Before DMAC operation, registers are configured by CPU. The source address is the address of memory, and the destination address is the address of peripheral. When request signal is triggered by peripheral, the DMA operation is process and return ACK signal to peripheral when it done.

4.2. Peripheral to memory

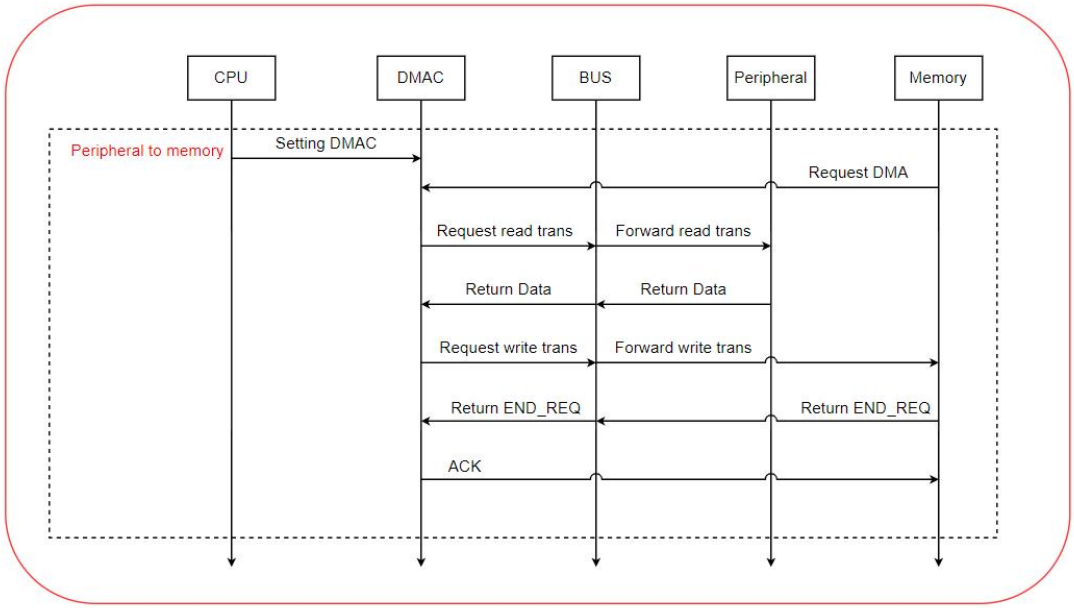

Figure 3: Peripheral to peripheral sequence diagram

Before DMAC operation, registers are configured by CPU. The source address is the address of peripheral, and the destination address is the address of memory. When request signal is triggered by memory, the DMA operation is process and return ACK signal to memory when it done.

4.3. Peripheral to peripheral

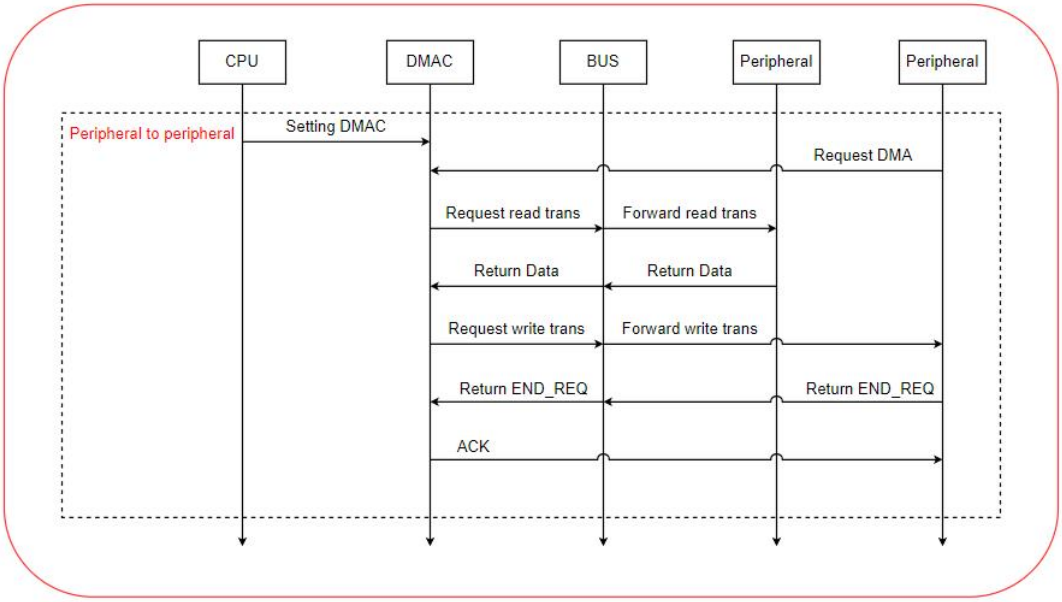

Figure 4: Peripheral to peripheral sequence diagram

Before DMAC operation, registers are configured by CPU. The source address is the address of peripheral, and the destination address is the address of peripheral. When request signal is triggered by peripheral, the DMA operation is process and return ACK signal to peripheral when it done.

4.4. Memory to memory

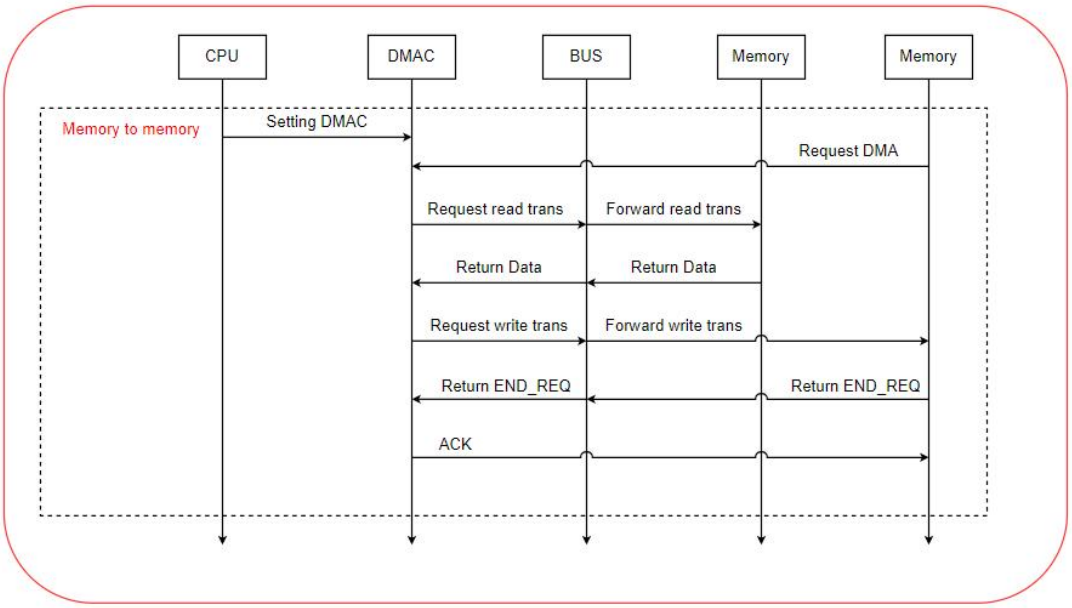

Figure 5: Memory to memory sequence diagram

Before DMAC operation, registers are configured by CPU. The source address is the address of memory , and the destination address is the address of memory . When request signal is triggered by memory , the DMA operation is process and return ACK signal to memory when it done.

5. FLOW DIAGRAM

In this section, the document mainly explains detail about flow diagrams of each function that demonstrates for DMAC operation. The flow diagrams illustrate for operations of each function, thread, and method. Note that the document just provide flow diagram for main functions that have important roles in DMA operation.

5.1. Nb_transport_fw

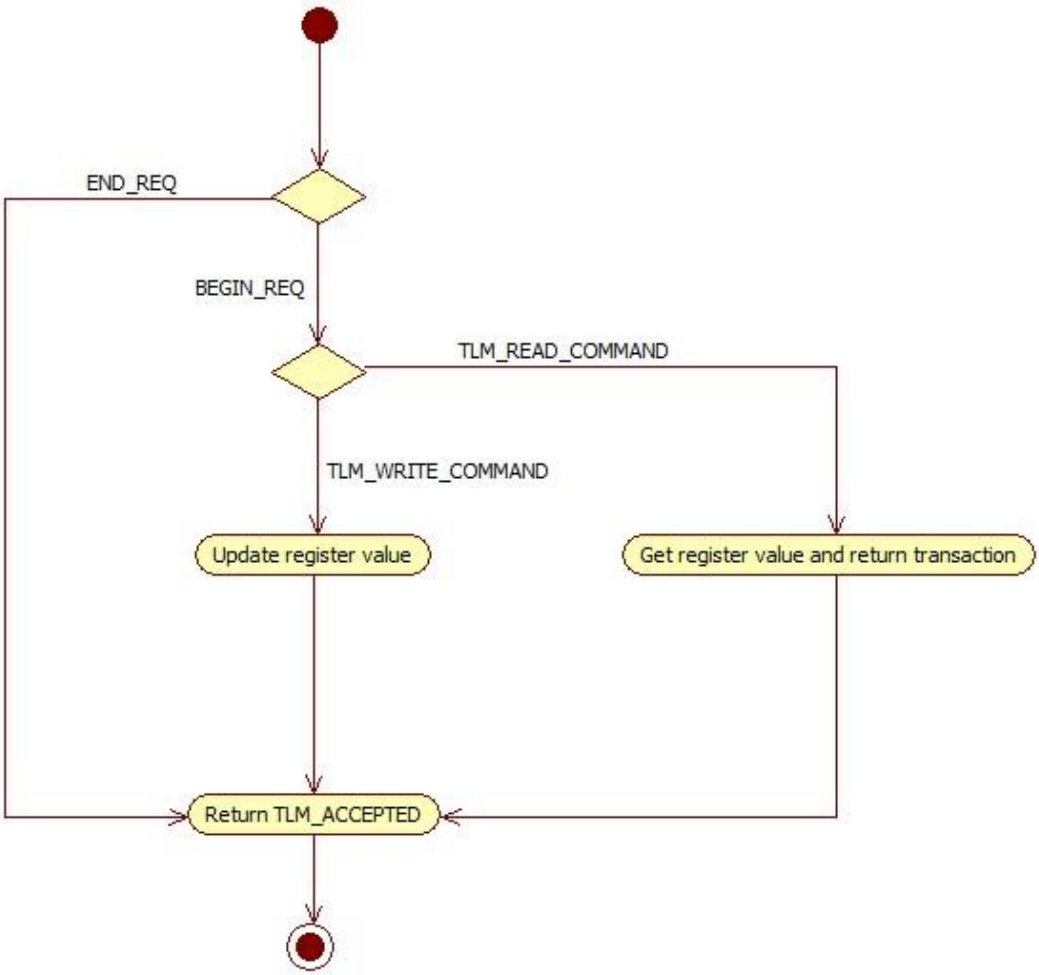

Figure 6: nb_transport_fw flow diagram

The nb_transport_fw function is called by target socket as callback function. When target socket receives data from bus, the operation of nb_transport_fw is decoding address and copying the data into the corresponding register.

5.2. Nb_transport_bw

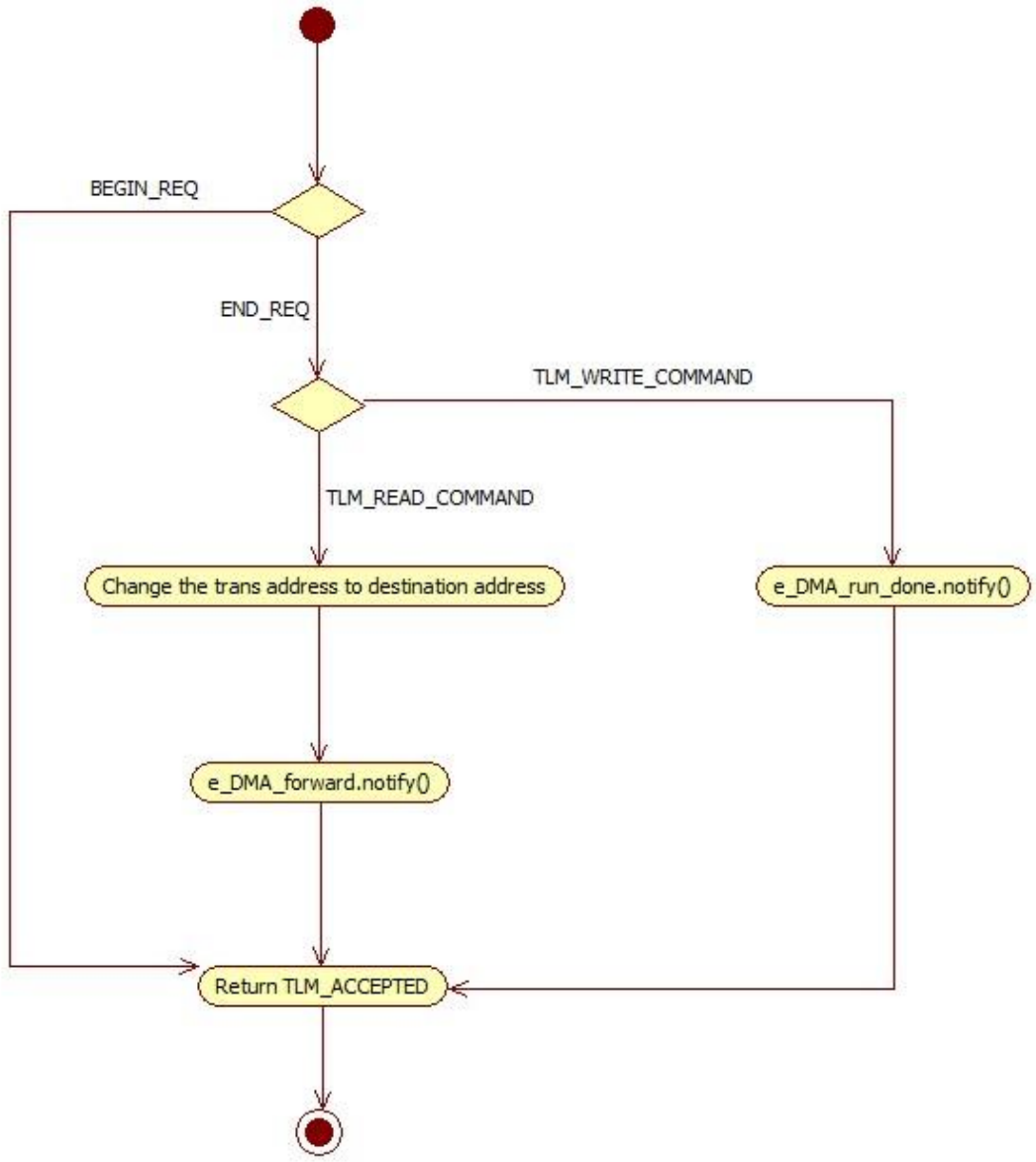

Figure 7: nb_transport_bw flow diagram

The nb_transport_bw is called by initiator socket as callback function, when the data is return from the source, it changes the address to destination address and triggers forwarding transaction operation.

5.3. Mth_request_signals

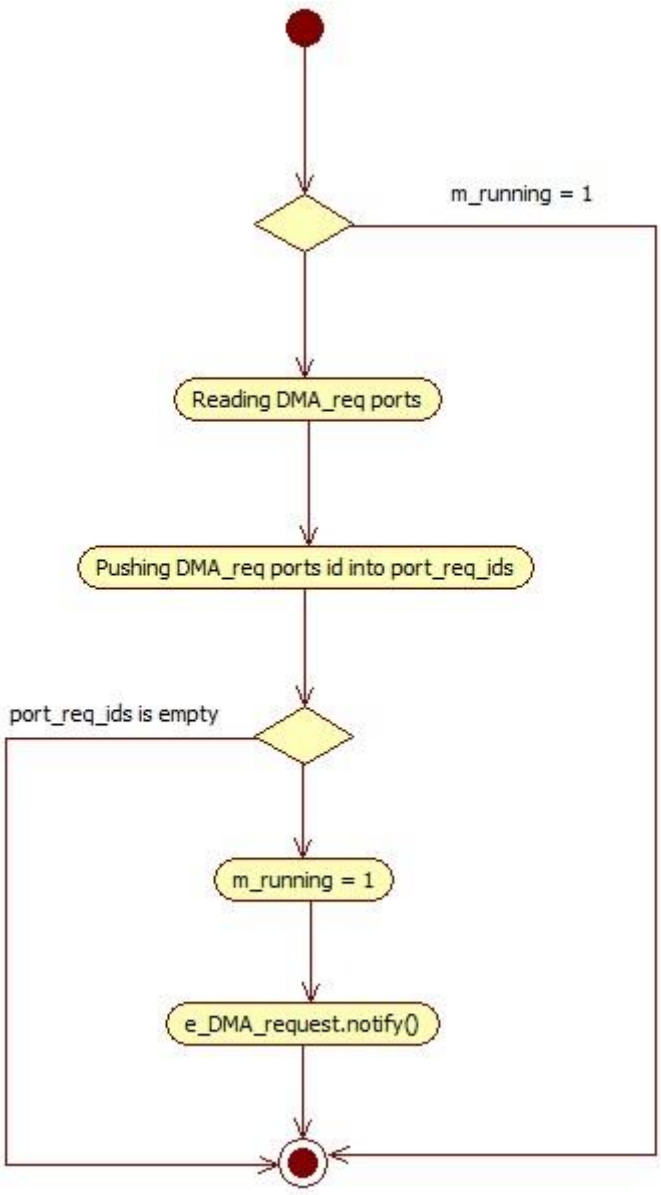

Figure 8: mth_request_signals flow diagram

The mth_request_signals operates as SC_METHOD, when any DMA_req signals is triggered , it will be called. In the end, it will trigger DMA request to handle priority operation.

5.4. Thr_priority_process

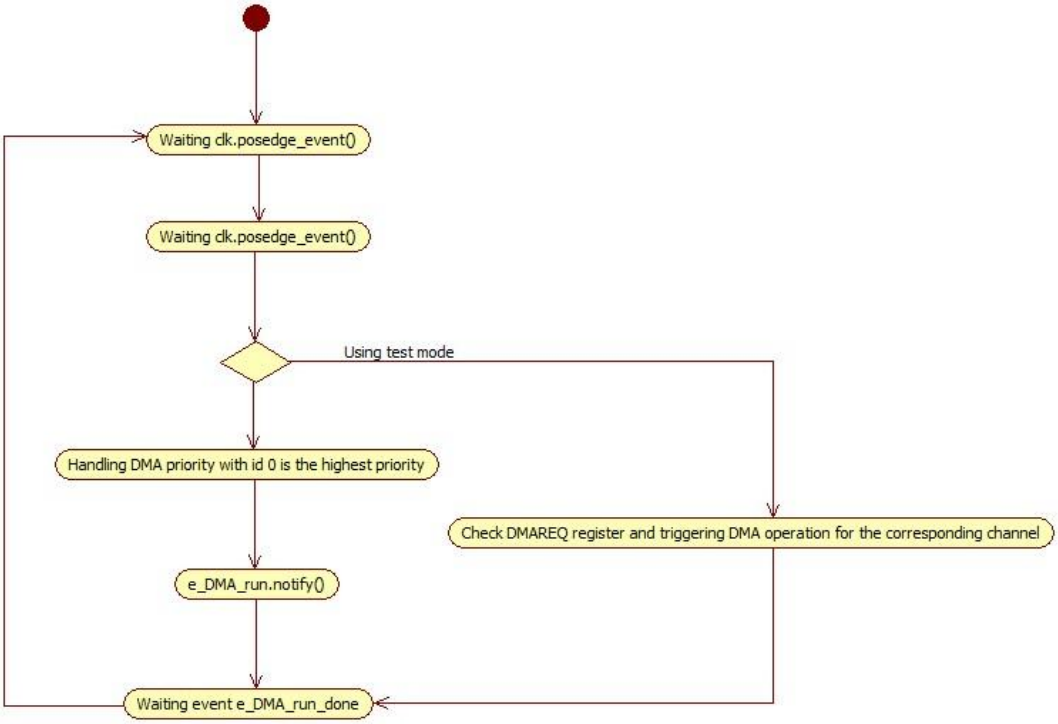

Figure 9: thr_priority_process flow diagram

The thr_priority_process operates as SC_THREAD, it waits DMA request event and synchronizes with clock cycle. In this state, DMA will check all DMA_req ports and push available triggered port id into a queue. The DMA run operation will be called by triggering DMA_run event.

5.5. Thr_DMA_run_process

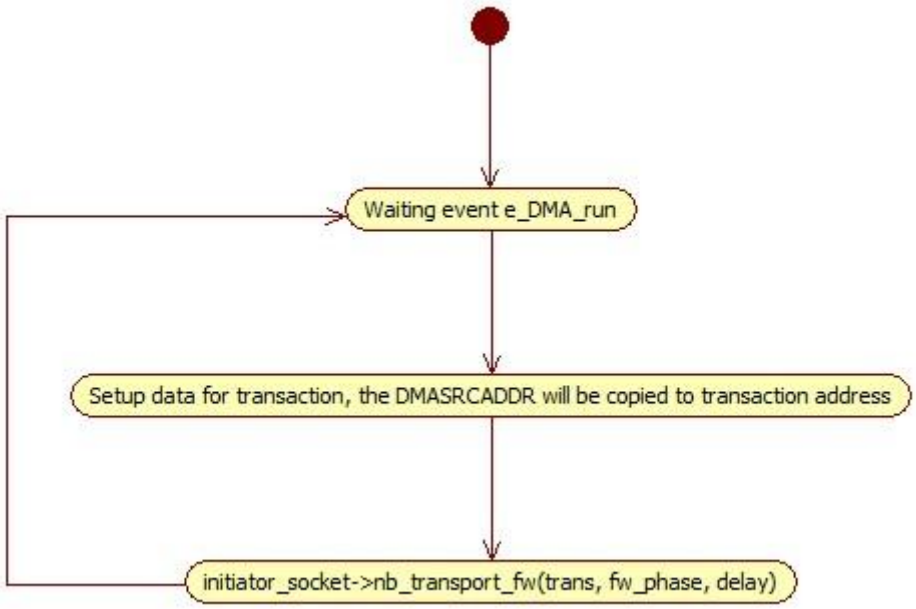

Figure 10: thr_DMA_run_process flow diagram

The thr_DMA_run_process operates as SC_THREAD, it waits DMA run event. In this process, DMA prepares transaction and sends it through initiator socket.

5.6. Thr_DMA_forward_process

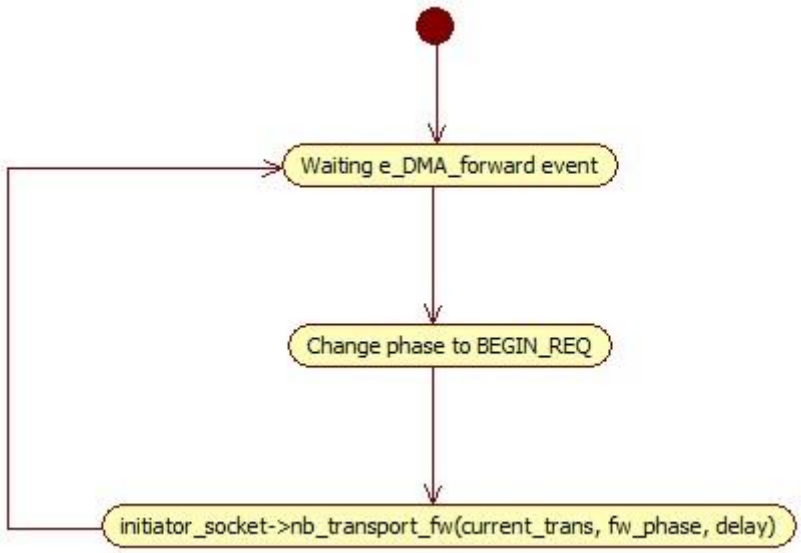

Figure 11: thr_DMA_forward_process flow diagram

The thr_DMA_forward_process operates as SC_THREAD, it will be triggered by e_DMA_forward event. In this process, DMAC will change the forward state to BEGIN_REQ, it occur when the reading data from source was complete.