Pixel Remapper

T1130

Mentor: Bertl

Preeti Menghwani

Indian Institute of Technology Kanpur, Uttar Pradesh, India.
+91-7024877731
priti@iitk.ac.in
IRC Nick: preetimenghwani
Github: https://github.com/preetimenghwani

Contents

Overview

The Axiom Beta currently uses CMV12000 which is a high-speed CMOS image sensor with 4096 by 3072 pixels (22.5 mm x 16.9 mm). The image array consists of 5.5 µm x 5.5 µm pipelined global shutter pixels, which allow exposure during read-out. CMV12000 consists of 64 LVDS data output pins each pin is capable of transmitting 128 pixels in one burst. The image is stored in the pixel (global shutter) and they are read out sequentially, row-by-row.

Higher frame rates are achieved by using different modes like subsampling, Binning, Windowing. The pixels are located at different channels and are read out at a different moment in time. The end-user is able to remap the pixels on the outputs to their correct image array location with the help of mapping strategies used in different modes. Axiom Beta currently uses a remapper but it is very limited as it handles only 32sensel and 12bit (with 16bit padding) at the moment. My aim will be implementing a new remapper with low gate count, high throughput, which will be able to handle different bit depths of 12bit, 10bit, and 8bit and supporting two-sided as well as one-sided readout mode.

Goals

1. **Implement remapping strategies for known sensors (CMV12000)**
2. **Optimize for low gate count and high throughput**
3. **Simulate/Test the re-mapper with test-streams.**

These goals can be broken into the following subgoals:

1. Understanding and testing of current remapper

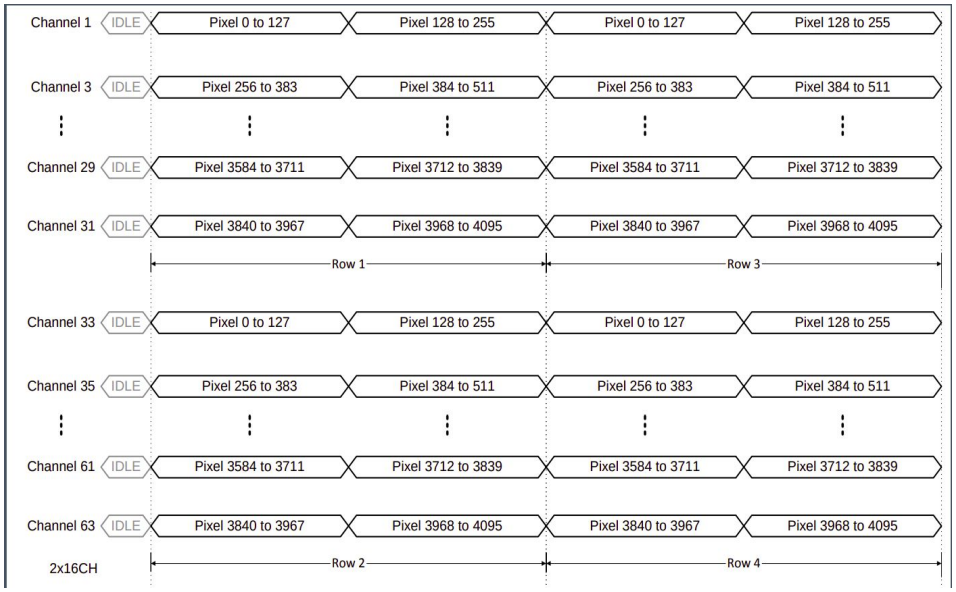
   The current remapper is not very flexible as it only supports 32sensel, 12bit mode at the moment. Using the current remapper as a reference will give a good start for my project. The current remapper uses buffer memory, writing pixels in memory and at the end of the line reads it in the correct order. There are 64 data output LVDS pins in total but the current remapper uses only 32 of them in a dual readout mode. By using fewer output pins more number of bursts are required for reading one row of pixels. The remapper only works for 12bit pixel size although the CMV12000 features pixel size of 8bit, 10bit as well. Using a larger pixel size reduces the frame rate.

   Reference:-https://github.com/apertus-open-source-cinema/axiom-beta-firmware/blob/master/peripherals/soc_main/pixel_remap.vhd

2. Recording pixels data in correct order in 32sensel 12bit mode

   My first aim will be building a new remapper which will have a similar function as the existing one so that I can compare simulation and design, this would help me in optimization, decreasing the gate count by partial separation i.e. using cycles efficiently.

   One of the 66 LVDS output pins gives out DDR clock pulses and the output data can be sampled on these pulses. In 32sensel 12bit mode the mapping of pixels is done in the following way:



Referring to the following diagram the lower we get the output of 32pixels at a time but these pixels are not adjacent thus rearrangement of pixels is required so to get an ordered array of pixels.
An overhead time exists between two bursts of 128 pixels. This overhead time has the length of one-pixel read-out (i.e. the length of 8, 10 or 12 bits at the selected data rate).
The Control LVDS output channel will be used for valid data synchronization and timing of output channels. The first three bits of the control channel output will be used for this purpose i.e DVAL, LVAL, FVAL.

3. Testing and optimization of 32sensel 12-bit mode

   Once the remapper is implemented and using simulations the model is optimized, I will test it either using a functional model of sensor or by testing it on Remote Beta.

4. Extending it for 8bit and 10bit mode

   Once the remapper works fine for 12bit mode I will extend it to 8bit and 10bit mode (i.e the output data array size would decrease). The mapping strategy in these modes is the same as that of 12bit mode so it would be easier to extend the model to these modes.
   Remapping in 10bit, 8bit mode will increase the frame rate.

5. Testing and validation

   Once the model works fine for 8bit, 10bit mode as well the model can be verified by making a functional model and obtaining simulations for the same.

6. Using all the output pins to decrease the number of bursts required

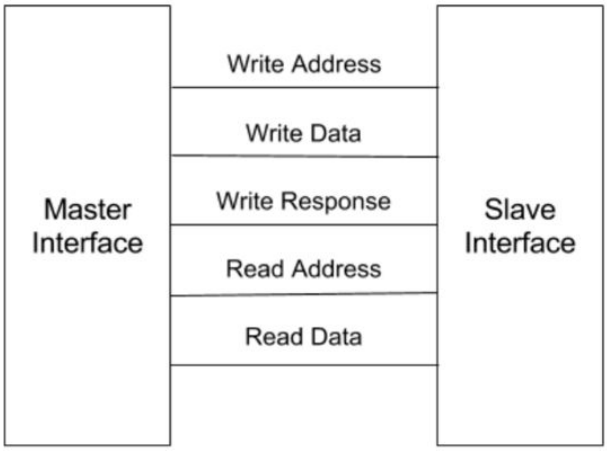
   After this, I will extend my model to use all 64 data output pins. While working in 32*2 mode the bottom pins output odd rows while the top pins output the even rows, each LVDS pin outputs 128 pixels in one burst which implies 2 rows are read in a single burst. Therefore with the knowledge of this mapping strategy, we can rearrange the output data of individual pins into an array in good order.
   Repacking and rearranging steps need to be performed as done above.

7. Repacking of data to write in RAM

The DDR memory is accessed via the AXI bus interface and to achieve maximum throughput, burst writes are required this means blocks of 64/128 bit words need to be sent which implies the sensel data needs to be repacked to fit into those 64/128 bit words.
So the data recorded needs to be reshuffled (qualification task would work as a good reference) into 64/128 bit words.

AXI Protocol Specifications

- Before transmission of any control signal/address/data, both master and slave must extend their “hand” for a handshake via ready and valid signals.
- Separate phases exist for the transmission of control signal/address and data.
- Separate channels exist for the transmission of control signal/address and data.
- Burst type communication allows for continuous transfer of data.



Channel connections between master and slave interfaces

8. Testing of data stored

Once the storage of repacked data in the DDR memory is done testing of the whole integrated block will be performed. First obtaining Simulation then testing on hardware. To verify that data is stored correctly I will use the ARM cores.

9. Remapper for Y-Subsampling mode

   I will build a new remapper that will work in Y subsampling mode the sensor skips a specific number of rows the image quality, in that case, reduces but less time and less power consumption are required to output the whole image array.

   The mapping, in this case, is quite different, rearranging and repacking will be done in the same way.

10. Testing of the whole integrated code

    This would be the last stage of the project, Simulation of the whole code will be obtained, Real-time testing will be done using the remote beta.

Future Goals

These goals can be achieved during GSoC projects if time permits otherwise I would love to implement these even after Summers!

Focussing on other modes of mapping

Cmv12000 also features other modes such as windowing, binning, etc

1. X and Y Subsampling

   With subsampling in both X and Y, the sensor skips both rows and columns. There are only 2 schemes possible in this mode: read 1 row/column, skip 1 row/column (monochrome) and read 2 rows/columns, skip 2 rows/columns (color). Once the Y subsampling is successful, the code can be extended to implement X and Y subsampling.

2. Windowing:

   This mode is used to limit the amount of data to increase the framerate of the sensor, it can only be done in the Y direction. The start address of the window and the number of lines can be specified by programming appropriate registers.
   The remapper that will be made during the project can be used for it just the size of the input array will decrease.

3. Binning:

   This mode is only supported in the two-sided read-out. To maintain the same field of view but reduce the noise coming out of the sensor, a binning mode is implemented on the chip. This mode will sum four pixels (in the analog domain) to reduce the noise and data coming from the chip.
   Based on the mapping strategy remapper for binning will be made.

Timeline

Dates	Task	Timeline Event
May 4 - June 1	1. Familiarize with existing codebase, and installation of required softwares. 2. Reading and understanding the datasheets of CMV12000, ZYNQ7000. 3. Reading about dual memory and getting familiar with the AXI interface.	**Community bonding period**
June 1 - June 7	1. Testing of current remapper and understanding design. 2. Start writing code for recording data in 32sensel 12bit mode.	**Coding officially begins**
June 8 - June 13	1. Working on code, optimization, fixing bugs 2. Obtaining simulations	
June 13 - June 20	1. Testing the new remapper through functional model 2. Testing using Remote beta.	
June 21 - June 28	1. Extending the remapper to 64sensel mode 2. Recording data according to the remapping strategy.	

June 29 - July 3	1. Documentation of work done till now. 2. Maintenance of all the code written.	**Phase 1 Evaluation**
July 4 - July 10	1. Testing of remapper in 64sensel mode. 2. Simulations, bug fixing, and optimization	
July 11 - July 16	1. Rearrangement and repacking of data to be written in RAM via the AXI bus. 2. Implementing the model and integrating it with the rest.	
July 17 - July 26	1. Simulation, bug fixing 2. Testing using ARM cores.	
July 27 - July 31	1. Documentation of work done till now 2. Fixing bugs and formatting the code.	**Phase 2 Evaluation**
July 31 - August 6	1. Start a new remapper for Y subsampling. 2. Recording data converting it into a good order.	
August 7 - August 13	1. Testing, simulation and bug fixing. 2. Real-time testing	

August 14 - August 20	1. Rearrangement and repacking of data for writing on RAM, bug fixing. 2. Testing using Remotely available hardware	
August 21 - August 31	1. Documentation of work 2. Code formatting, start working on sketch goals.	**Final Evaluation and Project Submission**

Qualification Task

https://github.com/preetimenghwani/Serdes-task1-T871

About Me

1. My CV

Here is the link to my resume:

https://github.com/preetimenghwani/Serdes-task1-T871/blob/master/Preeti_CV.pdf

2. What interests you most about the apertus° AXIOM project?

Apertus° AXIOM, an open hardware and free software digital cinema camera family of devices, in recent years has made significant progress. It would be my honor to contribute to the company by working for the project.

The qualification task was very challenging and it was a great learning experience. The task was so interesting that it kept me involved in it for hours. The shttps://github.com/preetimenghwani/Serdes-task1-T871/blob/master/Preeti_Resume_1.pdfkills that I would learn from this project would help me in the long run.

The mentors are really supportive and helpful, this would help me to the project with full enthusiasm and sincerity. As a whole, it would be a wonderful experience if I get selected for the project.

3. As mentors and project coordinators, how can we get the best out of you?

It would really boost my enthusiasm if my mentors ask me for daily updates, this would also help me in working sincerely.

Mentors are really helpful and supportive, I would be really glad to seek their help throughout the summer of code.

4. Is there anything that you'll be studying or working on whilst working alongside us?

No, I have nothing else planned apart from the GSoC project, I will devote my full time to the project. To be on a safe side I would like to start working on my task ten days before the actual coding begins.

5. Are there any techniques and tools which you use to keep yourself organized?

To keep myself organized I try to set goals on a daily basis that I can finish by the end of the day this makes the work easier and less hectic.

Taking small breaks after a few hours of work helps me in working more efficiently and with more focus.

I try to have a detailed roadmap of my work which helps me in planning things more effectively.