1. INTRODUCTION

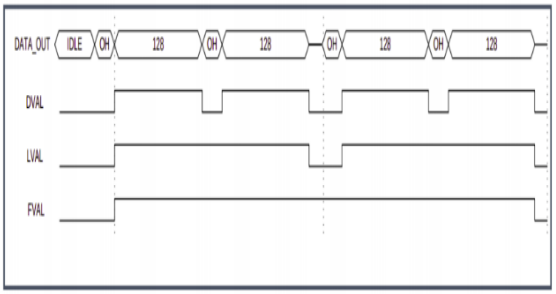
The task "Pixel Remapper" (T1130) aims at rearranging and repacking of the sensor data to provide a compact stream for memory writers, reducing the gate count and optimising it to give high throughput. This report summarizes the work done in the GSoC project with the Apertus Association.

2. TASKS IMPLEMENTED: FIRST PHASE

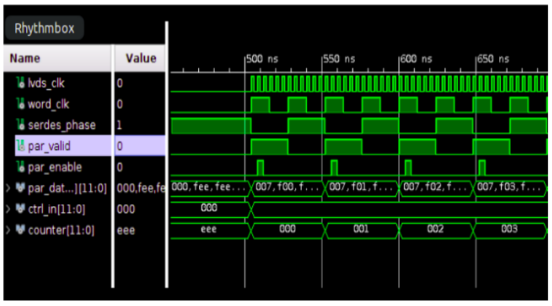
2.1 Designed Fake Data Generator

In order to test the pixel remapper on hardware without the use of CMV12000, a fake data generator is made which imitates the input of all the channels. The second week was mostly spent in designing this fake data generator. The module ser_to_par was modified in such a way that fake data of known pattern is generated which can be easily verified by dumping in memory over the primed pattern.

- The following figure shows the reference waveform and alignment on the basis of which ser_to_par module was modified to produce the fake data. This figure depicts the control signal timing with reference to the output sensels.



- The following figure shows the simulation of the fake data generator in which each channel is currently connected to counter.

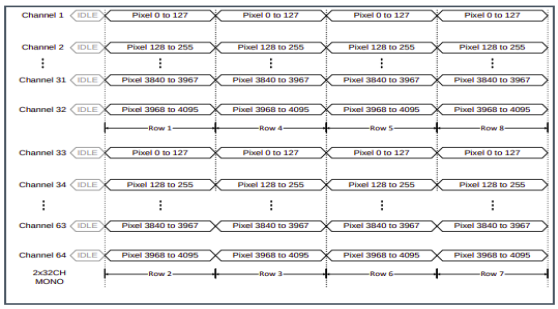


- The following figure shows the memory output where odd rows are represented by 1 in MSB of the hex value and even rows are represented by 0 in MSB of the hex. And in the current fake data, all the channels are connected with the counter which is represented by the remaining 3 LSB of the hex value. The data comes in packets of 64bits where the first 48bits representing 4 sensels and last 16bits are not written in the memory, hence showing the data with which the memory was primed initially (here it is "55555555").
- In this way any fake data can be assigned to the channels and the output can be verified on the basis of the expected pattern. Here is the link to repository (pixelremapdev) : https://github.com/preetimenghwani/axiom-firmware/tree/fake_data_pipeline
- The default settings for the registers were also added. Here is the link to the repository for the same: https://github.com/preetimenghwani/axiom-firmware/tree/reg_file_defaults

3. TASKS IMPLEMENTED: SECOND PHASE

3.1 Y subsampling mode

- The following figure shows the remapping style in case of Y subsampling.



- This means that in odd lval_counts first 32 channels give odd rows as output and next 32 channels give even rows as output.
- While in case of even lval_counts the first 32 channels give even rows as output and next 32 channels give odd rows as output.
- So when there is odd lval_count the remapping process is the same i.e the one which is followed currently while in case of even lval_counts the remap process is changed (i.e the first 32 channels are switched by the next 32 channels).
- Since the signal lval_out in pixel_remap.vhd represents the read out of a row so its count is maintained by a signal named lval_count_m which is 0 for odd count and 1 for even count.
- Here is the link to the patch: https://github.com/preetimenghwani/GSoC2020/blob/master/y_subsampling_mode.patch
- The following figure shows the output of Y Subsampling in 2x16 channels mode.

```
18000000: 0000018008015555 0020038028035555 0040058048055555 0060078068075555
18000020: 0080098088095555 00A00B80A80B5555 00C00D80C80D5555 00E00F80E80F5555
18000040: 0100118108115555 0120138128135555 0140158148155555 0160178168175555
18000060: 0180198188195555 01A01B81A81B5555 01C01D81C81D5555 01E01F81E81F5555
18000080: 0200218208215555 0220238228235555 0240258248255555 0260278268275555
180000A0: 0280298288295555 02A02B82A82B5555 02C02D82C82D5555 02E02F82E82F5555
180000C0: 0300318308315555 0320338328335555 0340358348355555 0360378368375555
180000E0: 0380398388395555 03A03B83A83B5555 03C03D83C83D5555 03E03F83E83F5555
18000100: 0400418408415555 0420438428435555 0440458448455555 0460478468475555
18000120: 0480498488495555 04A04B84A84B5555 04C04D84C84D5555 04E04F84E84F5555
18000140: 0500518508515555 0520538528535555 0540558548555555 0560578568575555
18000160: 0580598588595555 05A05B85A85B5555 05C05D85C85D5555 05E05F85E85F5555
18000180: 0600618608615555 0620638628635555 0640658648655555 0660678668675555
180001A0: 0680698688695555 06A06B86A86B5555 06C06D86C86D5555 06E06F86E86F5555
180001C0: 0700718708715555 0720738728735555 0740758748755555 0760778768775555
180001E0: 0780798788795555 07A07B87A87B5555 07C07D87C87D5555 07E07F87E87F5555
18000200: 0800818808815555 0820838828835555 0840858848855555 0860878868875555
18000220: 0880898888895555 08A08B88A88B5555 08C08D88C88D5555 08E08F88E88F5555
```

4. TASKS IMPLEMENTED: FINAL PHASE

4.1 Reduced Framework

- The reduced framework was made by only adding the required VHDL files in the firmware, the framework is the reduced version of the fake data generator created during the first phase.
- Changes were made to remove instantiations from the top module and directly connect the required instantiations.
- This was made as it made it easier for me to perform changes and testing as it took less time to build. The framework made it easy to integrate fatures with the full pipeline.
- Here is the link to the repository of Reduced Framework: https://github.com/preetimenghwani/GSoC2020/tree/master/soc_main

4.2 Address Generator

- The address generator that was used currently had some issues, for example: It did not give a proper offset for certain values of row increment. So a new address generator was made which solved this problem and helped to implement 64 channel mode.
- The address generator was made without DSP and the operations that were earlier performed by DSP were included in ccnt_proc.
- The address generator was tested and integrated with the main pipeline, it works till 220MHZ AXI clock.
- Here is the link to the repository: https://github.com/preetimenghwani/GSoC2020/blob/master/addr_gen.vhd

4.3 64 channel mode

- Currently the firmware supports only 32 channel mode for CMV12000, 64 sensel mode was added by using two writers by interleaving them.
- The interleaving is performed by using the base address as 0x18000000 for first writer and 0x18004000 for second writer and by giving a gap of one row after each writer writes.
- The patch was merged and tested for the main pipeline, both the writers worked properly but it cannot be tested for 64 sensel mode since currently the sensor doesn't support it.
- Link to the patch: https://github.com/preetimenghwani/GSoC2020/blob/master/64_channel.patch

4.4 Double throughput for 32 channel mode

- In order to support 500MHZ cmv clock (currently it supports 250MHZ) the a new VHDL file for fifo chop was made (fifo_chop16.vhd) which takes input from 16 channels so two fifo_chop16 were used and in this way there is essentially the throughput of from a single fifo_chop is same but since we have used two such so the overall throughput is doubled.
- Link to the patch: https://github.com/preetimenghwani/GSoC2020/blob/master/double_throughput.patch

4.5 Pixel Remapper Wrapper

- Support for both RG/GB and RGRGRGRG....GBGBGBGB format was added for the 32 channels mode.
- Link to the patch:https://github.com/preetimenghwani/GSoC2020/blob/master/RGGB_double_throughput.patch
- A new file named pixel_remap_wrap is made to easily select the mode i.e 32 channels, 64 channels, RGRGRGRG....GBGBGBGB format and RG/GB format, testing on full pipeline needs to be done.
- Link to the patch: https://github.com/preetimenghwani/GSoC2020/blob/master/pixel_remap_configurable.patch
- The patch has been tested using the test framework, the testing on remote hardware with full beta needs to be done.

5. CHALLENGES FACED

- Since the sensor was not connected to the beta it became necessary to build a test framework which required a lot of knowledge of Axiom Beta internals but this knowledge was eventually very useful.
- Faced problems with the address generator as it did not work as expected so it was necessary to build a new one.

6. LEARNING

- In order to work on the hardware i.e. remote Axiom Beta, my first step was to understand the Zynq-7000 architecture, on both PS and PL side.
- Read about AXI-DMA and different types of block RAM for basic understanding of memory addressing in RAM.
- Learned about the Axiom-Beta commands used for programming Zynq PL, and about the required tools.
- Learned about the current pixel remapper i.e the current remapping style and the way it is done, in order to make a reduced framework it was required to understand the relevant parts of the top module, to build a new fifo chop for doubling the thoughput understanding of fifo chop was required and also understood other modules which made my work easier.

7. FUTURE GOALS

- Once the address generator which supports 250MHZ clock is made the firmware will be extended to support 600MHZ CMV clock.
- Currently the firmware is not configurable for 8b, 10b it supports 12b bit depth, different architectures for fifo_chop need to made in order to make it configurable.
- Adding X-Y Subsampling mode, changes need to me done inside the pixel remapper to make it configurable for X-Y Subsampling mode.