Synthesis of Digital Systems
Lab Lecture-4

Video Processing on ZedBoard
Hardware Implementation

Munish Jassi

Recap from Lab-C

Learnings from Lab-C

- Implementation of HDMI external interface via HDMI-out port.
- Implementation of Camera external interface using Pmod.
- Using Camera custom IP for Camera interface.
- Software implementation and using auto generated hardware drivers.

Content of the Lab Course

- System prototyping on FPGA and associated software tool chain.
- Industry standard High Level Synthesis (HLS) tool chain.
- Implementation of video processing hardware System and software interface.
- **Hardware Accelerator (HA) for performance improvement.**

OpenCV - Computer Vision Library

- Software library functions for real time Computer vision application (since 1999).
- Developed by Intel and now available under open source BSD license.
- Face recognition, HCI, motion detection/tracking, 2D/3D feature toolkit.
- Supported platforms: Linux, Win, Android, OS X and others.
- Interfaces to other languages like, Java, Python, Matlab.
- OpenCL for GPU interface.

Gray Scale Conversion – Software implementation

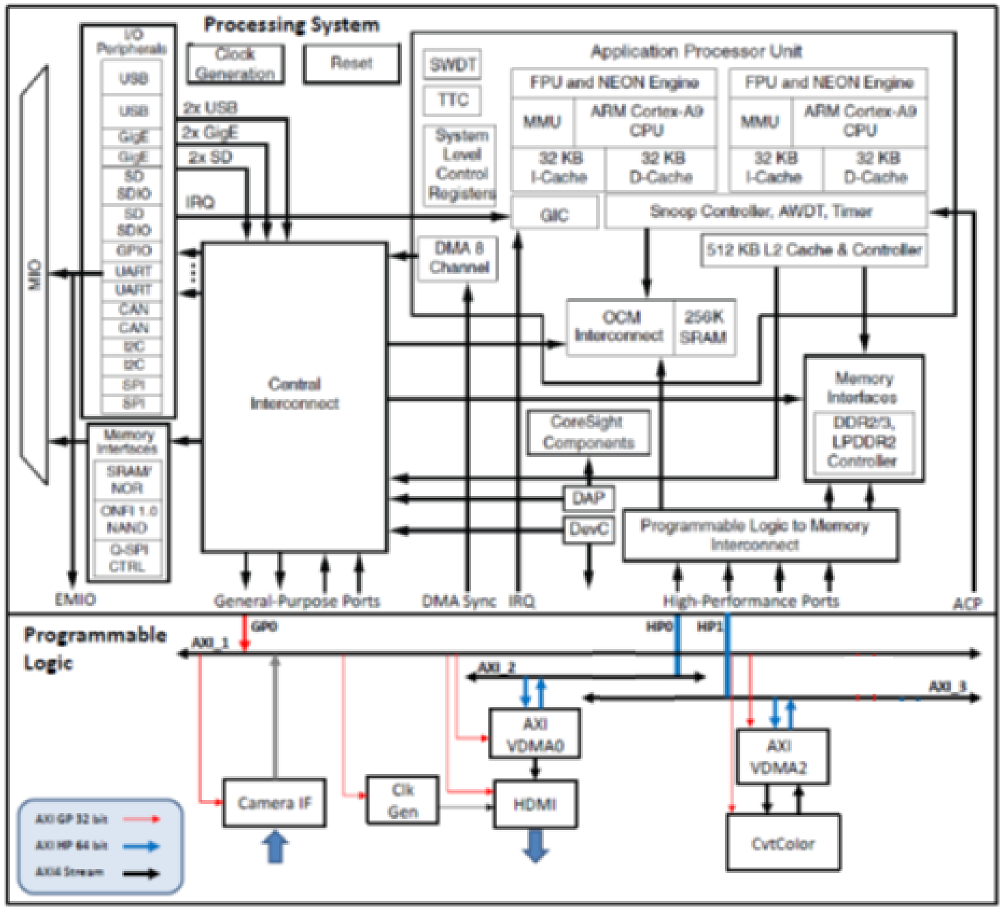
```
void ConvToGrayHLS(unsigned long ImgIn_BaseAddr,unsigned
long ImgOut_BaseAddr,unsigned short horizontalActiveTime)
{
        int i,j,V_offset;
        //kernel_CvtColor<CONVERSION>  kernel_opr;
        int     cols=640;
        int    rows=480;
        float  par[3] = {0.114,0.587,0.299};


        int _s; // src pixel
        int _d; // dst pixel
        for(i= 0; i < rows; i++) {
                V_offset = i * horizontalActiveTime;
                for (j= 0; j < cols; j++) {
                        _s = Xil_In32(ImgIn_BaseAddr + (V_offset + j) * 4);
                //Get the colored image pixel
                        kernel_apply(&_s,&_d, par);
                        Xil_Out32( ImgOut_BaseAddr + ((V_offset+j) * 4) ,
                _d ); // write to another locationgray pixel
} } }
```

```
void kernel_apply(int* _src, int* _dst,float _par[3])
{
        int b,g,r;
        r=_par[0]*( *_src & 0xff ); //_src.val[2];
        b=_par[1]*((*_src & 0xff00 ) >> 8); //_src.val[1];
        g=_par[2]*((*_src & 0xff0000) >> 16); //_src.val[0];
        int c;
        c=r+g+b;
        if(c>255)
        {
        c=255;
        }
        *_dst = (c & 0xff) | ((c & 0xff)<<8) | ((c & 0xff)<<16) ;

}
```

Gray Scale Conversion – Hardware implementation

```
class kernel_CvtColor<HLS_BGR2GRAY> {
public:
  template<typename SRC_T, typename DST_T,typename PAR_T, int CN1,int CN2,int CN3>
  void apply(Scalar<CN1,SRC_T>& _src, Scalar<CN2,DST_T>& _dst,Scalar<CN3,PAR_T>_par) {
#pragma HLS inline
    ap_fixed<32,11,AP_RND> b,g,r;
    r=_par.val[0]*_src.val[0];
    b=_par.val[1]*_src.val[1];
    g=_par.val[2]*_src.val[2];
    ap_fixed<12,12,AP_RND> c;
    c=r+g+b;
    if(c>255)
    {
      c=255;
    }
      _dst.val[0]=c;
  }
};
```

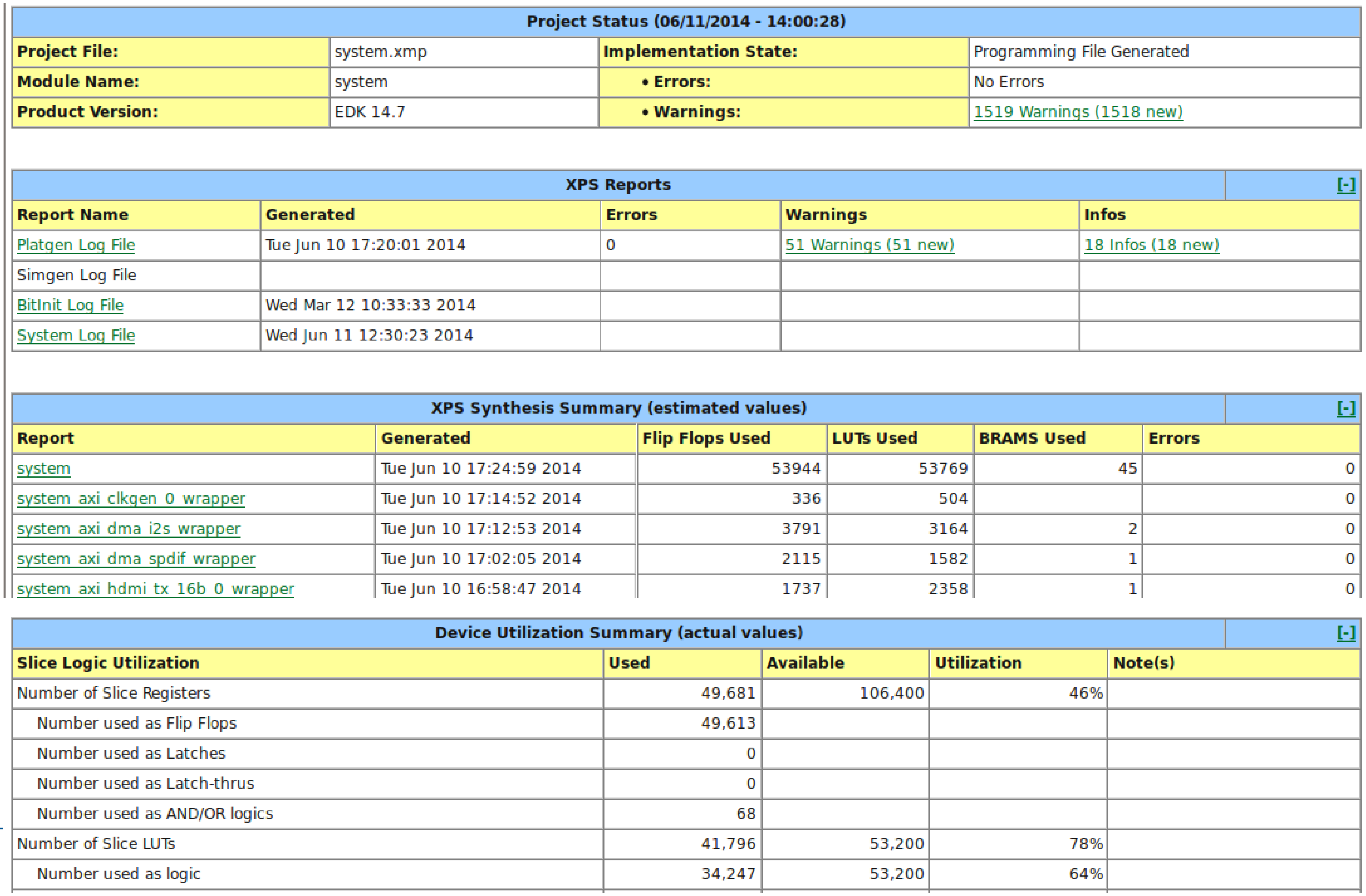
Gray Scale Application – Hardware acceleration



Steps for Lab-D

1. Vivado project set-up, HLS of Gray scale C-code and IP-generation.
2. IP-integration of filter IP to the system.
3. Software implementation for hardware accelerated system.
4. Application profiling using CPU cycle count register.

Performance Analysis: Area Utilization



Project Status (06/11/2014 - 14:00:28)			
Project File:	system.xmp	**Implementation State:**	Programming File Generated
Module Name:	system	• **Errors:**	No Errors
Product Version:	EDK 14.7	• **Warnings:**	1519 Warnings (1518 new)

XPS Reports				[-]
Report Name	**Generated**	**Errors**	**Warnings**	**Infos**
Platgen Log File	Tue Jun 10 17:20:01 2014	0	51 Warnings (51 new)	18 Infos (18 new)
Simgen Log File				
BitInit Log File	Wed Mar 12 10:33:33 2014			
System Log File	Wed Jun 11 12:30:23 2014			

XPS Synthesis Summary (estimated values)					[-]
Report	**Generated**	**Flip Flops Used**	**LUTs Used**	**BRAMS Used**	**Errors**
system	Tue Jun 10 17:24:59 2014	53944	53769	45	0
system_axi_clkgen_0_wrapper	Tue Jun 10 17:14:52 2014	336	504		0
system_axi_dma_i2s_wrapper	Tue Jun 10 17:12:53 2014	3791	3164	2	0
system_axi_dma_spdif_wrapper	Tue Jun 10 17:02:05 2014	2115	1582	1	0
system_axi_hdmi_tx_16b_0_wrapper	Tue Jun 10 16:58:47 2014	1737	2358	1	0

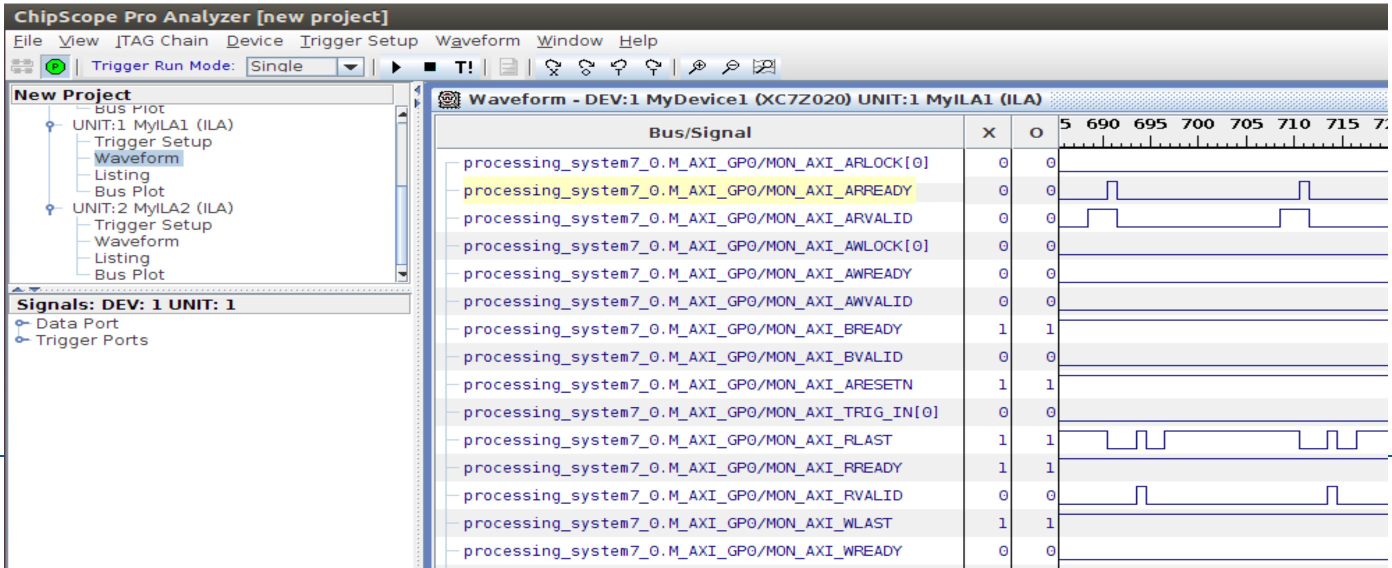
Device Utilization Summary (actual values)				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	49,681	106,400	46%	
Number used as Flip Flops	49,613			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	68			
Number of Slice LUTs	41,796	53,200	78%	
Number used as logic	34,247	53,200	64%	

Performance Analysis: CPU Load Analysis

- Cortex-A9 has PMU with 6 counters which acknowledge 58 events.
- Cortex-A9 reference manual Rev.: r4p1.
- Custom functions available in profile_cnt.h
- Available profiling functions ,
  - EnablePerfCounters()
  - init_perfcounter(do_reset, enable_divider)
  - get_cyclecount()
- These functions uses Assembly instructions to access the CPU counter registers to get the CPU cycles.

Performance Analysis: Bus Load Analysis

- ChipScope AXI Monitor:
  - Probing of any signal on the AXI interconnect
  - Multiple bus monitor support
  - ChipScope tool for graphical interface
  - User can define the Sample size before Synthesis.
  - User can sample the values either via setting the Interrupt or trigger it manually on ChipScope.
  - **Area- LUT: 729, FF: 1141, BRAMs:9**



Bus/Signal	X	O
processing_system7_0.M_AXI_GP0/MON_AXI_ARLOCK[0]	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_ARREADY	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_ARVALID	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_AWLOCK[0]	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_AWREADY	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_AWVALID	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_BREADY	1	1
processing_system7_0.M_AXI_GP0/MON_AXI_BVALID	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_ARESETN	1	1
processing_system7_0.M_AXI_GP0/MON_AXI_TRIG_IN[0]	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_RLAST	1	1
processing_system7_0.M_AXI_GP0/MON_AXI_RREADY	1	1
processing_system7_0.M_AXI_GP0/MON_AXI_RVALID	0	0
processing_system7_0.M_AXI_GP0/MON_AXI_WLAST	1	1
processing_system7_0.M_AXI_GP0/MON_AXI_WREADY	0	0

Performance Analysis: Bus Load Analysis

- AXI Performance Monitor:
  - Performance metrics monitoring such as Write/Read Bytes/Transactions.
  - Bus Write/Read Latency/Ideal clock cycles.
  - Multiple monitor support
  - Register-based performance values access
  - Area required:
    - **1 monitor and 1 counter: LUT: 992, FF: 1221**
    - **1 monitor and 10 counters: LUT: 1567, FF: 4561**

Xilinx EDK flow: Scripted approach - XPS

```
$ xmd
```

runxps.tcl

```
xps -nw -scr mhs_update.tcl
make -f $SYS.make bits
make -f $SYS.make exporttosdk

cp $PRJDIR/implementation/$SYS.bit
$PRJDIR/SDK/SDK_ Export/hw
```

mhs_update.tcl

```
xload xmp $env(XMP)
xload mhs $env(MHS)
run resync
save proj
exit
```

Xilinx EDK flow: Scripted approach - SDK

runsdk.tcl

```
appguru  -hw $PRJDIR/SDK/SDK_Export/hw/system.xml  -app empty_application
 -pe ps7_cortexa9_0 \
 -lp /nfs/tools/xilinx/ise/14.7/ISE_DS/ISE/data/zynqconfig/ps7_internals/pcores \
 -stdin ps7_uart_1  -stdout ps7_uart_1  -od  $BSPDIR  -lp ps7_cortexa9_0

libgen  -hw $PRJDIR/SDK/SDK_Export/hw/system.xml -pe ps7_cortexa9_0 \
 -log libgen.log  -mhs $PRJDIR/$MHS -lp $APPDIR/lib  -lp $PRJDIR/pcores  \
 -p xc7z020clg484-1  system.mss

cp $BSPDIR/lscript.ld $APPDIR/src
cd $APPDIR/Debug
make clean
make all
```

Xilinx EDK flow: Scripted approach – FPGA Programming

progfpga.tcl

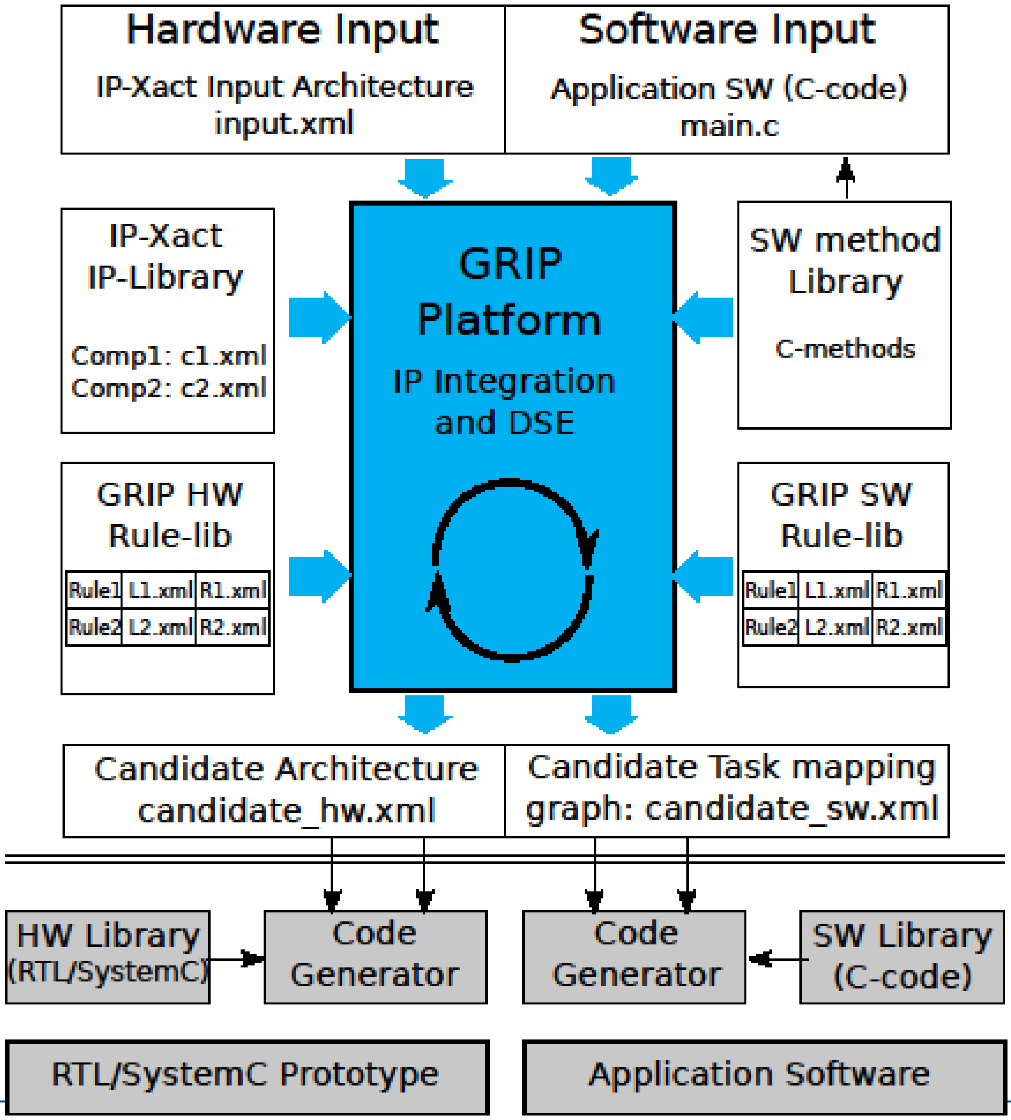
```
set bitFile $env(HWPLAT)/$env(SYS).bit
set tclFile $env(HWPLAT)/ps7_init.tcl
set elfFile $env(APPDIR)/Debug/$env(APPNAME).elf

fpga -debugdevice devicenr 2 -f $bitFile
connect arm hw
rst -processor
source $tclFile
ps7_init
init_user
dow $elfFile
run
exit
```

Research directions at EDA – System Level Automation

- Challenge is to formalize the method of IP-integration.
- Industry standard in/out formats for System description must be employed (IP-Xact).
- Designer’s knowledge about step-by-step integration of IP must be encoded.
- Configurability of parameterizable IP-core to be considered.
- Associated interconnections for buses, Interrupt, Control, and Synchronization signals.
- Code generation for translating IP-Xact to tool specific configuration files.
- Performance analysis.

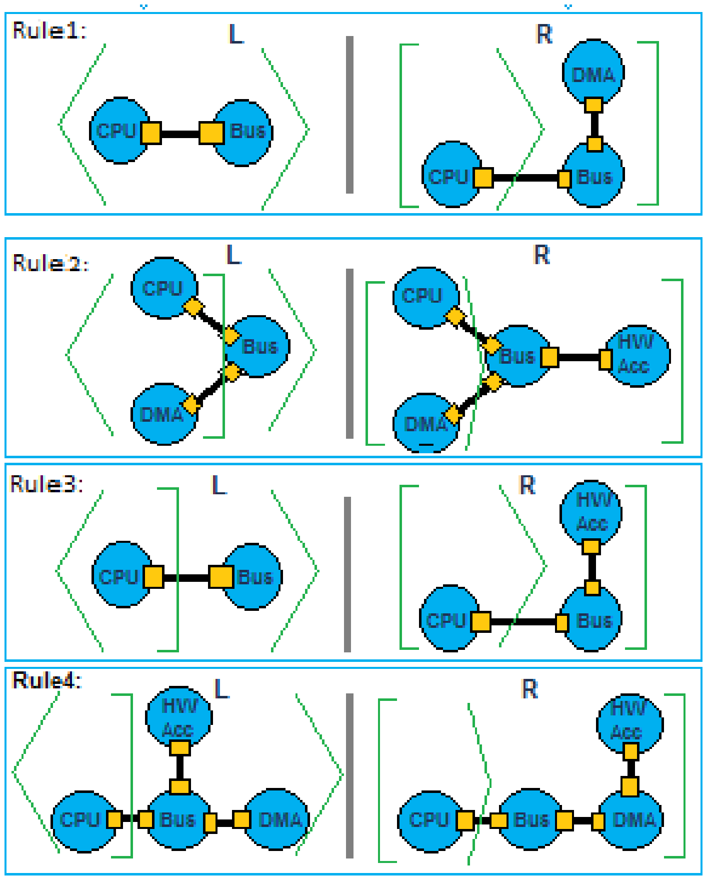
Target GRIP platform

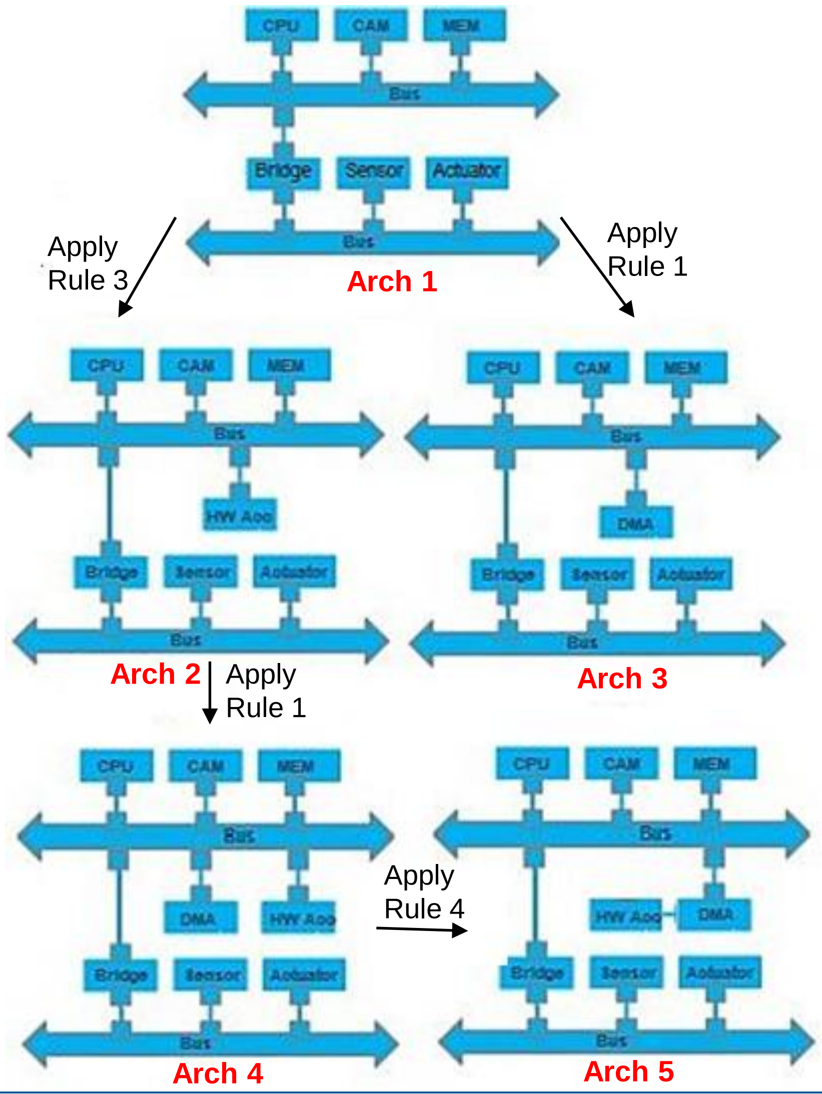
- Two inputs from IP provider
  - Component-Library (IP-Xact)
  - GRIP Rule-library
- Input from SoC developer
  - Initial SoC design (IP-Xact)
- Inputs are IP-xact based for easy integration with other EDA tools
- GRIP platform (EMF):
  - Outputs improved SoC design
  - Code generation for implementation or simulation



Design Space Exploration

- GRIP design rules





Acknowledgements

- Dr. –Ing Daniel Müller-Gritschneder
- M.Sc. Umair Razzaq
- B.Sc. Benjamin Bordes

Thank You

for your participation

And this is the end of Lab part for SDS.