CMOS Sensor Input

Sahand Kashani

February 28, 2016

1 Core Overview

The `cmos_sensor_input` core is a highly configurable interface that makes it simple to connect a CMOS sensor to a host processor & memory system.

The core is configurable at runtime through an Avalon Memory-Mapped (Avalon-MM) interface, and provides an Avalon Streaming (Avalon-ST) interface from the CMOS sensor.

It can be instantiated to accomodate various sensor pixel depths and sampling edges. Optionally, the core can also perform debayering and can pack multiple pixels together into a bigger word size in order to divide the required acquisition frequency and reduce pressure on the memory system.

The core comes with a set of C library interfaces that can be used to configure it and start its various operations.

2 CMOS Sensors

A CMOS sensor outputs 4 signals with which it is possible to sample its data:

- `clock`
- `frame_valid (1-bit)`
- `line_valid (1-bit)`
- `data (n-bit)`

Figure 1 shows the relationship between the different signals for 2 frames that contains 2 rows and 3 columns each.

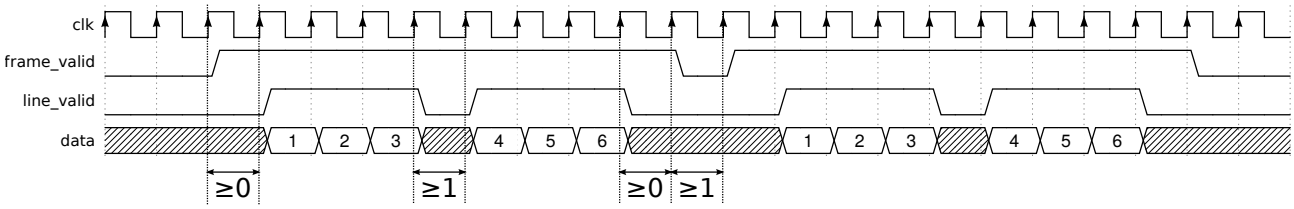

Figure 1: CMOS sensor output signals for two 3×2 frames with a pixel depth of 3 bits. Spacing requirements between the various signals are specified in clock cycles.

3 Block Diagram

Figure 2 shows a high-level view of the core, and Figure 3 shows the building blocks that compose it when in its *largest* configuration (all optional units enabled, i.e. debayering unit and packer).

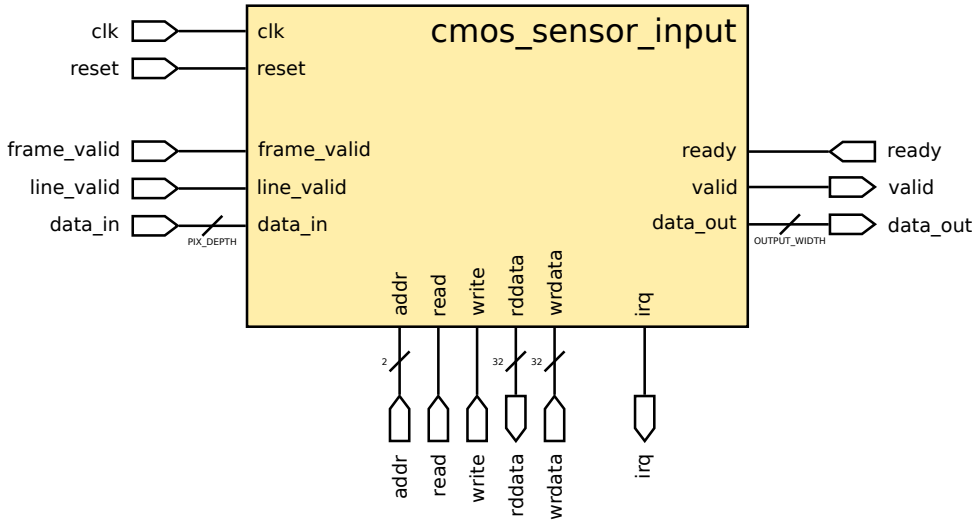

Figure 2: High-level block diagram.

The `cmos_sensor_input` core is clocked by the `clock` output generated by the CMOS sensor and takes the `frame_valid`, `line_valid` and `data` signals as inputs. Note that the `cmos_sensor_input` core does *not* need to be told what the dimensions of the incoming frame are. It solely relies on the `frame_valid` and `line_valid` signals to correctly acquire the data.

The core is composed of 7 components:

`MM-Slave` Provides an Avalon-MM slave interface from the unit to which a host processor can be connected. This interface allows the processor to submit commands and query the status of the unit.

`Synchronizer` Captures all incoming signals from the CMOS sensor. The `synchronizer` can be parameterized to sample signals on the rising or falling edge of its input clock. The signals are synchronized by the `synchronizer` and are sent to the `sampler` on the next rising edge of the clock. All components of the `cmos_sensor_input` core use the rising edge of the input clock for their operations.

`Sampler` Acts as the valve on the stream of raw data coming from the sensor. It is responsible for determining the characteristics of the incoming frame supplied by the `synchronizer`, and, more importantly, for filtering and modifying the data and control signals to an internal format suitable for deterministic processing by the rest of the system.

`Debayer` Applies a 3×3 debayering pattern over the incoming frame supplied by the `sampler`. The debayering pattern used can be configured at runtime to accomodate for the 4 possible pixel layouts of any sensor.

`Packer` Packs consecutive pixels received from the previous stage into a larger word. When no more pixels can be packed in the output word size, then the word is sent out of the unit.

`SC_FIFO` Buffer that stores data ready to be sent out of the unit.

`ST-Source` Provides an Avalon-ST source interface from the unit. The unit supports backpressure due to the presence of the `ready` port.

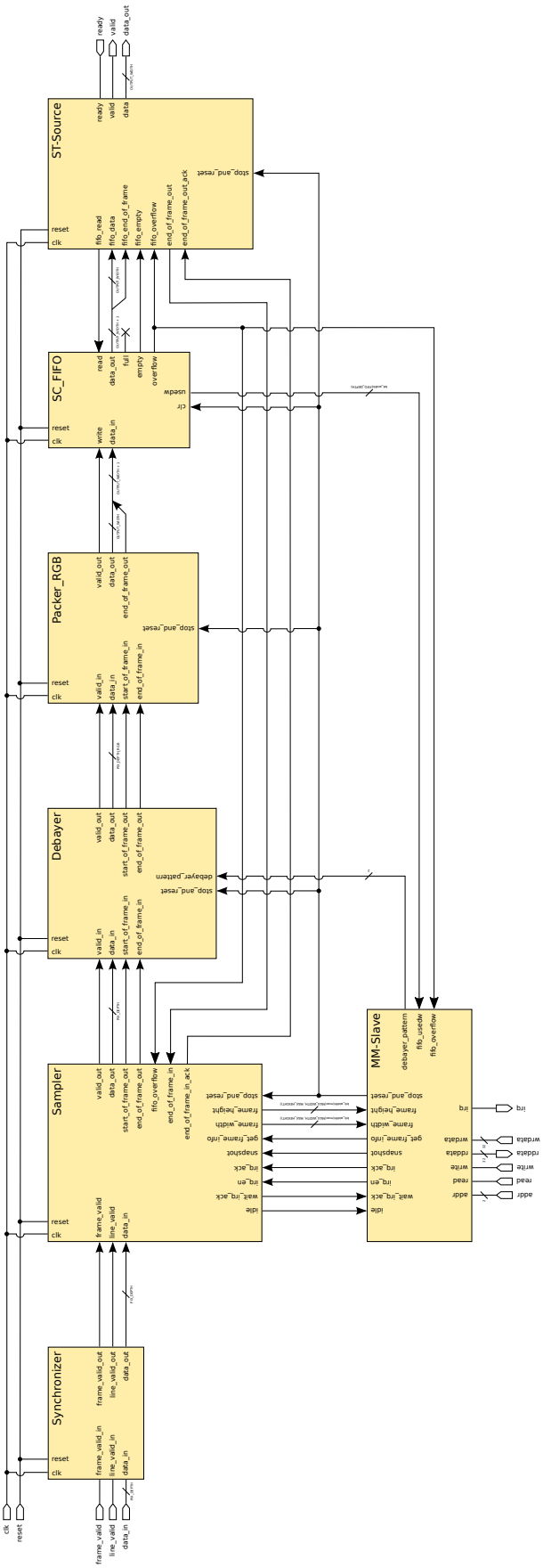

Figure 3: Internal block diagram.

4 Qsys Interface

Figure 4 shows the Qsys configuration interface for the core.

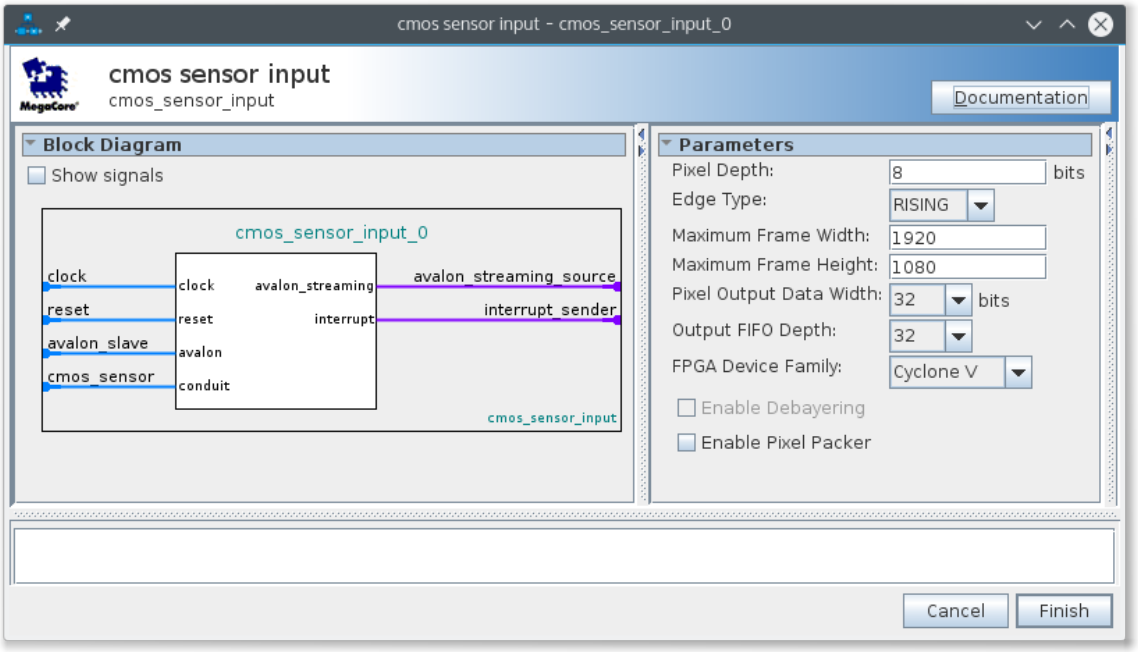

Figure 4: Qsys Configuration Interface.

It can be configured through 9 parameters, shown in Table 1.

Parameter	Type	Values	Default Value
PIX_DEPTH	Positive	1, 2, 3, ..., 32	8
SAMPLE_EDGE	String	"RISING", "FALLING"	"RISING"
MAX_WIDTH	Positive	2, 3, 4, ..., 65535	1920
MAX_HEIGHT	Positive	1, 2, 3, ..., 65535	1080
OUTPUT_WIDTH	Positive	8, 16, 32, ..., 1024	32
FIFO_DEPTH	Positive	8, 16, 32, ..., 1024	32
DEVICE_FAMILY	String	"Cyclone V", "Cyclone IV E"	"Cyclone V"
DEBAYER_ENABLE	Boolean	FALSE, TRUE	FALSE
PACKER_ENABLE	Boolean	FALSE, TRUE	FALSE

Table 1: Controller register map. All registers are 32 bits wide.

Some general notes about the possible parameter values:

- `PIX_DEPTH` is arbitraily restricted to go until 32 bits, but it can be increased without any design changes.
- `MAX_WIDTH` and `MAX_HEIGHT` are restricted to 65535 due to the Avalon-MM slave interface. The data width is 32 bits, and both the frame's current width and height take up 16 bits of a common `FRAME_INFO` register when read, so their maximum value is restricted to 65535. If the width of the Avalon-MM slave is increased, then the maximum width and height can take much larger values.
- `MAX_HEIGHT` can go down to as low as 1 row, however `MAX_WIDTH` can only go down to 2 colums. As such, the minimum capturable frame is of `2x1`. This restriction is in place to avoid the `start_of_frame` and `end_of_frame` signals used between the various components from overlapping. This also ensures that at least 2 pixels fit in a packed word (so the `packer` actually is useful).
- `OUTPUT_WIDTH` is the bit width of an Avalon-ST interface, and therefore must be a multiple of 8. The possible values are arbitrarily limited to powers of 2 instead to make the list of suggested values short in the Qsys GUI. If this requirement causes issues for your designs, you can modify the Qsys file describing the component to allow non-power of two values (as long as they remain multiples of 8).

- FIFO_DEPTH must be a power of two for technology reasons.
- DEVICE_FAMILY is needed to choose the appropriate implementation of the FIFO for the intended target device. Currently, this parameter only supports "Cyclone V" and "Cyclone IV E" as values. However, this choice was arbitary in the sense that they are the only devices on which the unit was tested. There is actually no restriction involved, and any other family should also work if you need to target another device.

5 Components

5.1 MM-Slave

Table 2 shows the register map exposed by the Avalon-MM slave interface.

Offset	Type	Name
0x00	RW	CONFIG
0x04	WO	COMMAND
0x08	RO	STATUS
0x0C	RO	FRAME_INFO

Table 2: Avalon-MM slave register map. All registers are 32 bits wide.

5.1.1 CONFIG register

The unit is configured through its CONFIG register, shown in Table 3.

Bit	Name	Value	Description
31:3	reserved	N/A	N/A
2:1	DEBAYER_PATTERN	0	RGGB
		1	BGGR
		2	GRBG
		3	GBRG
0	IRQ	0	Interrupt disable
		1	Interrupt enable

Table 3: CONFIG register definitions. Note that the CONFIG register can only be modified when the core is in the IDLE state.

If the IRQ bit is set, then interrupts are generated for successful GET_FRAME_INFO and SNAPSHOT commands, and upon FIFO overflows.

5.1.2 COMMAND register

Commands are submitted to the unit through its COMMAND register, shown in Table 4.

Name	Value	Description
GET_FRAME_INFO	0	Analyze a frame to determine its width and height
SNAPSHOT	1	Capture a frame
IRQ_ACK	2	Acknowledge interrupt
STOP_AND_RESET	3	Stop sampler and reset all other units

Table 4: COMMAND register definitions.

5.1.3 `STATUS` register

The unit's current state can be read through its `STATUS` register, shown in Table 5.

Bit	Name	Value	Description
31:13	reserved	N/A	N/A
12:2	FIFO_USEDW	0:FIFO_DEPTH	FIFO fill level
1	FIFO_OVERFLOW	0	NO_OVERFLOW
		1	OVERFLOW
0	STATE	0	Idle
		1	Busy

Table 5: `STATUS` register definitions.

5.1.4 `FRAME_INFO` register

Information about the analyzed frame (as observed by the unit when the `GET_FRAME_INFO` was submitted) can be read through the unit's `FRAME_INFO` register, shown in Table 6.

Bit	Name	Value	Description
31:16	FRAME_HEIGHT	1:MAX_HEIGHT	Observed frame height
15:0	FRAME_WIDTH	2:MAX_WIDTH	Observed frame width

Table 6: `FRAME_INFO` register definitions.

5.2 Sampler

The `sampler` is the most complicated component of the `cmos_sensor_input` core, as can be seen by its state machine diagram, shown in Figure 6.

The `sampler`'s main task is to transform the CMOS sensor's signals to the format shown in Figure 5.

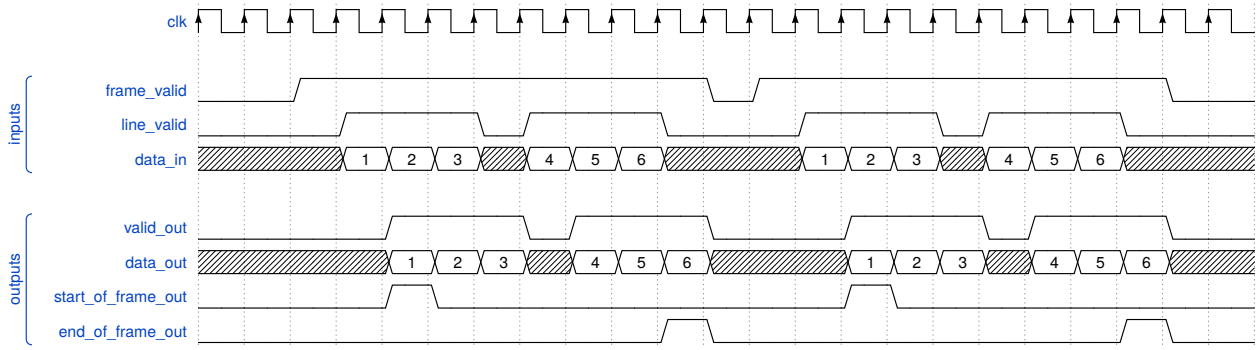

Figure 5: Sampler output data format.

Note that `start_of_frame` (respectively, `end_of_frame`) must be generated on the first (respectively, last) cycle where `data_out` is valid. It must *not* be generated before (respectively, after) the `valid` signal is asserted (respectively, deasserted).

In order to satisfy these timing constraints, the `sampler` needs to know the incoming frame width and height in advance. However, we want to avoid the user of the `cmos_sensor_input` core from having to manually program the frame dimensions through the Avalon-MM slave interface each time the core is used.

To get around this issue, we instead provide a `GET_FRAME_INFO` command. By submitting this command to the core, you instruct the `sampler` to analyze a frame flowing through it in order to automatically determine the frame dimensions. Note that the `sampler` does not perform any outputting when analyzing a frame. Outputting is only done upon reception of a `SNAPSHOT` command.

Note that the `sampler` stops immediately upon a FIFO overflow to allow the host to reconfigure the core.

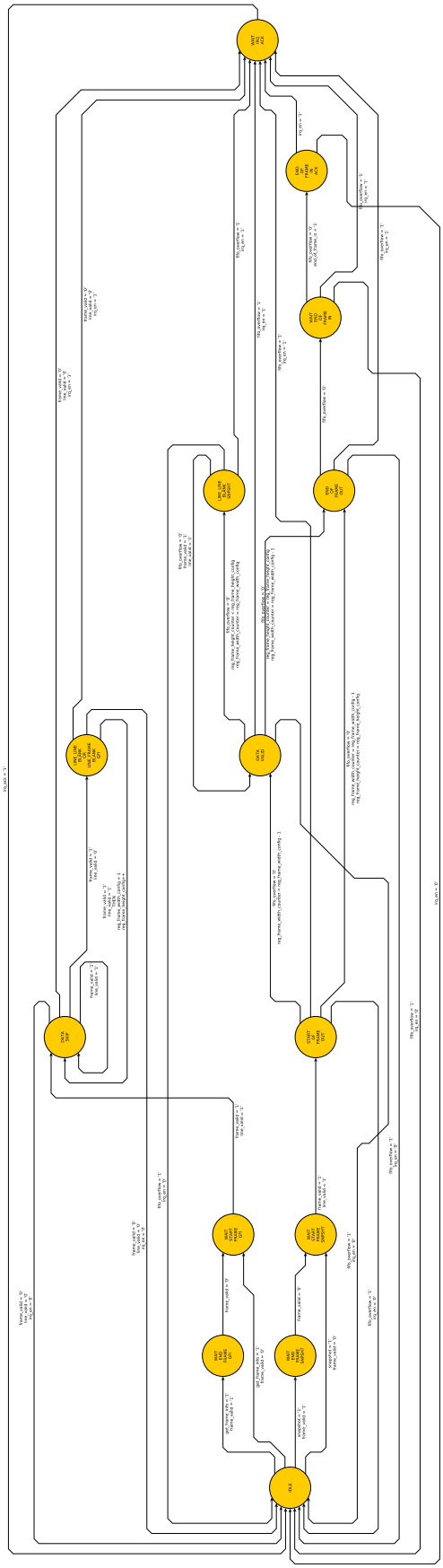

Figure 6: Sampler state machine. For legibility, the diagram does not show the outputs at each state, or the synchronous reset transition from all states back to the `IDLE` state upon receiving `stop_and_reset`.

5.3 Debayer

The `debayer` *unit is currently unimplemented. If enabled, it will simply copy its input to its output (appropriately resizing data to match the required bit widths). As such, please do not enable this option at this this time. This unit will be implemented in a future revision of the* `cmos_sensor_input` *core.*

5.4 Packer

The `packer` essentially consists of a shift-register. Incoming data is shifted in from the right until as many pixels that the data width supports are received. The remaining bits are filled with zeros. Figures 7 and 8 show the behaviour of the `packer` for different configurations.

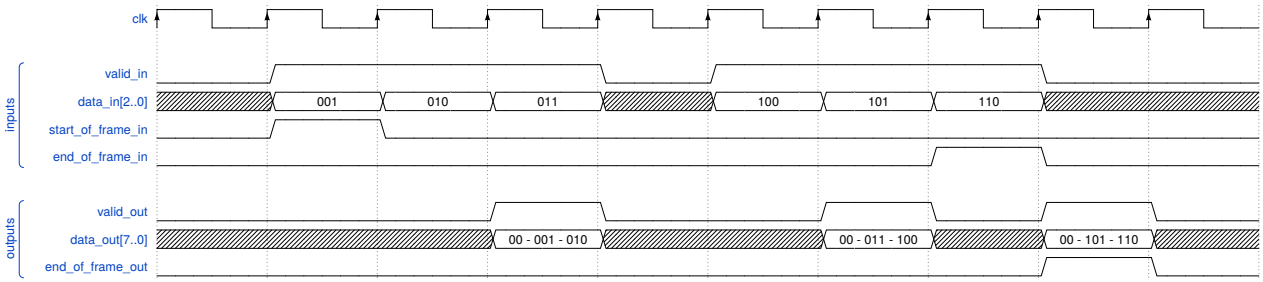

Figure 7: Packer waveform for two consecutive 3×2 frames (`PIX_DEPTH = 3`, `OUTPUT_WIDTH = 8`).

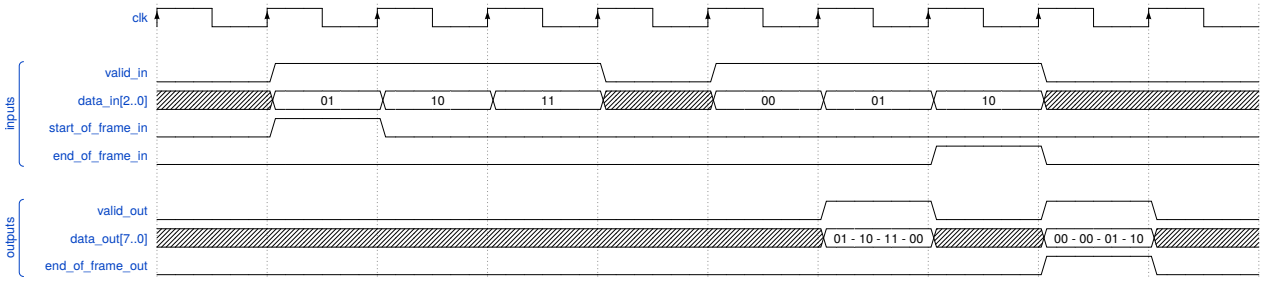

Figure 8: Packer waveform for two consecutive 3×2 frames (`PIX_DEPTH = 2`, `OUTPUT_WIDTH = 8`).

5.5 SC_FIFO

This component consists of a single-clocked FIFO that holds the output of the `cmos_sensor_input` core until it is sent out of the unit. It is actually a *wrapper* around a specific FIFO implementation depending on the device family the design is instantiated on.

In Figure 3, we see that the `empty` output is used downstream by the `ST-Source` component to know if the FIFO can be read. However, the `full` output is not used upstream by the `packer` to avoid writing to a full FIFO. This is not an issue, as write protection is implemented within the `sc_fifo` component itself, and external units do not need to test before writing. This is done to centralize write protection to a single place (since the core is modular and components can be moved around).

If the FIFO overflows, then you must submit a `STOP_AND_RESET` command to reinitialize the device.

6 Extensibility

The core is versatile: it is possible to add any additional filters needed for your application between the `sampler` and the `packer`. The only requirement is that *all* components placed between these two points use the same data format outputted by the `sampler` for their inputs *and* outputs. This is required so that different elements can be easily reordered and composed.

For example, one can add any convolutional filter after the `debayer` unit.