# AER VHDL code documentation

This is the VHDL code for the Xilinx Coolrunner2 of the DVS128 USB camera system.

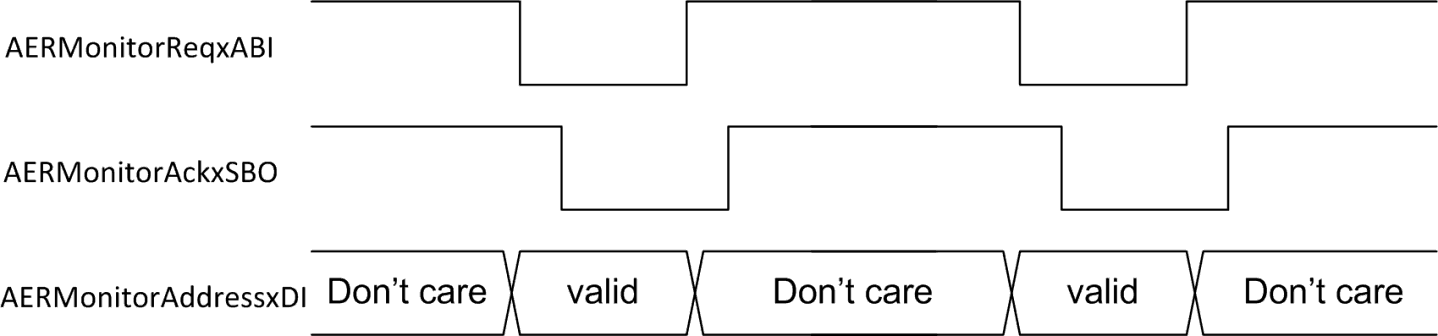
It handshakes with an AER sender and if Run is active, writes the events out to a FiFo interface. In this current implementation, the state machine does not write to the FiFo if the FifoInFull signal is active.

With every address, a timestamp is written to the Fifo. To use bandwidth more efficiently, the state machine could be changed in a way so that the timestamp is only written if it is not equal to the timestamp of the last event.

The timestamp counter uses 14 bits, the tick (how many clock cycles per timestamp increment) is right now fixed by a constant to produce a 1us timestamp. When the 14 bit timestamp counter overflows, a special event is inserted into the event stream. This special event tells the receiver to increment its ‘wrap-add’. This ‘wrap-add’ is added to every 14 bit timestamp received from the device to form a 32 bit timestamp.

There is an input signal to reset the timestamp counter to zero. This will also insert a special event into the event stream; this special event tells the receiver that the timestamp has been reset and the ‘wrap-add’ should be set to zero.

AERMonitorReqxABI and AERMonitorAckxSBO are the handshake signals, the protocol used is four-phase handshake. The address is valid as long as the request is active. See the figure below for illustration.



The AERMonitorReqxABI signal is asynchronous, thus it has to be synchronized to the clock by using a sequence of flip-flops (we normally use two flip-flops in series). Otherwise it could violate setup and hold times of the state machine inputs. In the four-phase handshake AER protocol, timing is unspecified.

Below the different files are described.

## USBAER\_top\_level.vhdl

This is the top level entity. A block diagram is shown in topLevelBlockDiagram.

#### Inputs:

AERMonitorReqxABI : AER request, asynchronous and active low

AERMonitorAddressxDI: AER address

ClockxCI: system clock

ResetxRBI: chip reset, active low

FifoInFullxSBI: event fifo is full, active low. with the current implementation, the AER\_FSM does not write data into the FIFO if full.

HostResetTimestampxSI: reset the timestamp counter to zero. This also inserts a special event into the event stream to tell the receiver to reset its ‘wrap-add’.

RunxSI: If active, events are written into the fifo, if inactive, the monitor state machine handshakes with the AER sender, but does not write events to fifo and timestamp counter is not incremented.

#### Outputs:

AERMonitorAckxSBO: AER acknowledge, active low

FifoDataxDO: data to be written into event fifo

FifoWritexEBO: fifo write enable, active low.

## monitorStateMachine.vhdl

This is the state machine that handshakes with the AER sender. The monitorStateMachine tells the fifoStateMachine that it acquired an event by means of the monitorEventReady register.

If the fifoAlmostFull input is active, the monitorStateMachine continues to handshake with the AER sender, but does not set the monitorEventReady register, events are discarded. Trigger events are still written until the fifoInFull is active.

#### Inputs:

AERMonitorReqxSBI : AER request, after synchronization, active low

ClockxCI: system clock

ResetxRBI: chip reset, active low

FifoInFullxSI: event fifo is full, active high. In the current implementation, the AER\_FSM does not write data to the FIFO if the Fifo is full (and thus FifoInFullxSI is active), but this behaviour should be selectable with a configuration input, to select whether to overwrite old data or not.

RunxSI: If active, events are written into the fifo, if inactive, AER\_FSM handshakes with AER sender, but does not write events to fifo.

TimestampOverlowxSI: timestamp counter is overflown, write a timestamp wrap message to fifo

ResetTimestampxSBI: timestamp is reset, active low. Write timestamp reset message to fifo.

#### Outputs:

AERMonitorAckxSBO: AER acknowledge, active low

FifoWritexEO: fifo write enable

AddressTimestampSelectxSO: controls the FifoData multiplexer, four different inputs:

- AER address

- AER timestamp

- timestamp wrap message

- timestamp reset message

AddressMSBxDO: bit 15 of the fifo data, selects whether this fifo data word is an address (timestamp wrap and timestamp reset are special addresses) or a timestamp.

TimestampRegWritexEO: write timestamp to register

## fifoStateMachine.vhd

This state machine controls writing to the Fifo in the FX2 USB2 transceiver.

## synchronizerStateMachine.vhd

Controls the timestamp counter. The tick (how many clock cycles per timestamp increment) is fixed to 1 us. It receives an external input that tells the state machine to reset the timestamp counter. This input can be active for several clock cycles. The output ResetTimestampxSBO is active for only one clock cycle to avoid sending multiple ‘reset timestamp’ events.

The ConfigxSI input defines whether this device acts as timestamp master (controlling the timestamp counter itself) or as timestamp slave (the timestamp increment is synchronized to an external device).

In timestamp slave mode, the device starts incrementing the timestamp counter on the falling edge of SyncInxABI. It then increments the timestamp counter 99 times. Then again it waits for a falling edge of SyncInxABI, to increment the timestamp counter. This way, the timestamps counter is resynchronized to the timestamp master every 100us.

In timestamp master mode, the timestamp is incremented every microsecond if RunxSI is active. A 10kHz square wave clock is output on SyncOutxSBO to synchronized potential slave devices. On falling edges of SyncInxABI, the synchronizerStateMachine tells the monitorStateMachine by means of TriggerxS to insert a trigger event into the event stream. Using this special event the camera output can be synchronized to external stuff.

#### Inputs:

ClockxCI: system clock

ResetxRBI: chip reset, active low

HostResetTimestampxSI: reset the timestamp counter

RunxSI: If active, timestamp is incremented every counterInc clock cycles, if inactive, timestamp counter is not incremented.

ConfigxSI: Whether this device is timestamp master or timestamp slave.

SyncInxABI:

#### Outputs:

ResetTimestampxSBO: reset the timestamp counter and signal the AER\_FSM to write a timestamp reset message to fifo.

IncrementCounterxSO: increment timestamp counter by one.

## timestampCounter.vhd

This is the counter that holds the current time. If it wraps around, it tells the fifoStateMachine so by means of the OverflowxSO output. The fifoStateMachine then writes a timestamp wrap message to the Fifo.

#### Inputs:

ClockxCI: system clock

ResetxRBI: reset, active low (this is not chip reset, this reset signal is specific to the timestamp counter). Reset the counter value to zero.

IncrementxSI: increment the timestamp by one

#### Outputs:

OverflowxSO: the timestamp has wrapped, active for one clock cycle

DataxDO: the current timestamp

## wordRegister.vhd

A register, size can be specified.

#### Inputs:

ClockxCI: system clock

ResetxRBI: chip reset, active low

WriteEnablexEI: write enable

DataInxDI: register input

#### Outputs:

DataOutxDO: register output

## Signal naming convention:

Signalnamex[CRESD][B][IO]

where

C clock signal

R reset signal

E enable signal

D data signal

S some other signal

B active low signal

I input

O output

IO input/output

---------------------------------

Signal polarities for AERMonitorReq and AERMonitorACK are fixed active low

Polarities of other signals can be changed if necessary.