# 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.

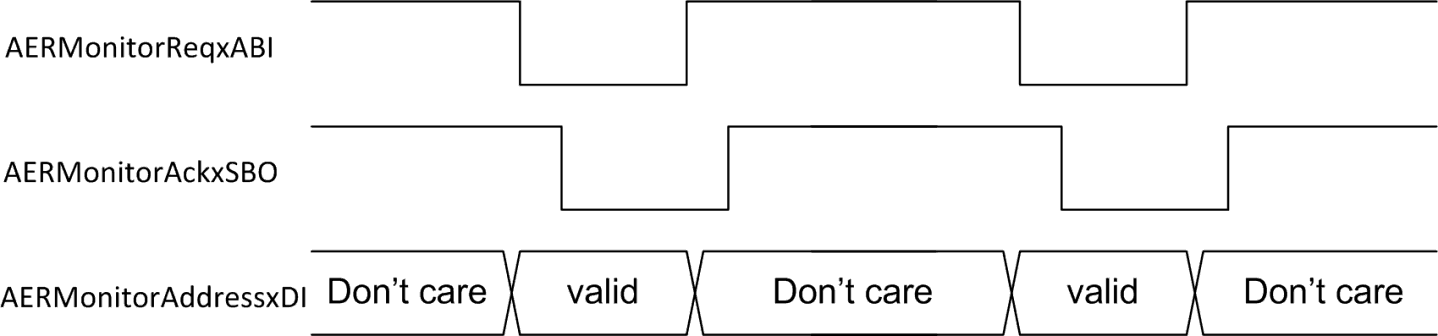
With every address, a timestamp is written to the Fifo. To use bandwidth more efficiently, the state machine could be changed in a way so that the timestamp is only written if it is not equal to the timestamp of the last event.

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.

The timestamp counter can be synchronized to external devices (slave mode). The timestamp counter is controlled by the synchronizerStateMachine.

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.

By means of the earlyPaketTimer and the eventCounter, the code ensure that a paket is commited to the USB domain at least every few milliseconds. This is to ensure that the host computer received data regularly even if the activity is low. Every fifo buffer in the FX2 can hold 128 events, the fifoStateMachine increments the eventCounter for every event that is written to the Fifo and if the counter reaches 128, a paket has been commited automatically and the earlyPaketTimer is reset. If the earlyPaketTimer overflows (which means that the fifoStateMachine did not write 128 events to the fifo recently), the fifostateMachine will assert PaketEndxSBO to tell the FX2 that the current fifo should be commited to the USB domain.

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.

FifoFlagAxSBI (FifoInAlmostFullxSBI): event fifo is almost full, active low. If this signal is active, the monitorSM continues to handshake, but events are discarded.

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’.

RunMonitorxSI: 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.

SyncInxABI: If TimestampMasterxSI=1, falling edges on this input create trigger events. If TimestampMasterxSI=0, this input is used to synchronize the timestamp counter to an external device.

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

PC3xSI (HostTriggerxS): the host computer tells the fifo state machine to insert a trigger event into the event stream.

#### Outputs:

AERMonitorAckxSBO: AER acknowledge, active low

FifoDataxDIO: data to be written into event fifo

FifoWritexEBO: fifo write enable, active low.

FifoReadxEBO: fifo read enable, active low. Unused, hardcoded to ‘1’.

FifoPktEndxSBO: this output is to commit an FX2 Fifo buffer to the USB domain even if it is not full. This is done after the EarlyPaketTimer is expired, to ensure a minimum USB packet rate even if the input data rate is low.

FifoAddressxDO: which FX2 USB Endpoint we are writing data to. Hardcoded.

SyncOutxSBO: If TimestampMasterxSI=1 (timestamp master) and runxSI=1, a 10kHz square wave is output to synchronize potential slave devices. If TimestampMasterxSI=0, the output is equal to syncInxABI

Interrupt1xSBO: interrupt 1 of the Fx2, active low. Unused, hardcoded to ‘1’.

LEDxSO: led control

Debug1xSO and Debug2xSO: debug output that go to a two pin header. Right now configured to be fifofull and fifoalmostfull flags.

## 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.

#### Inputs:

AERMonitorReqxSBI : AER request, asynchronous, active low

ClockxCI: system clock

ResetxRBI: chip reset, active low

FifoInAlmostFullxSBI: event fifo is almost full, active low. If this signal is active, the monitorSM continues to handshake, but events are discarded.

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

EventReadyxSI: if active, the fifo state machine still has not read the last AER event, so the monitor state machine has to wait before it can acquire the next AER event.

#### Outputs:

AERMonitorAckxSBO: AER acknowledge, active low

MonitorRegWritexEO: write AER address to register

SetEventReadyxSO: tell the fifo state machine that there is an AER event to be written to the fifo.

## fifoStateMachine.vhd

This state machine controls writing to the Fifo in the FX2 USB2 transceiver. It is told that there is an event to write to the Fifo by means of the monitorEventReady register. If it has written this event to the Fifo, it clears the monitorEventReady register.

#### Inputs:

ClockxCI: system clock

ResetxRBI: chip reset, active low

FifoInFullxSBI: event fifo is full, active low. AER events and trigger events are discarded, the number of wrap messages is counted and all of them are sent after FifoInFullxSBI is inactive again.

FifoInAlmostFullxSBI: event fifo is almost full, active low.

MonitorEventReadyxSI: Tells the fifoStateMachine that the monitorStateMachine acquired an AER event that should be written to the Fifo

TimestampOverflowxSI: the timestamp counter has overflown, now the fifoStateMachine should write a timestamp wrap message to the fifo.

ResetTimestampxSBI: active low. The timestamp counter has been reset, now the fifoStateMachine should write a timestamp reset message to the fifo.

TriggerxSI: the fifoStateMahcine should write a trigger event to the fifo.

EarlyPaketTimerOverflowxSI: the earlyPaketTimer has overflown, the fifo state machine should assert FifoPktEnd to commit an FX2 fifo buffer to the USB domain.

#### Outputs:

FifoWritexEBO: fifo write enable

FifoPktEndxSBO: this output is to commit an FX2 Fifo buffer to the USB domain even if it is not full. This is done after the EarlyPaketTimer is expired, to ensure a minimum USB packet rate even if the input data rate is low.

FifoAddressxDO: which FX2 USB Endpoint we are writing data to. Hardcoded.

AddressRegWritexEO: writing the current AER address and timestamp to the registers.

ClearMonitorEventxSO: clears the monitorEventReady register. This allows the monitorStateMachine to safe a new event.

IncEventCounterxSO: increments the event counter which is used in conjunction with the earlyPaketTimer

ResetEventCounterxSO: resets the event counter which is used in conjunction with the earlyPaketTimer

ResetEarlyPaketTimerxSO: resets the earlyPaketTimer

AddressTimestampSelectxSO: controls the FifoData multiplexer, two different inputs:

- AER address

- AER timestamp

AddressMSBxDO: bit 15 of the fifo data, selects whether this fifo data word is an AER (‘0’) or a trigger event (‘1’).

TimestampMSBxDO: bit 14 and 15 of the fifo data, selects whether this fifo data word is a timestamp (‘00’), a timestamp wrap message (‘10’) or a timestamp reset message (‘01’).

FifoTransactionxSO: active if the fifoStateMachine is not idle, which means it is writing something to the fifo.

## 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 synchronize 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: If configxSI=1, falling edges on this input create trigger events. If configxSI=0, this input is used to synchronize the timestamp counter to an external device.

#### 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.

TriggerxSO: in timestamp master mode (configxSI=1), a falling edge on SyncInxABI was received.

SyncOutxSBO: If configxSI=1 (timestamp master) and runxSI=1, a 10kHz square wave is output to synchronize potential slave devices. If configxSI=0, the output is equal to syncInxABI

## 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

## HostTriggerSyncGenerator

Creates a one clock cycle output pulse on HostTriggerxSO from the multi-clock cycle input pulse on HostSyncxSI.

This allows inserting special trigger events into the event stream, when the host computer signals to do so. Like this the timing of commands from the host computer can be synchronized with input data.

## EventCounter

Counts the number of events written to the fifo. If this counter reaches 128, it resets the earlyPaket timer.

#### Inputs:

ClockxCI: system clock

ResetxRBI: chip reset, active low

IncrementxSI: increment the counter by one.

ClearxSI: resets the counter to zero

#### Outputs:

OverflowxSO: counter reached 128

## EarlyPaketTimer

Timer to ensure a minimum packet rate on the USB.

#### Inputs:

ClockxCI: system clock

ResetxRBI: chip reset, active low

ClearxSI: resets the timer to zero

#### Outputs:

TimerExpiredxSO: tells the fifoStateMachine to assert PaketEndxSBO to commit a fifo to the USB domain even if the fifo is not full.

## 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.