A success story of leveraging SDSoCTM to accelerate customer software algorithm

Agenda:
- Corporate Profile
- Story Summary
- Story Detail
- MPSoC DEMO

Wataru Takahashi
Manager

OKI IDS
<table>
<thead>
<tr>
<th>Corporate name</th>
<th>OKI IDS Co., Ltd. (OIDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office address</td>
<td>3-1 Futaba-cho, Takasaki, Gunma-ken, Japan</td>
</tr>
<tr>
<td>Capital</td>
<td>50 million yen</td>
</tr>
<tr>
<td>Stockholder</td>
<td>OKI IDS Co., Ltd. (100% owned by Oki)</td>
</tr>
<tr>
<td>Founded</td>
<td>March 10, 2014</td>
</tr>
<tr>
<td>CEO</td>
<td>Noriaki Anada</td>
</tr>
<tr>
<td>Employees</td>
<td>78</td>
</tr>
<tr>
<td>Business description</td>
<td>Development, design, sales and consulting of firmware/hardware for multimedia (video, audio, communication) Product development for Medical instruments, Industrial instruments</td>
</tr>
</tbody>
</table>

The largest natural water flow of all hot springs in Japan
Kusatsu Hot Spring

Registered as Unesco World Heritage in 2014
Tomioka Silk Mill

Takasaki city is the top producer of Daruma Doll in Japan
Takasaki Daruma
OKI IDS strongly supports customer’s product design with highly-versatile solution products based the technologies of its specialty.
Goal of the project was to achieve moving object detection system on ZC706.

The system was designed to process input video from camera in real time, and output specific signal when a moving object is detected.

The customer already had entire algorithm developed on PC, and requested OKI to implement it onto ZC706.

In so doing, part of the processing needed to be hardware accelerated.

SDSoC worked efficiently for hardware acceleration of the system.

SDSoC enabled hardware acceleration of key processing of the system and greatly contributed to customer satisfaction.
The processing time for 1 frame at this point was:

400 ms for 1000 feature points  ->  2~3 fps
Corner Harris

(1) Open CV version Corner Harris Software

```c
void cornerHarris(const Mat& src, Mat& dst, int blockSize, int apertureSize, double k, int borderType=BORDER_DEFAULT)
```

(2) Vivado HLS version Corner Harris

```c
template<int blockSize, int Ksize, typename KT, int SRC_T, int DST_T, int ROWS, int COLS>
void CornerHarris(hls::Mat<ROWS,COLS,SRC_T>& _src, hls::Mat<ROWS,COLS,DST_T>& _dst, KT k);
```

(3) Auviz Corner Harris

```c
template<int ROWS, int COLS, int DEPTH, int NPC, int IN_WW, int OUT_WW, int FILTERSIZE, int MAXPNTS, int PIPELINEFLAG>
void auCornerHarris( hls::stream< AU_TNAME(IN_WW) >& _src_mat, hls::stream< AU_TNAME(OUT_WW) >& _dstlist, uint16_t img_height, uint16_t img_width, uint16_t_filter_width, uint16_t_block_width, uint16_t_nms_radius, uint16_t_threshold, uint16_t_val, uint32_t* nCorners)
```
OpenCV / VivadoHLS / Auviz

<table>
<thead>
<tr>
<th></th>
<th>OpenCV</th>
<th>VivadoHLS</th>
<th>Auviz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input format</strong></td>
<td>Software</td>
<td>Hardware</td>
<td>Hardware</td>
</tr>
<tr>
<td><strong>Output format</strong></td>
<td>Mat Format</td>
<td>Mat Format</td>
<td>Mat Format</td>
</tr>
<tr>
<td><strong>NonMaxSuppression</strong></td>
<td>Not Supported</td>
<td>Not Supported</td>
<td>Supported</td>
</tr>
<tr>
<td><strong>processing,</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>pick up feat.point</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The basic principles of Corner Harris

\[ S(u, v) = \sum w(x, y)[I(x+u, y+v) - I(x, y)]^2 \]
The basic principles of Corner Harris

\[ S( u, v ) = \sum_{xy} w( x, y )[ I( x+u, y+v ) - I(x, y) ]^2 \]

Take taylor expansion about \( I( x+u, y+v ) \).
Partial differentials of \( I \) are \( I_x, I_y \) then

\[ S( u, v ) \approx \sum_{xy} w( x, y )[ I_x(x,y)u + I_y(x,y)v ]^2 \]

\[ I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y} \]

Put it as matrix expression:

\[ S( u, v ) \approx (u \, v) A (u \, v)' \]

\[ A = \sum_{xy} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \]
The basic principles of Corner Harris

It can be determined by reviewing the eigenvalues, \( \lambda_1 \lambda_2 \). However the computational cost is high in this way. Instead, the following formula can be used to determine corner strength:

\[
M = \det(A) - k(\text{trace } A)^2
\]
SDSoC™

System Compiler
sds++, etc

Cross Compiler
aarch64-linux-gnu-g++
arm-linux-gnueabi-g++
armr5-none-eabi-g++
extc

SDSoC Library
libsds_lib.a
etc

Vivado

Vivado HLS
STEP 2

Making bitstream and interface library

ex)

feature_det.c

feature_det_sds.cpp

custom platform

SDSoC

System Compiler

Vivado

Vivado HLS

Cross Compiler

SDSoC Library

bitstream (boot.bin)

libfeature_det.so
STEP 2

Making application software

DMA and other interface lib

libfeature_det.so

application source files

application (elf file)

SDSoC

Cross Compiler

System Compiler
Source example

```
void feature_detector( unsigned short *image_in,
                        int *keypointlist,
                        int rows,
                        int cols,
                        int offset,
                        float kpara,
                        float threshold_f,
                        float threshold_f2,
                        int *keypointnum )
{
    myGoodFeature_sds( image_in,
                        keypointlist,
                        rows,
                        cols,
                        offset,
                        kpara,
                        threshold_f,
                        threshold_f2,
                        keypointnum );
}
```

```
#pragma SDS data_mem_attribute( image_in:PHYSICAL_CONTIGUOUS )
#pragma SDS data_mem_attribute( keypointlist:PHYSICAL_CONTIGUOUS )
#pragma SDS data_mem_attribute( keypointnum:PHYSICAL_CONTIGUOUS )
#pragma SDS data_access_pattern( image_in:SEQUENTIAL )
#pragma SDS data_access_pattern( keypointlist:SEQUENTIAL )
#pragma SDS data_access_pattern( keypointnum:SEQUENTIAL )
#pragma SDS data_mover( image_in:AXIDMA_SG )
#pragma SDS data_mover( keypointlist:AXIFIFO )
#pragma SDS data_mover( keypointnum:AXIFIFO )
#pragma SDS data_copy( image_in[0:WIDTH*HEIGHT] )

void myGoodFeature_sds( unsigned short image_in[ WIDTH * HEIGHT ],
                        int *keypointlist,
                        int rows,
                        int cols,
                        int offset,
                        float kpara,
                        float threshold_f,
                        float threshold_f2,
                        int *keypointnum )
{
    //
    //
    [
        HLS source code
    ]
    //
    //
SDSoC effects:
It's possible to make an environment that enables to compare the various results easier.

(1) Open CV  Corner Harris • • • • • Software

```c
void cornerHarris(const Mat& src, Mat& dst, int blockSize, int apertureSize, double k, int borderType=BORDER_DEFAULT)
```

(2) Vivado HLS  Corner Harris

```c
template<int blockSize,int Ksize,typename KT,int SRC_T,int DST_T,int ROWS,int COLS>
void CornerHarris(hls::Mat<ROWS,COLS,SRC_T> &_src, hls::Mat<ROWS,COLS,DST_T> &_dst, KT k);
```

(3) Auviz  Corner Harris

```c
template<int ROWS, int COLS, int DEPTH, int NPC, int IN_WW, int OUT_WW, int FILTERSIZE, int MAXPNTS,int PIPELINEFLAG>
void auCornerHarris( hls::stream < AU_TNAME(IN_WW) > & _src_mat, hls::stream < AU_TNAME(OUT_WW) > & _dstlist, uint16_t img_height, uint16_t img_width,uint16_t_filter_width, uint16_t_block_width, uint16_t_nms_radius,uint16_t_threshold, uint16_t val, uint32_t *nCorners)
```
● Installed the Corner Harris Detector to zcu102's PL
● Base design is Xilinx's TRD
Thank you for your attention!

OKI IDS