How to kickstart App Development using Xilinx Ultrascale+ RFSoC ZCU216 Eval Platform

seeteck@gmail.com

Content:

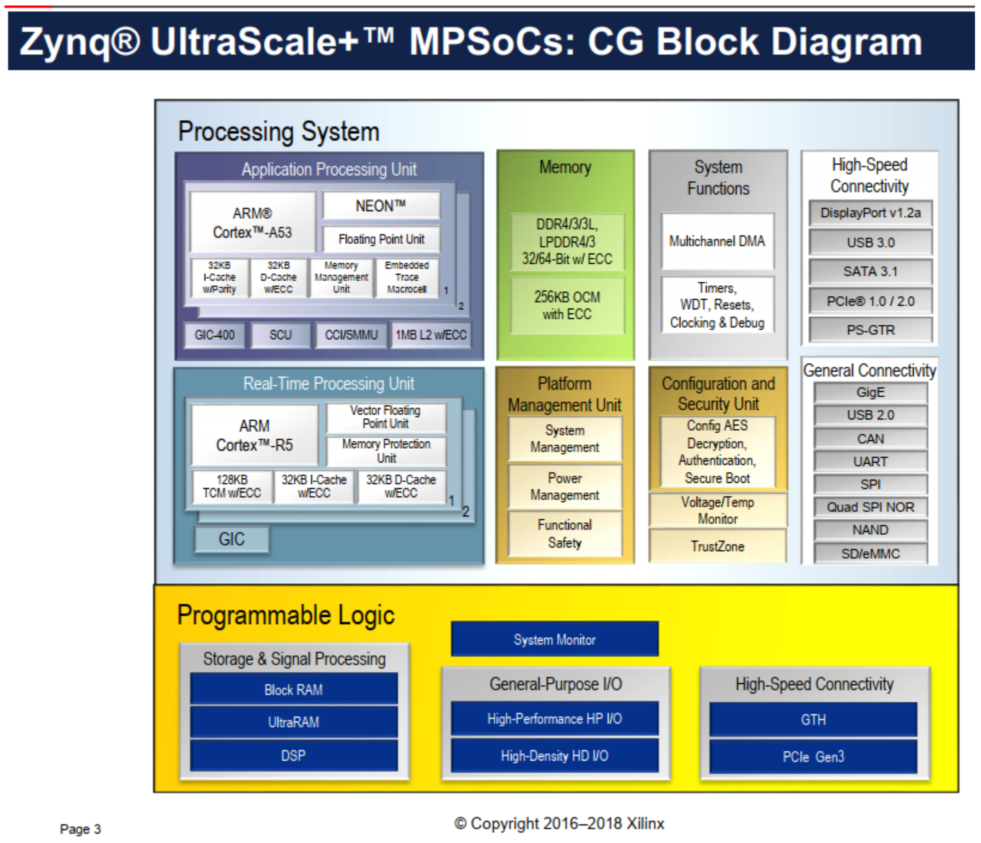
- ❑Skillset needed
- ❑Xilinx RFSoC
- ❑Tools & Reference Designs
- ❑Brief Intro to Reference Design
- ❑Demo: Vivado and Vitis Project Directory

Slide + Demon (if time allowed) will take about 50 min

Skillset - Roughly 5 Categories:

- ❑ Architecture (FPGA, Prototype Platform)
- ❑ HDL (hardware description language)
- ❑ FPGA Design Flow (Integration of IP, Simulation, Synthesis, Implementation, Verification)
- ❑ Embedded C
- ❑ Compiler and IDE (Tool Chain)

Architecture of FPGA

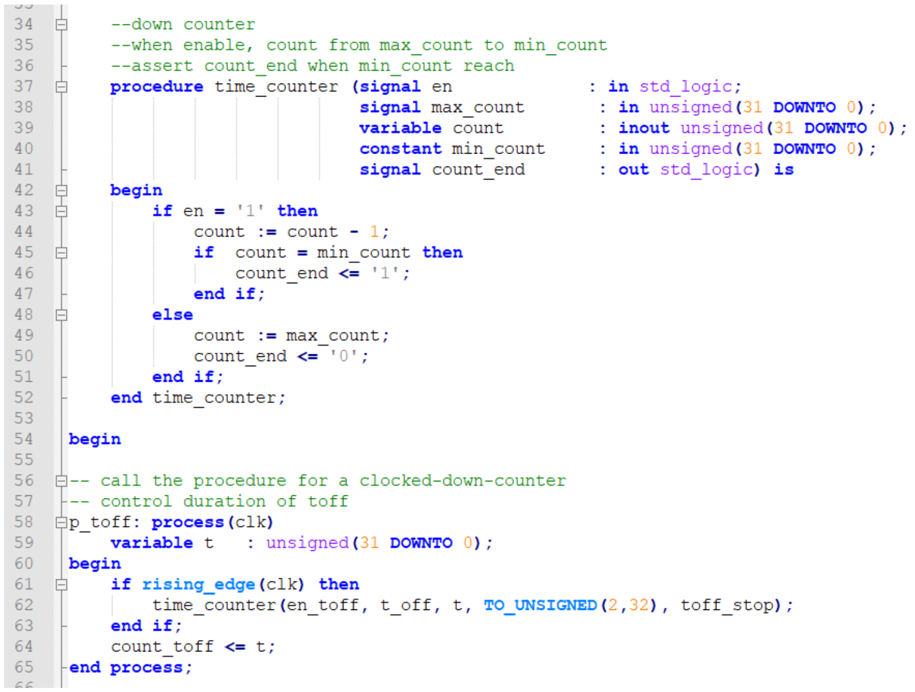


- ❑ Basic understanding of Processing System such as CPU, APU, RPU, MMU, Cache, DMA, DDR and etc.
- ❑ Basic understanding of commonly used connectivity such as UART, USB, I2C, SPI, ethernet and etc.
- ❑ Understanding of resources of Programmable Logic such as BRAM, URAM, DSP, GPIO, LUT, SLICE, CLB, IO standard and etc.
- ❑ Understanding of IP blocks such as FFT, FIR, DDS, NCO, PLL, Memory Controller and etc.

- ❑ RFSoC = MPSoC + Data Converters

HDL

- Either VHDL or SystemVerilog

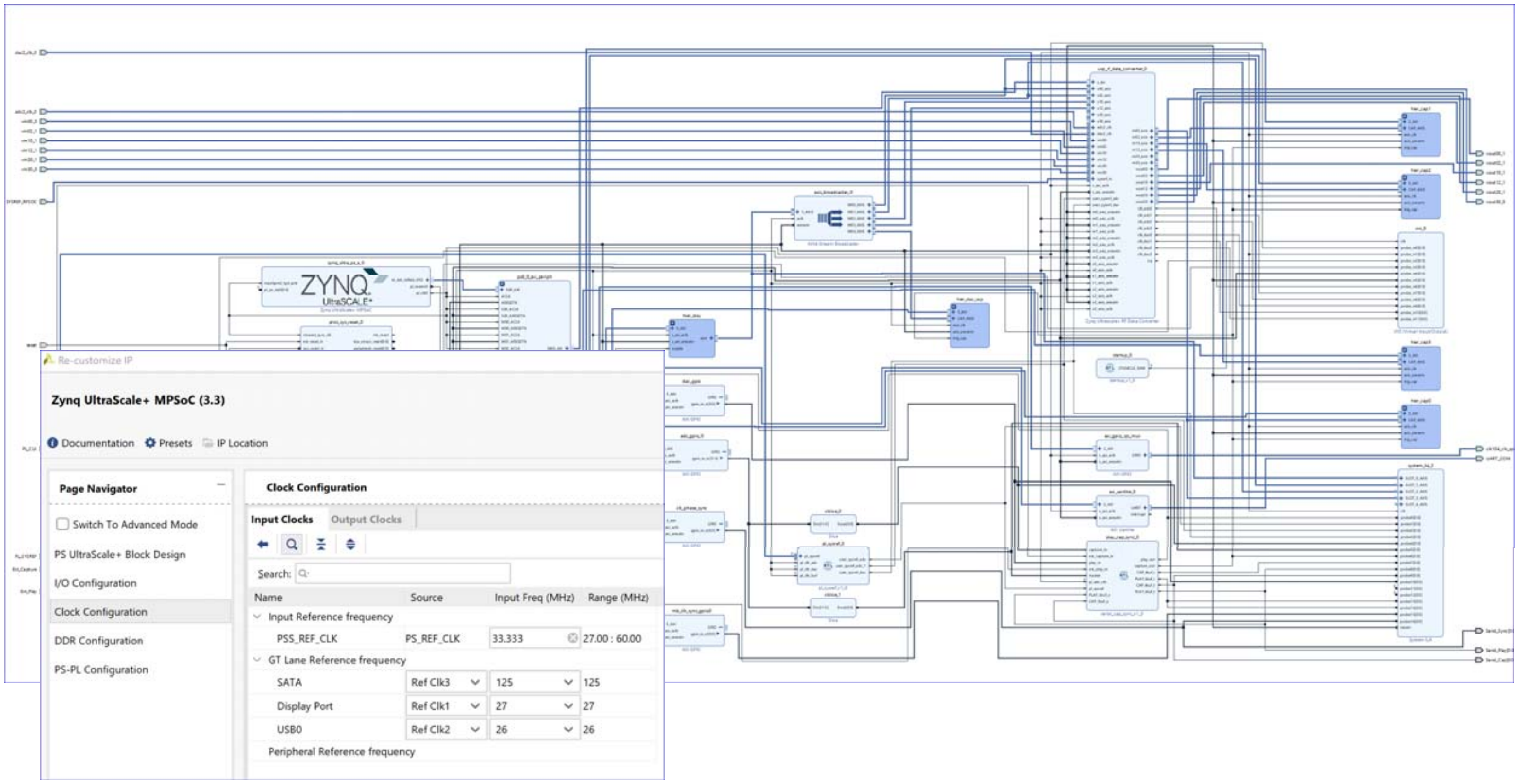


```vhdl
    --down counter
    --when enable, count from max_count to min_count
    --assert count_end when min_count reach
    procedure time_counter (signal en             : in std_logic;
                            signal max_count       : in unsigned(31 DOWNTO 0);
                            variable count         : inout unsigned(31 DOWNTO 0);
                            constant min_count     : in unsigned(31 DOWNTO 0);
                            signal count_end       : out std_logic) is
    begin
        if en = '1' then
            count := count - 1;
            if  count = min_count then
                count_end <= '1';
            end if;
        else
            count := max_count;
            count_end <= '0';
        end if;
    end time_counter;

begin

-- call the procedure for a clocked-down-counter
-- control duration of toff
p_toff: process(clk)
    variable t   : unsigned(31 DOWNTO 0);
begin
    if rising_edge(clk) then
        time_counter(en_toff, t_off, t, TO_UNSIGNED(2,32), toff_stop);
    end if;
    count_toff <= t;
end process;
```

- ❑ Able to write/read with 1 language (writing is always difficult than reading)
- ❑ Able to read with another language. Sometimes you need to understand, adapt and modify code that is written in another language
- ❑ Different level of mastery of language will impact design quality, upgradability and maintainability of code
- ❑ Undetected bug could be as simple as harmless and no body will ever detect it or disaster waiting to be happen

Design Flow – IP Integration

- Able to configure each IP Core and Connect them together
- IP-reuse is essential part of fpga design



Re-customize IP

Zynq UltraScale+ MPSoC (3.3)

Documentation · Presets · IP Location

Page Navigator

- Switch To Advanced Mode
- PS UltraScale+ Block Design
- I/O Configuration
- Clock Configuration
- DDR Configuration
- PS-PL Configuration

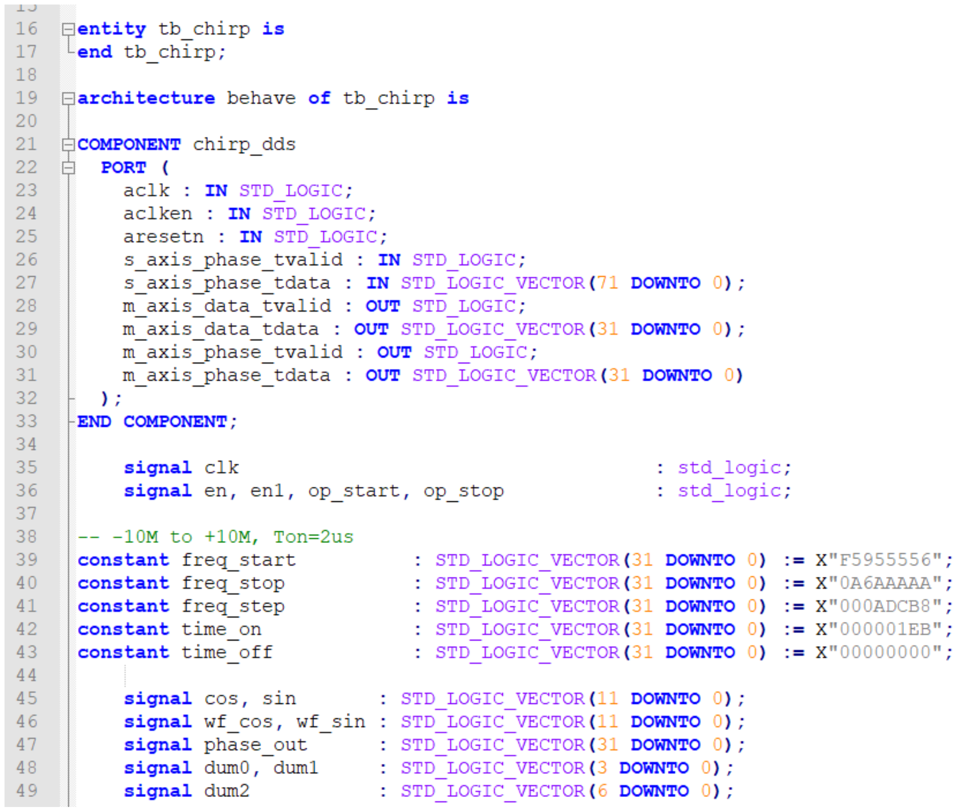
Clock Configuration

Input Clocks | Output Clocks

Search:

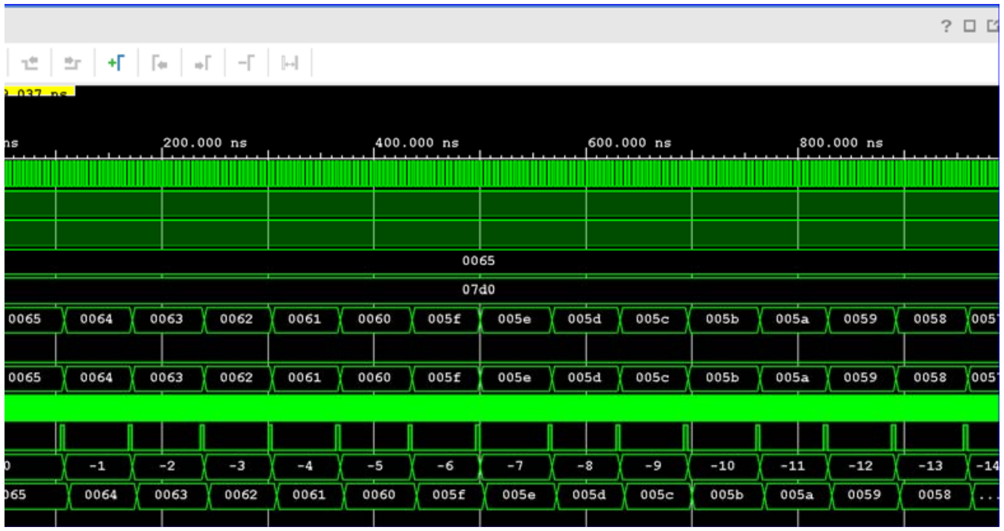
Name	Source	Input Freq (MHz)	Range (MHz)
Input Reference frequency			
PSS_REF_CLK	PS_REF_CLK	33.333	27.00 : 60.00
GT Lane Reference frequency			
SATA	Ref Clk3	125	125
Display Port	Ref Clk1	27	27
USB0	Ref Clk2	26	26
Peripheral Reference frequency			

Design Flow – Simulation



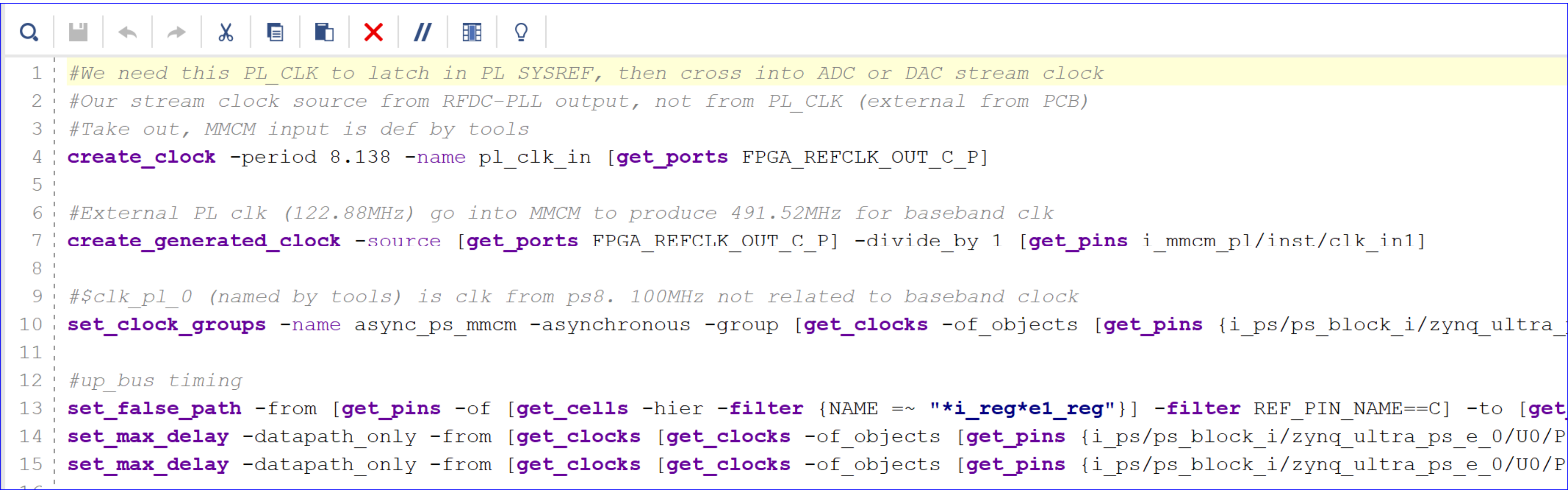
```vhdl
entity tb_chirp is
end tb_chirp;

architecture behave of tb_chirp is

COMPONENT chirp_dds
  PORT (
    aclk : IN STD_LOGIC;
    aclken : IN STD_LOGIC;
    aresetn : IN STD_LOGIC;
    s_axis_phase_tvalid : IN STD_LOGIC;
    s_axis_phase_tdata : IN STD_LOGIC_VECTOR(71 DOWNTO 0);
    m_axis_data_tvalid : OUT STD_LOGIC;
    m_axis_data_tdata : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
    m_axis_phase_tvalid : OUT STD_LOGIC;
    m_axis_phase_tdata : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;

    signal clk                                       : std_logic;
    signal en, en1, op_start, op_stop                : std_logic;

-- -10M to +10M, Ton=2us
constant freq_start          : STD_LOGIC_VECTOR(31 DOWNTO 0)  := X"F5955556";
constant freq_stop           : STD_LOGIC_VECTOR(31 DOWNTO 0)  := X"0A6AAAAA";
constant freq_step           : STD_LOGIC_VECTOR(31 DOWNTO 0)  := X"000ADCB8";
constant time_on             : STD_LOGIC_VECTOR(31 DOWNTO 0)  := X"000001EB";
constant time_off            : STD_LOGIC_VECTOR(31 DOWNTO 0)  := X"00000000";

    signal cos, sin        : STD_LOGIC_VECTOR(11 DOWNTO 0);
    signal wf_cos, wf_sin : STD_LOGIC_VECTOR(11 DOWNTO 0);
    signal phase_out       : STD_LOGIC_VECTOR(31 DOWNTO 0);
    signal dum0, dum1      : STD_LOGIC_VECTOR(3 DOWNTO 0);
    signal dum2            : STD_LOGIC_VECTOR(6 DOWNTO 0);
```

- ❑ Able to write good test bench to fully verify design, especially when designer and verifier are the same person



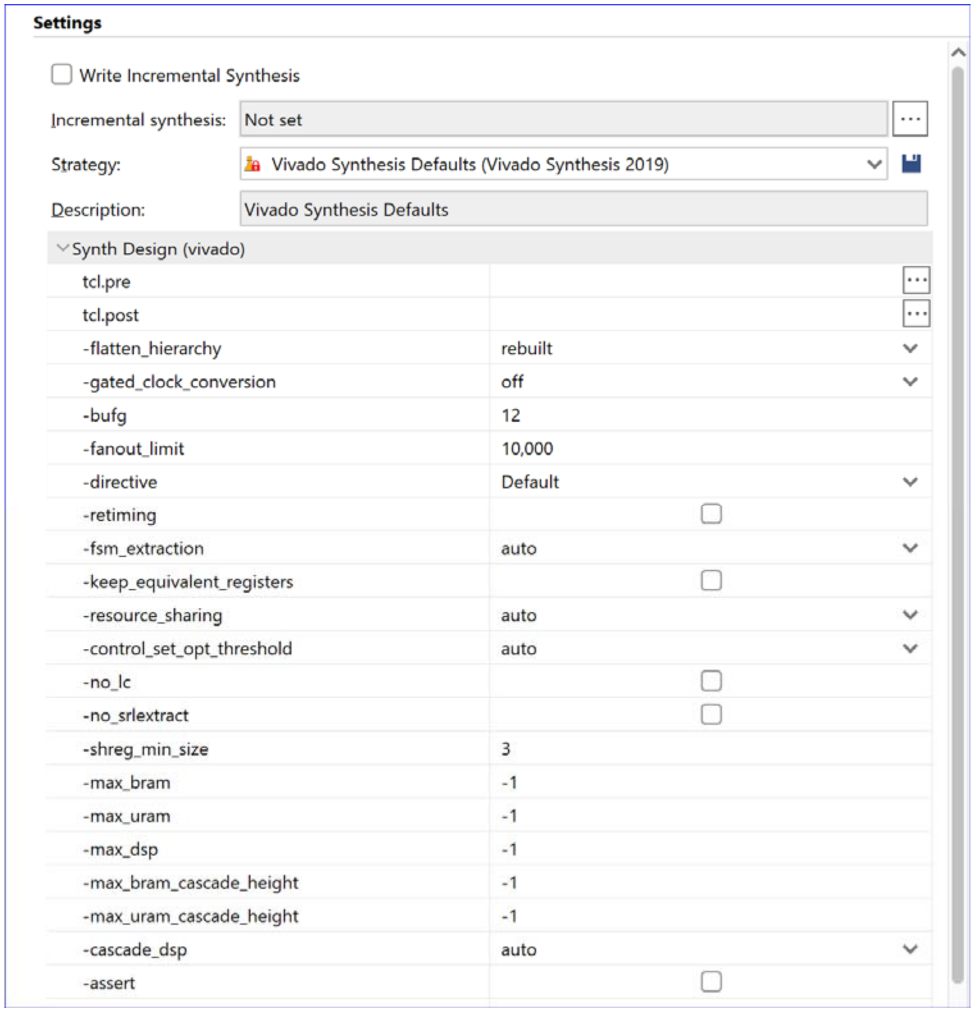
Design Flow – Verification (timing-closure)

- Concept of STA (static timing analysis), setup & hold time of digital circuit, clock domain, meta-stability and etc.
- Able to write the necessary constraint and false-path if necessary, in order to close timing
- Familiar with tcl language (google “ug835” and “ug894”



```
#We need this PL_CLK to latch in PL SYSREF, then cross into ADC or DAC stream clock
#Our stream clock source from RFDC-PLL output, not from PL_CLK (external from PCB)
#Take out, MMCM input is def by tools
create_clock -period 8.138 -name pl_clk_in [get_ports FPGA_REFCLK_OUT_C_P]

#External PL clk (122.88MHz) go into MMCM to produce 491.52MHz for baseband clk
create_generated_clock -source [get_ports FPGA_REFCLK_OUT_C_P] -divide_by 1 [get_pins i_mmcm_pl/inst/clk_in1]

#$clk_pl_0 (named by tools) is clk from ps8. 100MHz not related to baseband clock
set_clock_groups -name async_ps_mmcm -asynchronous -group [get_clocks -of_objects [get_pins {i_ps/ps_block_i/zynq_ultra_
 
#up_bus timing
set_false_path -from [get_pins -of [get_cells -hier -filter {NAME =~ "*i_reg*e1_reg"}] -filter REF_PIN_NAME==C] -to [get
set_max_delay -datapath_only -from [get_clocks [get_clocks -of_objects [get_pins {i_ps/ps_block_i/zynq_ultra_ps_e_0/U0/P
set_max_delay -datapath_only -from [get_clocks [get_clocks -of_objects [get_pins {i_ps/ps_block_i/zynq_ultra_ps_e_0/U0/P
```

Design Flow – Synthesis



Settings

☐ Write Incremental Synthesis

Incremental synthesis:	Not set
Strategy:	Vivado Synthesis Defaults (Vivado Synthesis 2019)
Description:	Vivado Synthesis Defaults

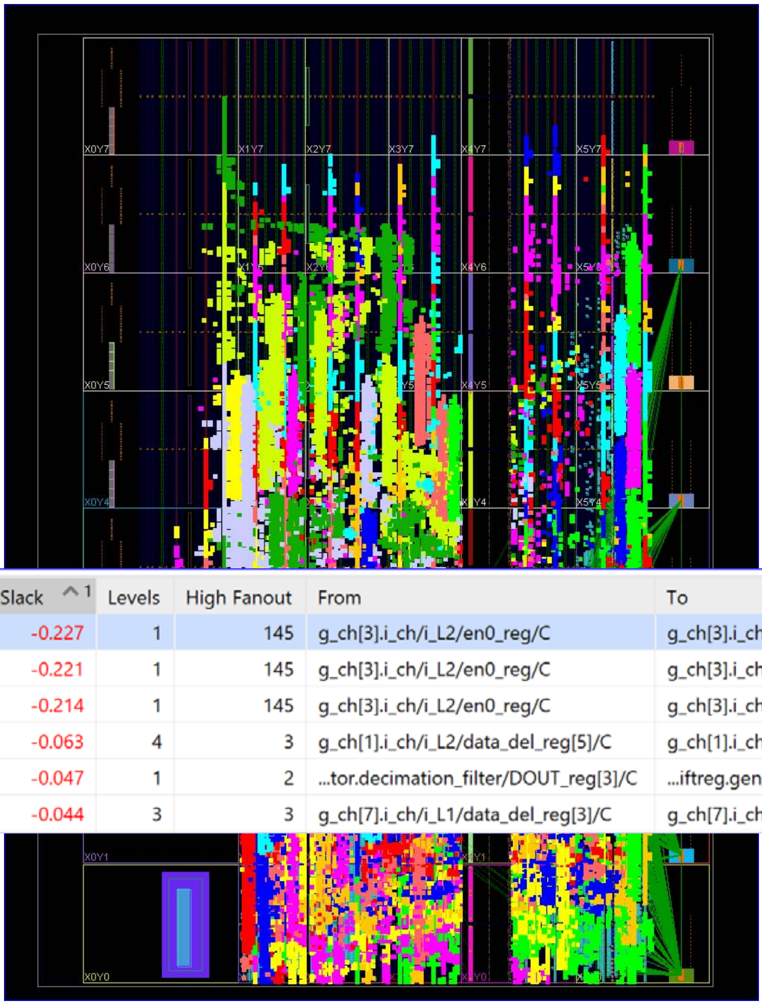
Synth Design (vivado)

tcl.pre	
tcl.post	
-flatten_hierarchy	rebuilt
-gated_clock_conversion	off
-bufg	12
-fanout_limit	10,000
-directive	Default
-retiming	☐
-fsm_extraction	auto
-keep_equivalent_registers	☐
-resource_sharing	auto
-control_set_opt_threshold	auto
-no_lc	☐
-no_srlextract	☐
-shreg_min_size	3
-max_bram	-1
-max_uram	-1
-max_dsp	-1
-max_bram_cascade_height	-1
-max_uram_cascade_height	-1
-cascade_dsp	auto
-assert	☐

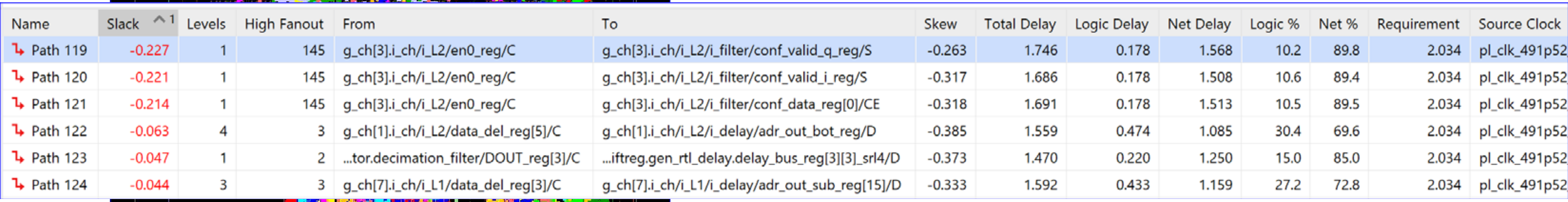
- ❑ Understand how the tool perform mapping between RTL construct and FPGA resource
- ❑ Familiar with various attributes or directives (google “ug901”)

UNCLASSIFIED

Design Flow – Implementation (place and route)

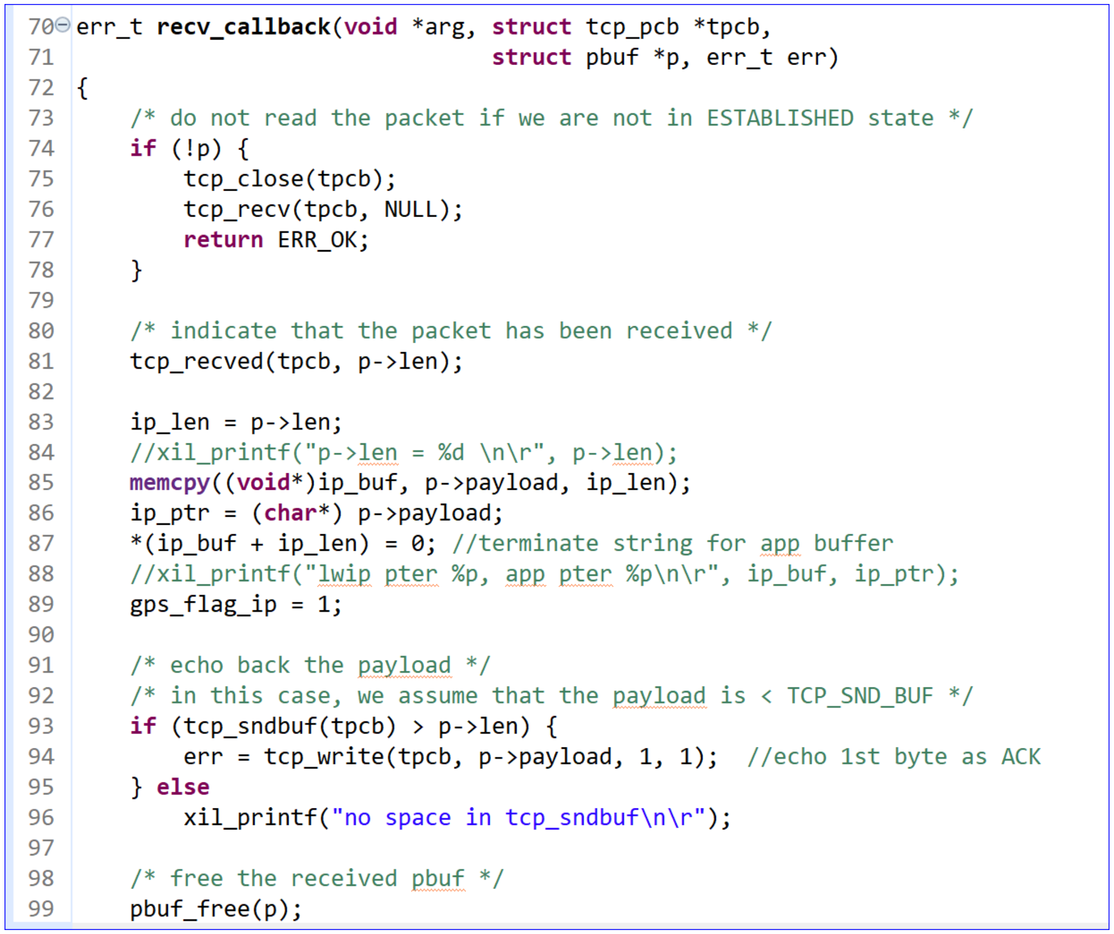


- ❑ Understand the quality of the implemented design, such as resource usage, power consumption and etc
- ❑ Familiar with various attributes and directives (google “ug904”) and how to apply them if implementation can’t pass timing



Name	Slack ^1	Levels	High Fanout	From	To	Skew	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock
Path 119	-0.227	1	145	g_ch[3].i_ch/i_L2/en0_reg/C	g_ch[3].i_ch/i_L2/i_filter/conf_valid_q_reg/S	-0.263	1.746	0.178	1.568	10.2	89.8	2.034	pl_clk_491p52
Path 120	-0.221	1	145	g_ch[3].i_ch/i_L2/en0_reg/C	g_ch[3].i_ch/i_L2/i_filter/conf_valid_i_reg/S	-0.317	1.686	0.178	1.508	10.6	89.4	2.034	pl_clk_491p52
Path 121	-0.214	1	145	g_ch[3].i_ch/i_L2/en0_reg/C	g_ch[3].i_ch/i_L2/i_filter/conf_data_reg[0]/CE	-0.318	1.691	0.178	1.513	10.5	89.5	2.034	pl_clk_491p52
Path 122	-0.063	4	3	g_ch[1].i_ch/i_L2/data_del_reg[5]/C	g_ch[1].i_ch/i_L2/i_delay/adr_out_bot_reg/D	-0.385	1.559	0.474	1.085	30.4	69.6	2.034	pl_clk_491p52
Path 123	-0.047	1	2	...tor.decimation_filter/DOUT_reg[3]/C	...iftreg.gen_rtl_delay.delay_bus_reg[3][3]_srl4/D	-0.373	1.470	0.220	1.250	15.0	85.0	2.034	pl_clk_491p52
Path 124	-0.044	3	3	g_ch[7].i_ch/i_L1/data_del_reg[3]/C	g_ch[7].i_ch/i_L1/i_delay/adr_out_sub_reg[15]/D	-0.333	1.592	0.433	1.159	27.2	72.8	2.034	pl_clk_491p52

Embedded C

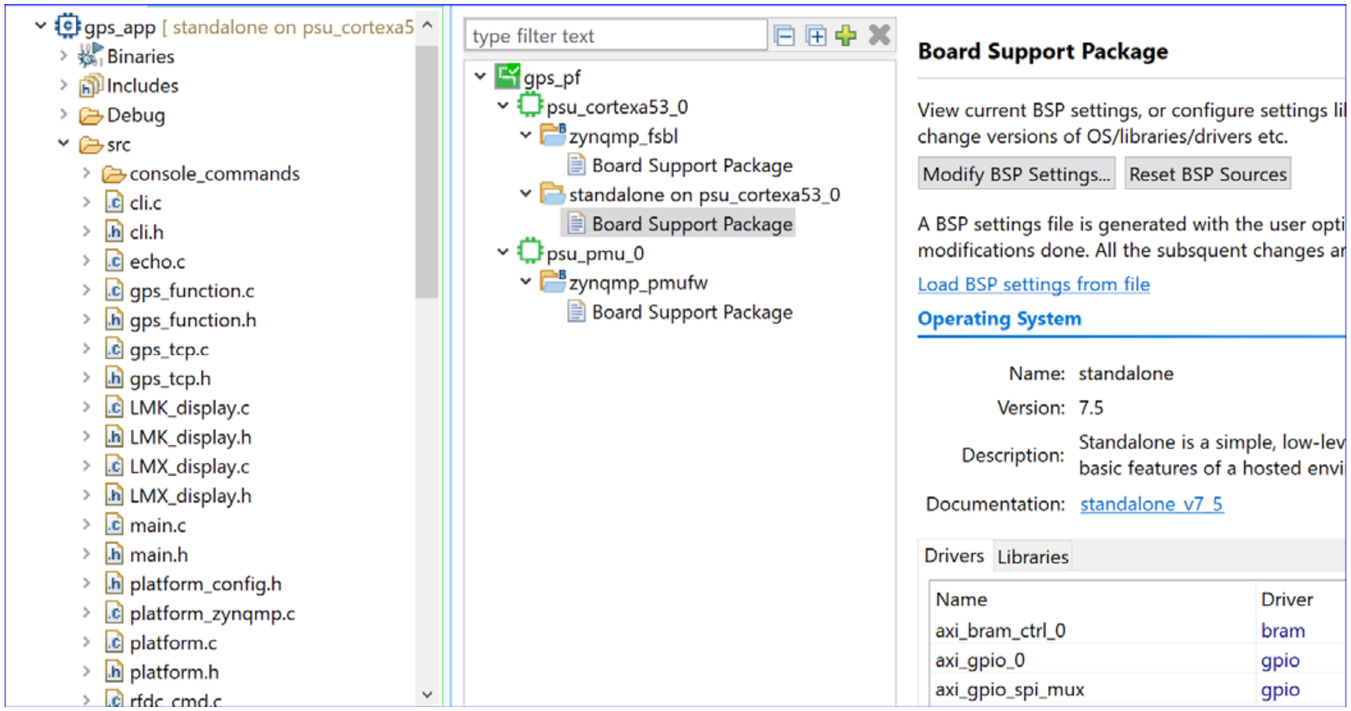


```
err_t recv_callback(void *arg, struct tcp_pcb *tpcb,
                               struct pbuf *p, err_t err)
{
    /* do not read the packet if we are not in ESTABLISHED state */
    if (!p) {
        tcp_close(tpcb);
        tcp_recv(tpcb, NULL);
        return ERR_OK;
    }

    /* indicate that the packet has been received */
    tcp_recved(tpcb, p->len);

    ip_len = p->len;
    //xil_printf("p->len = %d \n\r", p->len);
    memcpy((void*)ip_buf, p->payload, ip_len);
    ip_ptr = (char*) p->payload;
    *(ip_buf + ip_len) = 0; //terminate string for app buffer
    //xil_printf("lwip pter %p, app pter %p\n\r", ip_buf, ip_ptr);
    gps_flag_ip = 1;

    /* echo back the payload */
    /* in this case, we assume that the payload is < TCP_SND_BUF */
    if (tcp_sndbuf(tpcb) > p->len) {
        err = tcp_write(tpcb, p->payload, 1, 1);  //echo 1st byte as ACK
    } else
        xil_printf("no space in tcp_sndbuf\n\r");

    /* free the received pbuf */
    pbuf_free(p);
```

- ❑ Familiar with embedded C from basic to more advanced pointer
- ❑ Have basic concept of polling and interrupt service routine
- ❑ Able to use API or driver provided by BSP

Compiler and Toolchain

- ❑ Familiar with IDE such as Eclipse, Vitis or etc, and proper organization of C code and header files.



gps_app [standalone on psu_cortexa5
Binaries
Includes
Debug
src
console_commands
cli.c
cli.h
echo.c
gps_function.c
gps_function.h
gps_tcp.c
gps_tcp.h
LMK_display.c
LMK_display.h
LMX_display.c
LMX_display.h
main.c
main.h
platform_config.h
platform_zynqmp.c
platform.c
platform.h
rfdc_cmd.c

type filter text
gps_pf
psu_cortexa53_0
zynqmp_fsbl
Board Support Package
standalone on psu_cortexa53_0
Board Support Package
psu_pmu_0
zynqmp_pmufw
Board Support Package

Board Support Package

View current BSP settings, or configure settings li
change versions of OS/libraries/drivers etc.

Modify BSP Settings... Reset BSP Sources

A BSP settings file is generated with the user opti
modifications done. All the subsquent changes ar

Load BSP settings from file

Operating System

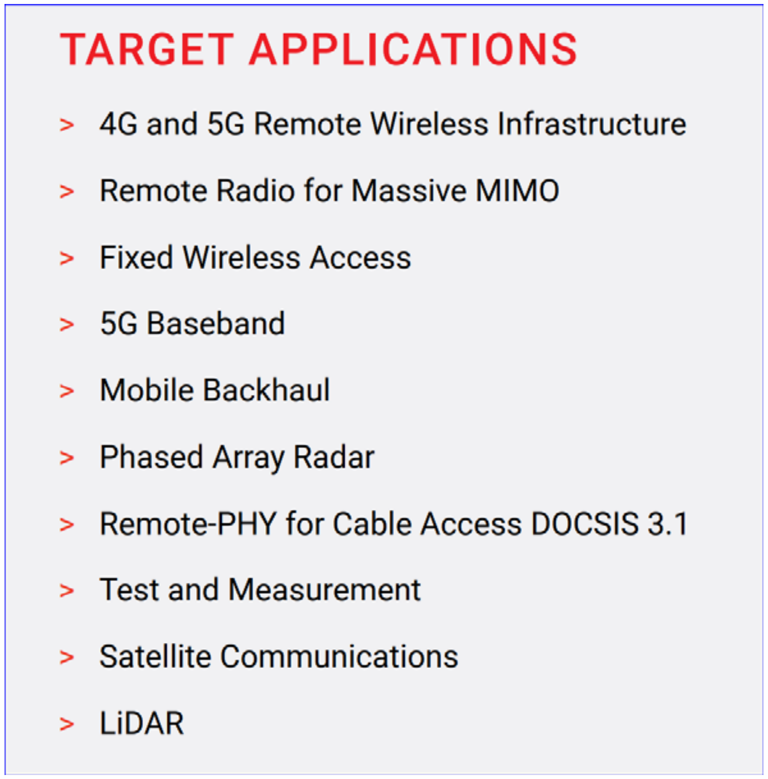
Name:	standalone
Version:	7.5
Description:	Standalone is a simple, low-lev basic features of a hosted envi
Documentation:	standalone_v7_5

Drivers Libraries

Name	Driver
axi_bram_ctrl_0	bram
axi_gpio_0	gpio
axi_gpio_spi_mux	gpio

- ❑ Familiar with C library and BSP provided by IP vendors
- ❑ Process knowledge about how CPU execute its instruction
- ❑ Understand the process of linking and compilation

Zynq UltraScale+ RFSoC ZCU216 Eval Platform


TARGET APPLICATIONS

- 4G and 5G Remote Wireless Infrastructure
- Remote Radio for Massive MIMO
- Fixed Wireless Access
- 5G Baseband
- Mobile Backhaul
- Phased Array Radar
- Remote-PHY for Cable Access DOCSIS 3.1
- Test and Measurement
- Satellite Communications
- LiDAR

- Ug1390
- Schematic
- xilinx-rfsoc-product-brief.pdf

Figure 2: **ZCU216 Component Locations**

(00) Round callout references a component on the front side of the board

[00] Square callout references a component on the back side of the board

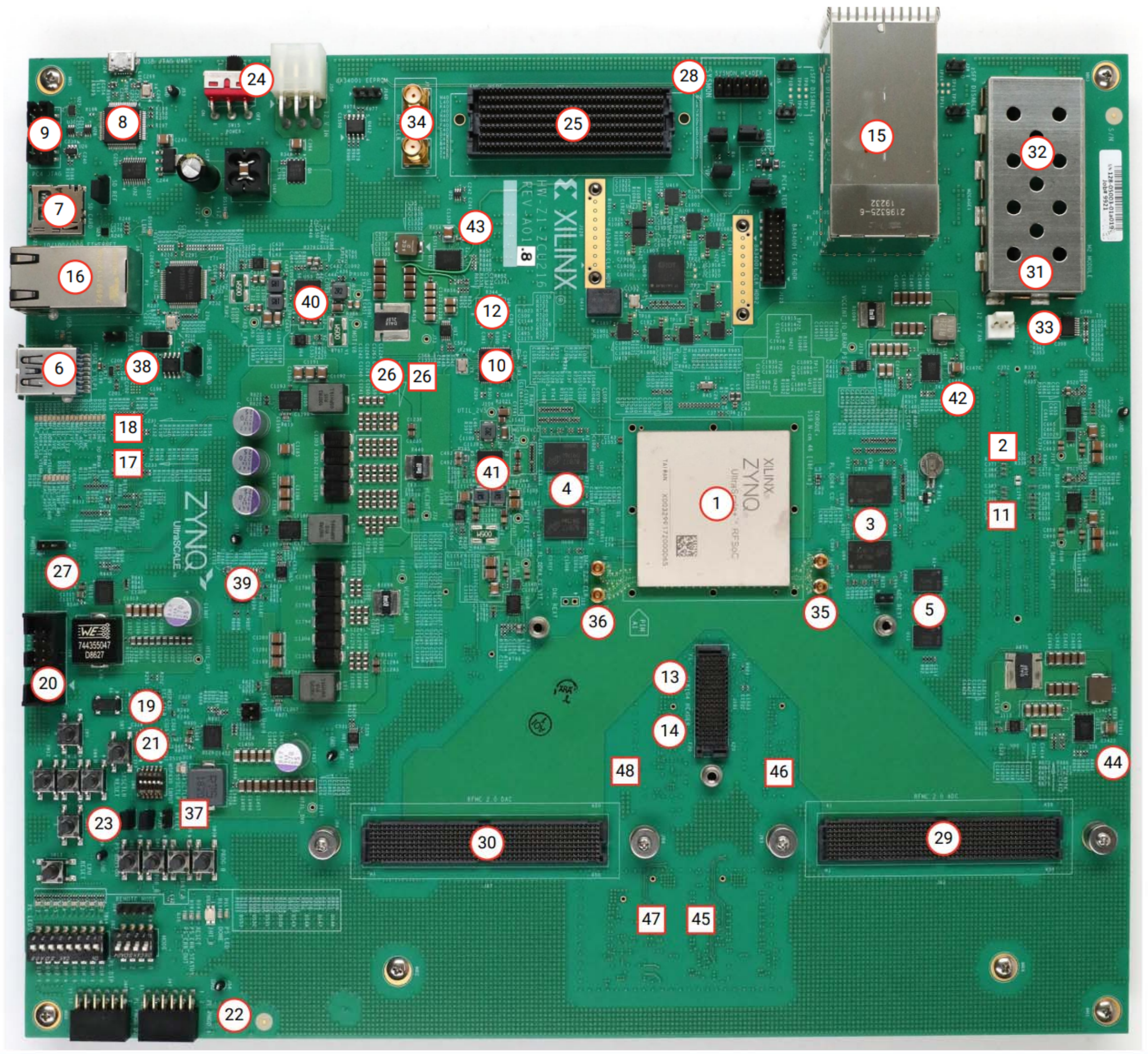
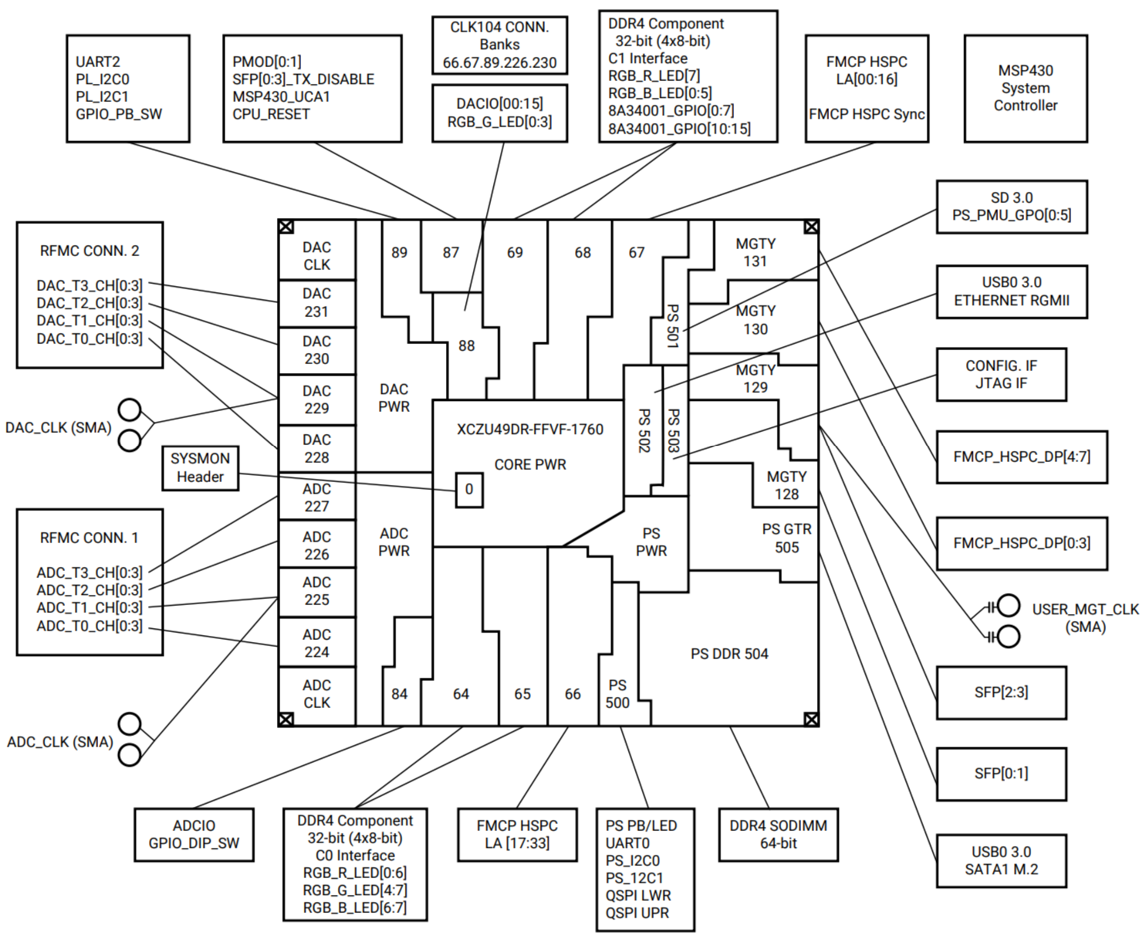


Figure 1: **ZCU216 Evaluation Board Block Diagram**



Zynq UltraScale+ RFSoC Feature Summary

URAM 80 block of 4K*72b (288Kb)
80*4K*72b = 320K*72b=23040Kb = = 22.5Mb

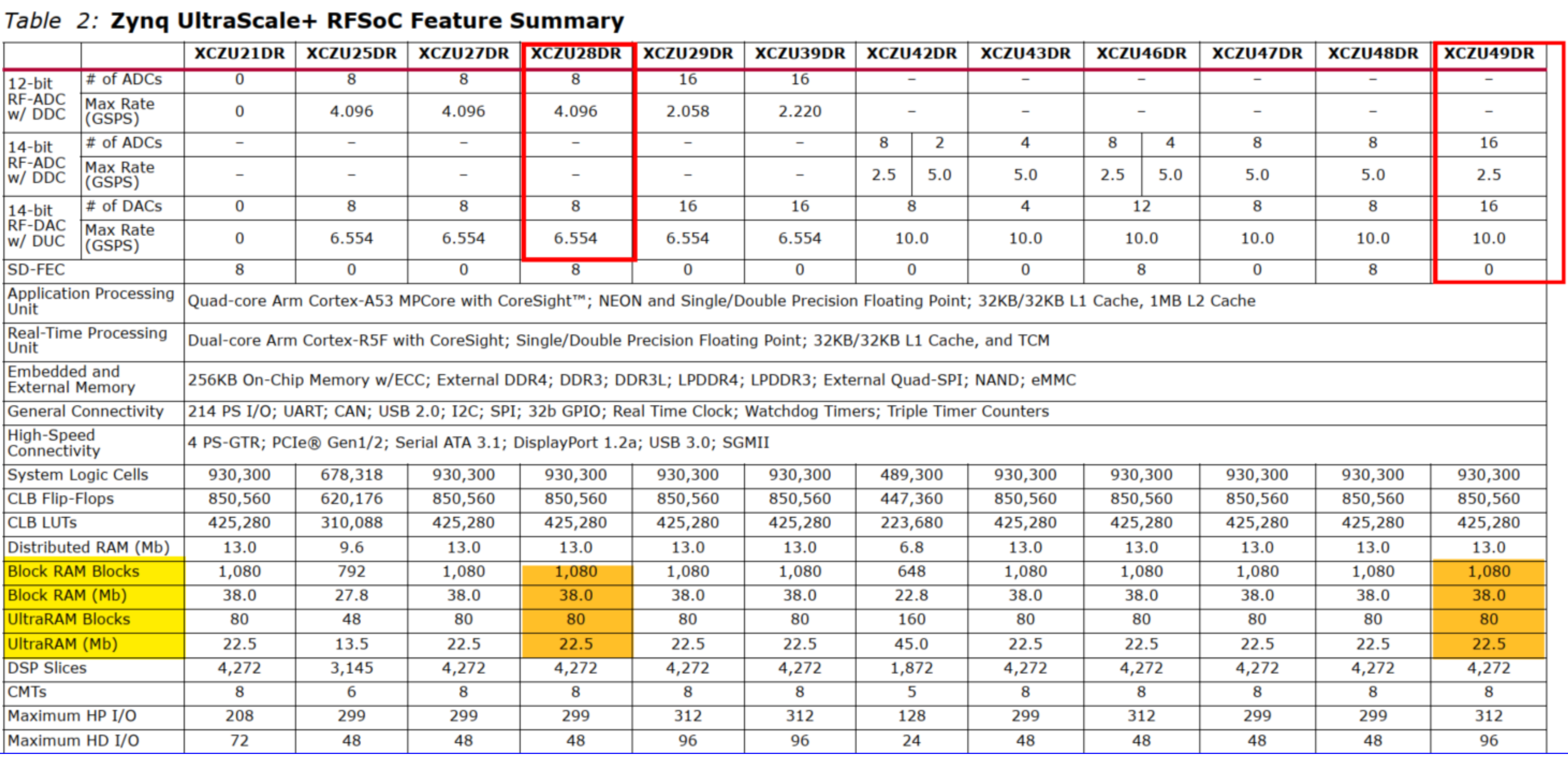
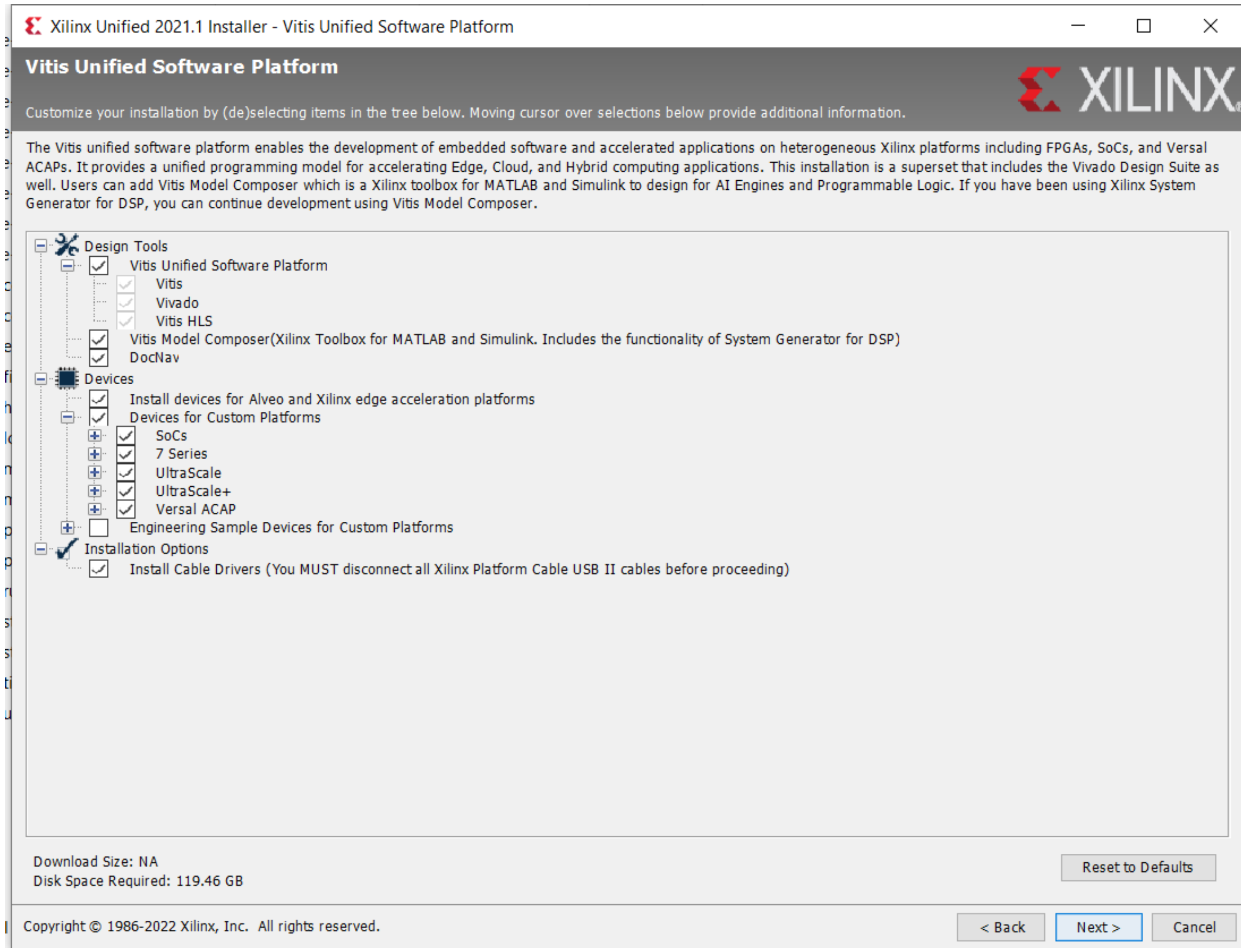


Table 2: **Zynq UltraScale+ RFSoC Feature Summary**

		XCZU21DR	XCZU25DR	XCZU27DR	XCZU28DR	XCZU29DR	XCZU39DR	XCZU42DR	XCZU43DR	XCZU46DR	XCZU47DR	XCZU48DR	XCZU49DR
12-bit RF-ADC w/ DDC	# of ADCs	0	8	8	8	16	16	–	–	–	–	–	–
	Max Rate (GSPS)	0	4.096	4.096	4.096	2.058	2.220	–	–	–	–	–	–
14-bit RF-ADC w/ DDC	# of ADCs	–	–	–	–	–	–	8 2	4	8 4	8	8	16
	Max Rate (GSPS)	–	–	–	–	–	–	2.5 5.0	5.0	2.5 5.0	5.0	5.0	2.5
14-bit RF-DAC w/ DUC	# of DACs	0	8	8	8	16	16	8	4	12	8	8	16
	Max Rate (GSPS)	0	6.554	6.554	6.554	6.554	6.554	10.0	10.0	10.0	10.0	10.0	10.0
SD-FEC		8	0	0	8	0	0	0	0	8	0	8	0
Application Processing Unit		Quad-core Arm Cortex-A53 MPCore with CoreSight™; NEON and Single/Double Precision Floating Point; 32KB/32KB L1 Cache, 1MB L2 Cache											
Real-Time Processing Unit		Dual-core Arm Cortex-R5F with CoreSight; Single/Double Precision Floating Point; 32KB/32KB L1 Cache, and TCM											
Embedded and External Memory		256KB On-Chip Memory w/ECC; External DDR4; DDR3; DDR3L; LPDDR4; LPDDR3; External Quad-SPI; NAND; eMMC											
General Connectivity		214 PS I/O; UART; CAN; USB 2.0; I2C; SPI; 32b GPIO; Real Time Clock; Watchdog Timers; Triple Timer Counters											
High-Speed Connectivity		4 PS-GTR; PCIe® Gen1/2; Serial ATA 3.1; DisplayPort 1.2a; USB 3.0; SGMII											
System Logic Cells		930,300	678,318	930,300	930,300	930,300	930,300	489,300	930,300	930,300	930,300	930,300	930,300
CLB Flip-Flops		850,560	620,176	850,560	850,560	850,560	850,560	447,360	850,560	850,560	850,560	850,560	850,560
CLB LUTs		425,280	310,088	425,280	425,280	425,280	425,280	223,680	425,280	425,280	425,280	425,280	425,280
Distributed RAM (Mb)		13.0	9.6	13.0	13.0	13.0	13.0	6.8	13.0	13.0	13.0	13.0	13.0
Block RAM Blocks		1,080	792	1,080	1,080	1,080	1,080	648	1,080	1,080	1,080	1,080	1,080
Block RAM (Mb)		38.0	27.8	38.0	38.0	38.0	38.0	22.8	38.0	38.0	38.0	38.0	38.0
UltraRAM Blocks		80	48	80	80	80	80	160	80	80	80	80	80
UltraRAM (Mb)		22.5	13.5	22.5	22.5	22.5	22.5	45.0	22.5	22.5	22.5	22.5	22.5
DSP Slices		4,272	3,145	4,272	4,272	4,272	4,272	1,872	4,272	4,272	4,272	4,272	4,272
CMTs		8	6	8	8	8	8	5	8	8	8	8	8
Maximum HP I/O		208	299	299	299	312	312	128	299	312	299	299	312
Maximum HD I/O		72	48	48	48	96	96	24	48	48	48	48	96

Tools:

- ❑ Get the right version



Xilinx Unified 2021.1 Installer - Vitis Unified Software Platform

Vitis Unified Software Platform

Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.

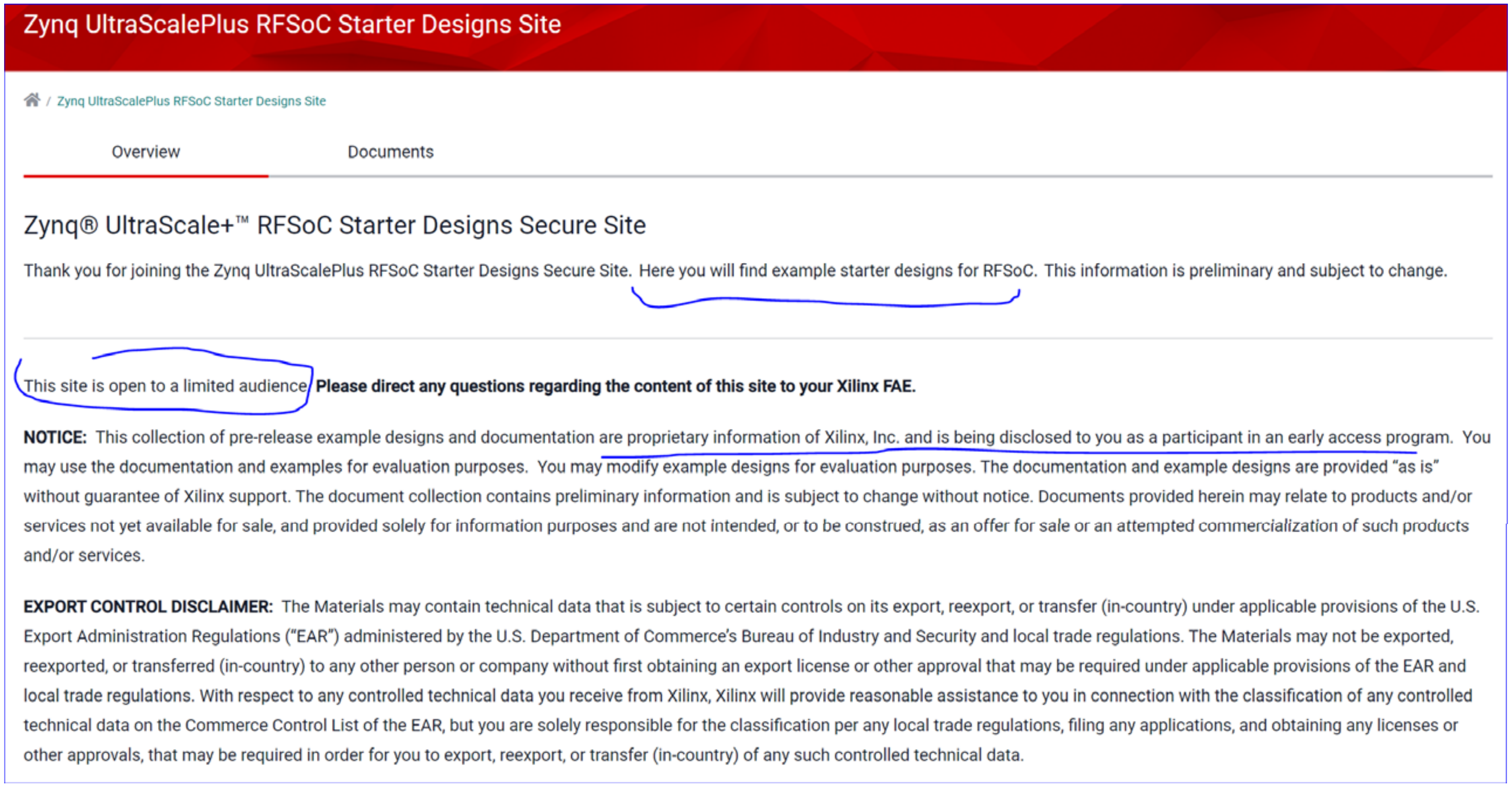
The Vitis unified software platform enables the development of embedded software and accelerated applications on heterogeneous Xilinx platforms including FPGAs, SoCs, and Versal ACAPs. It provides a unified programming model for accelerating Edge, Cloud, and Hybrid computing applications. This installation is a superset that includes the Vivado Design Suite as well. Users can add Vitis Model Composer which is a Xilinx toolbox for MATLAB and Simulink to design for AI Engines and Programmable Logic. If you have been using Xilinx System Generator for DSP, you can continue development using Vitis Model Composer.

- Design Tools
  - [✓] Vitis Unified Software Platform
    - [✓] Vitis
    - [✓] Vivado
    - [✓] Vitis HLS
  - [✓] Vitis Model Composer(Xilinx Toolbox for MATLAB and Simulink. Includes the functionality of System Generator for DSP)
  - [✓] DocNav
- Devices
  - [✓] Install devices for Alveo and Xilinx edge acceleration platforms
  - [✓] Devices for Custom Platforms
    - [✓] SoCs
    - [✓] 7 Series
    - [✓] UltraScale
    - [✓] UltraScale+
    - [✓] Versal ACAP
  - [ ] Engineering Sample Devices for Custom Platforms
- Installation Options
  - [✓] Install Cable Drivers (You MUST disconnect all Xilinx Platform Cable USB II cables before proceeding)

Download Size: NA
Disk Space Required: 119.46 GB

Reference Design

- https://www.xilinx.com/member/zuplus_rfsoc_starter_designs.html


Zynq UltraScalePlus RFSoC Starter Designs Site
Zynq UltraScalePlus RFSoC Starter Designs Site
Overview
Documents

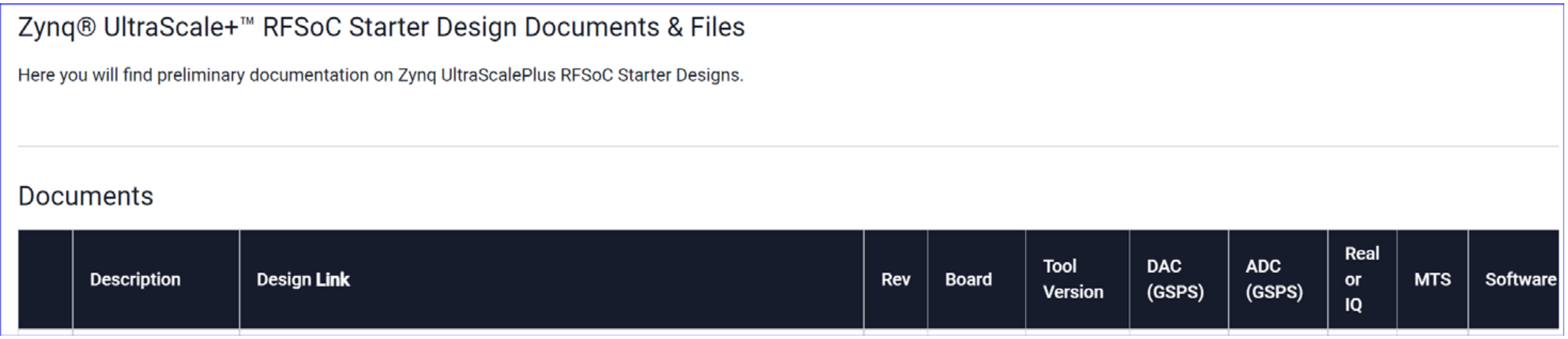
Zynq® UltraScale+™ RFSoC Starter Designs Secure Site

Thank you for joining the Zynq UltraScalePlus RFSoC Starter Designs Secure Site. Here you will find example starter designs for RFSoC. This information is preliminary and subject to change.

This site is open to a limited audience. **Please direct any questions regarding the content of this site to your Xilinx FAE.**

NOTICE: This collection of pre-release example designs and documentation are proprietary information of Xilinx, Inc. and is being disclosed to you as a participant in an early access program. You may use the documentation and examples for evaluation purposes. You may modify example designs for evaluation purposes. The documentation and example designs are provided "as is" without guarantee of Xilinx support. The document collection contains preliminary information and is subject to change without notice. Documents provided herein may relate to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of such products and/or services.

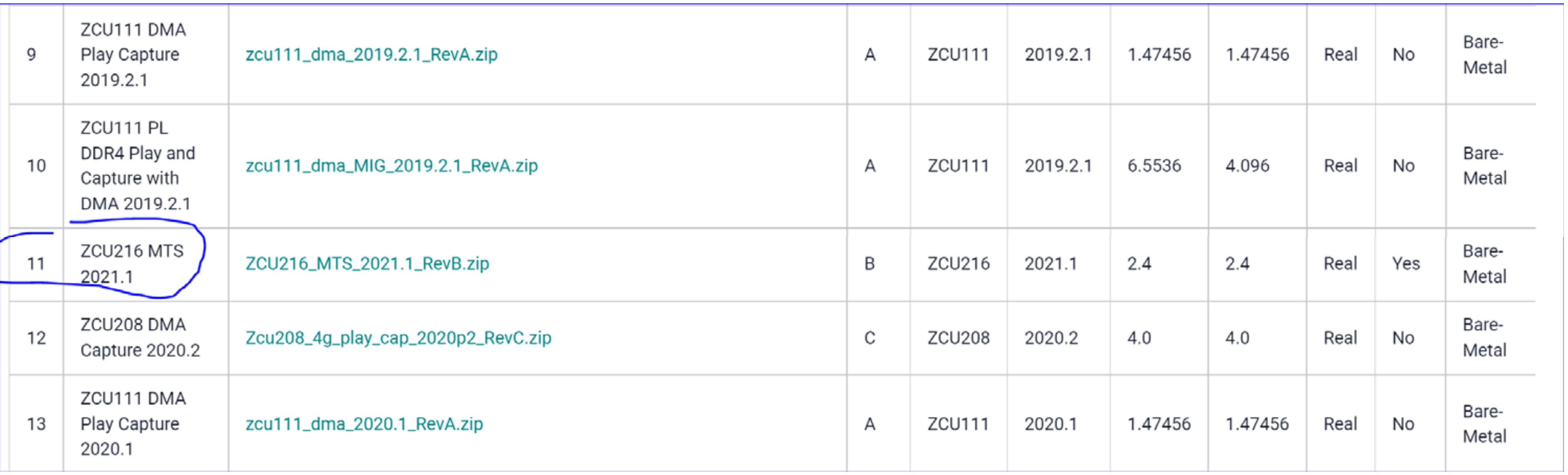
EXPORT CONTROL DISCLAIMER: The Materials may contain technical data that is subject to certain controls on its export, reexport, or transfer (in-country) under applicable provisions of the U.S. Export Administration Regulations ("EAR") administered by the U.S. Department of Commerce's Bureau of Industry and Security and local trade regulations. The Materials may not be exported, reexported, or transferred (in-country) to any other person or company without first obtaining an export license or other approval that may be required under applicable provisions of the EAR and local trade regulations. With respect to any controlled technical data you receive from Xilinx, Xilinx will provide reasonable assistance to you in connection with the classification of any controlled technical data on the Commerce Control List of the EAR, but you are solely responsible for the classification per any local trade regulations, filing any applications, and obtaining any licenses or other approvals, that may be required in order for you to export, reexport, or transfer (in-country) of any such controlled technical data.



Zynq® UltraScale+™ RFSoC Starter Design Documents & Files

Here you will find preliminary documentation on Zynq UltraScalePlus RFSoC Starter Designs.

Documents

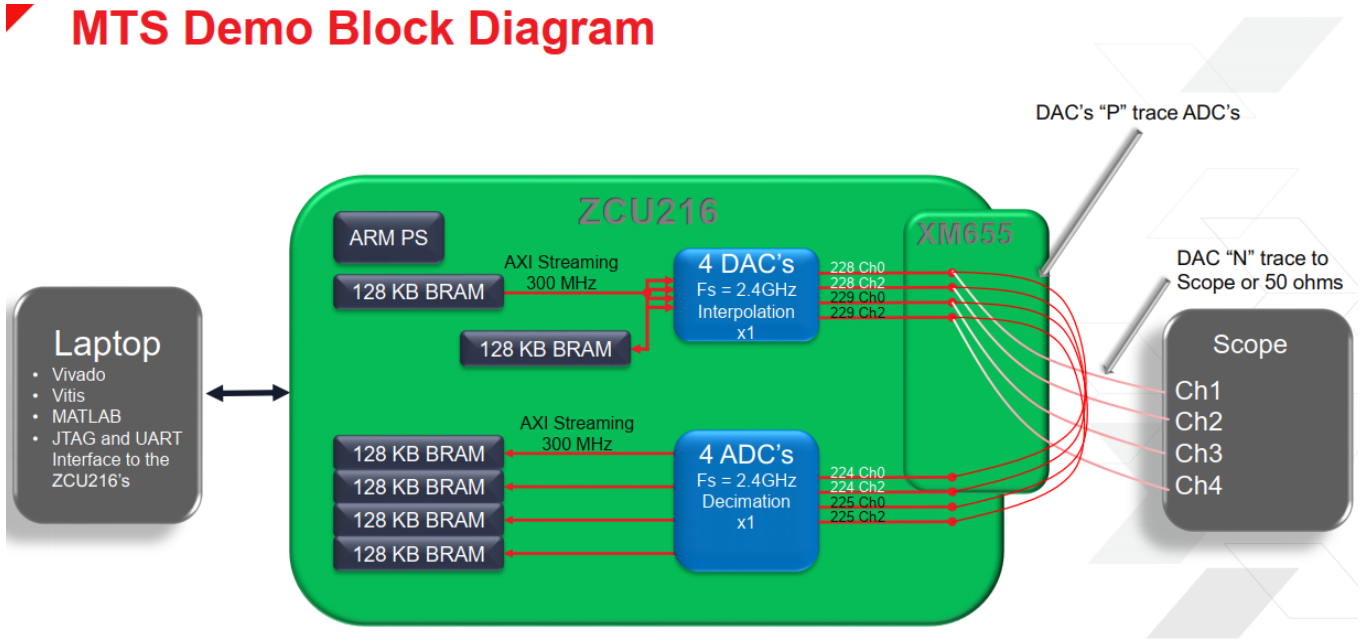


	Description	Design Link	Rev	Board	Tool Version	DAC (GSPS)	ADC (GSPS)	Real or IQ	MTS	Software
9	ZCU111 DMA Play Capture 2019.2.1	zcu111_dma_2019.2.1_RevA.zip	A	ZCU111	2019.2.1	1.47456	1.47456	Real	No	Bare-Metal
10	ZCU111 PL DDR4 Play and Capture with DMA 2019.2.1	zcu111_dma_MIG_2019.2.1_RevA.zip	A	ZCU111	2019.2.1	6.5536	4.096	Real	No	Bare-Metal
11	ZCU216 MTS 2021.1	ZCU216_MTS_2021.1_RevB.zip	B	ZCU216	2021.1	2.4	2.4	Real	Yes	Bare-Metal
12	ZCU208 DMA Capture 2020.2	Zcu208_4g_play_cap_2020p2_RevC.zip	C	ZCU208	2020.2	4.0	4.0	Real	No	Bare-Metal
13	ZCU111 DMA Play Capture 2020.1	zcu111_dma_2020.1_RevA.zip	A	ZCU111	2020.1	1.47456	1.47456	Real	No	Bare-Metal

Demonstration of design flow from RTL to Application C Code

- ❑ Use a reference design provided by Xilinx for ZCU216 RFSoC Prototype Board
- ❑ Ref Design
- ❑ To show the code and result at various check points of design flow using Xilinx Vivado and Vitis

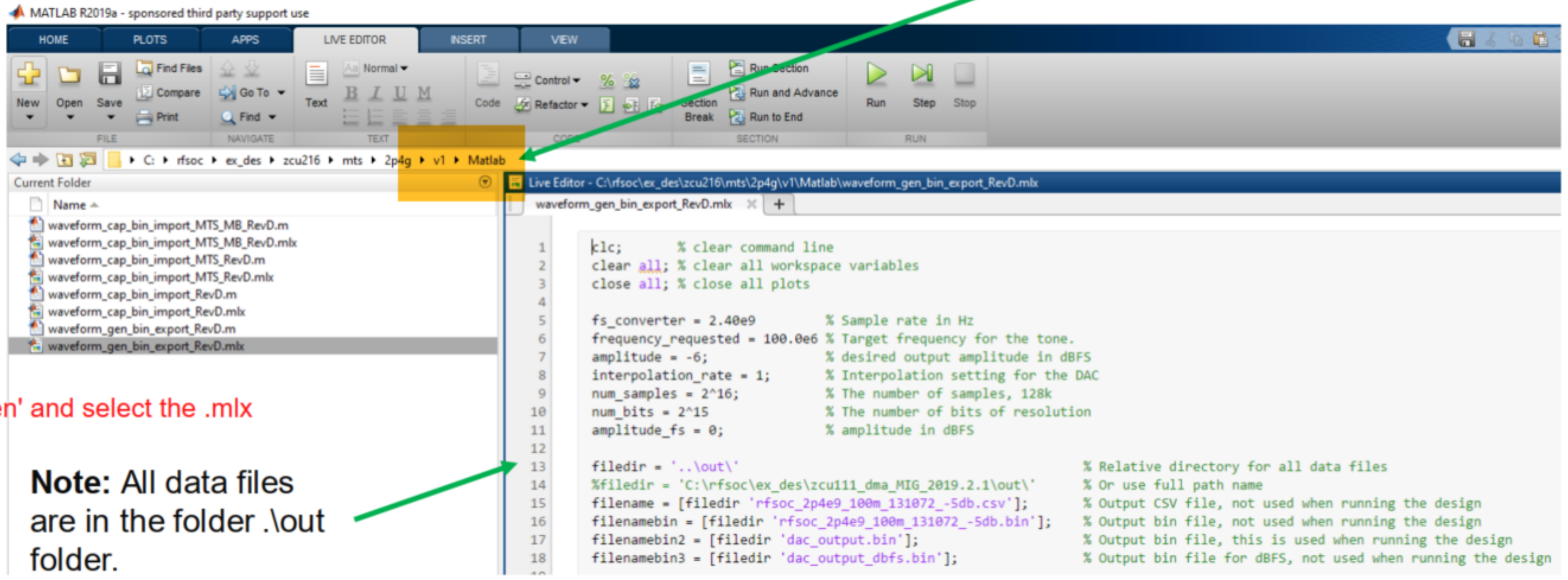
MTS Demo Block Diagram



MATLAB DAC Tone Generator

Just gen wf and write to binary file,
tcl cmd will sent this content to BRAM

New tones can be created using waveform_gen_bin_export_RevD.mlx
The design comes with a pre-generated 100 MHz tone to use.
Note: This file slightly adjusts the frequency of the tone to properly fit into the number of samples.
Note: All MATLAB and Octave files are in the folder .\Matlab folder. Choose that folder in MATLAB.


MATLAB R2019a - sponsored third party support use
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
New
Open
Save
Find Files
Compare
Print
Go To
Find
Text
Normal
Code
Control
Refactor
Section Break
Run Section
Run and Advance
Run to End
Run
Step
Stop
FILE
NAVIGATE
TEXT
CODE
SECTION
RUN
C: ▸ rfsoc ▸ ex_des ▸ zcu216 ▸ mts ▸ 2p4g ▸ v1 ▸ Matlab
Current Folder
Name
Live Editor - C:\rfsoc\ex_des\zcu216\mts\2p4g\v1\Matlab\waveform_gen_bin_export_RevD.mlx
waveform_gen_bin_export_RevD.mlx

- waveform_cap_bin_import_MTS_MB_RevD.m
- waveform_cap_bin_import_MTS_MB_RevD.mlx
- waveform_cap_bin_import_MTS_RevD.m
- waveform_cap_bin_import_MTS_RevD.mlx
- waveform_cap_bin_import_RevD.m
- waveform_cap_bin_import_RevD.mlx
- waveform_gen_bin_export_RevD.m
- waveform_gen_bin_export_RevD.mlx

```
clc;        % clear command line
clear all; % clear all workspace variables
close all; % close all plots

fs_converter = 2.40e9         % Sample rate in Hz
frequency_requested = 100.0e6 % Target frequency for the tone.
amplitude = -6;               % desired output amplitude in dBFS
interpolation_rate = 1;       % Interpolation setting for the DAC
num_samples = 2^16;           % The number of samples, 128k
num_bits = 2^15               % The number of bits of resolution
amplitude_fs = 0;             % amplitude in dBFS

filedir = '..\out\'                                              % Relative directory for all data files
%filedir = 'C:\rfsoc\ex_des\zcu111_dma_MIG_2019.2.1\out\'        % Or use full path name
filename = [filedir 'rfsoc_2p4e9_100m_131072_-5db.csv'];         % Output CSV file, not used when running the design
filenamebin = [filedir 'rfsoc_2p4e9_100m_131072_-5db.bin'];      % Output bin file, not used when running the design
filenamebin2 = [filedir 'dac_output.bin'];                       % Output bin file, this is used when running the design
filenamebin3 = [filedir 'dac_output_dbfs.bin'];                  % Output bin file for dBFS, not used when running the design
```

'open' and select the .mlx

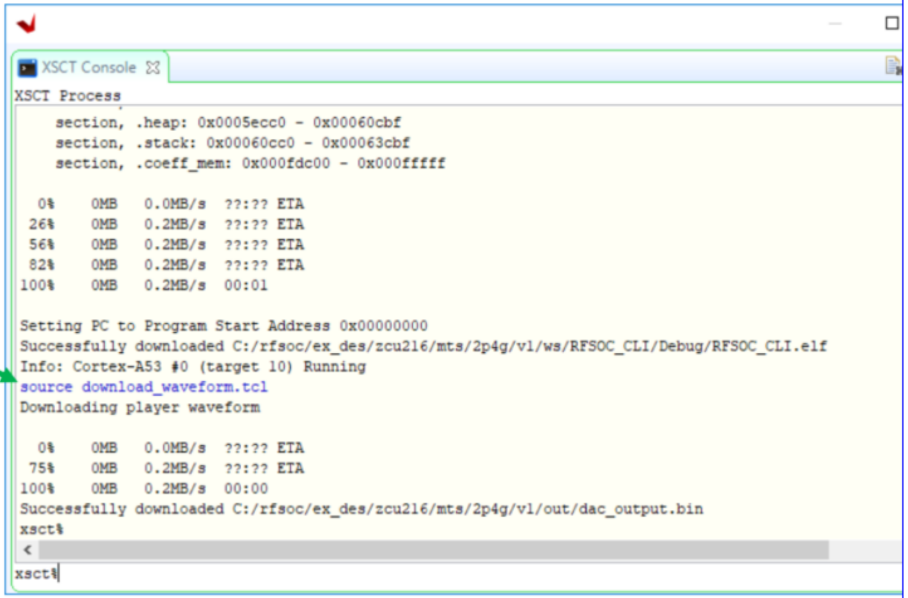
Note: All data files are in the folder .\out folder.

Download Waveform to memory

Adr Editor: /hier_play/axi_bram_ctrl_0/S_AXI is 0xa008000 of 128KB

download_waveform.tcl performs the following functions.

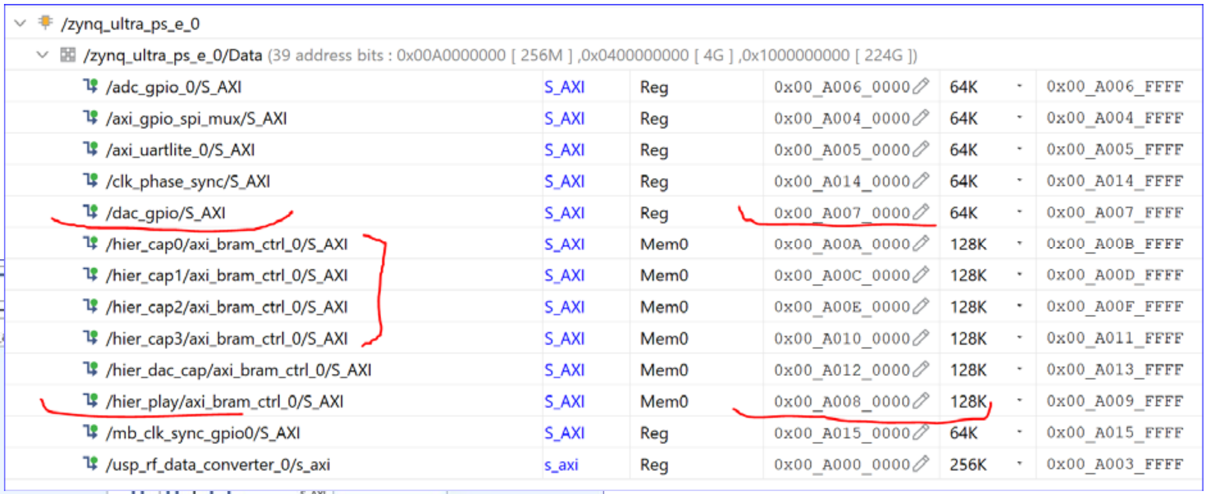
1. Downloads the file 'dac_output.bin' to 128kB of BRAM.
2. The file is downloaded to address 0xa0080000.
3. The default file is a 100 MHz tone in a 128 kB file size, 64k samples.



```
XSCT Process
    section, .heap: 0x0005ecc0 - 0x00060cbf
    section, .stack: 0x00060cc0 - 0x00063cbf
    section, .coeff_mem: 0x000fdc00 - 0x000fffff

  0%    0MB   0.0MB/s  ??:?? ETA
 26%    0MB   0.2MB/s  ??:?? ETA
 56%    0MB   0.2MB/s  ??:?? ETA
 82%    0MB   0.2MB/s  ??:?? ETA
100%    0MB   0.2MB/s  00:01

Setting PC to Program Start Address 0x00000000
Successfully downloaded C:/rfsoc/ex_des/zcu216/mts/2p4g/v1/ws/RFSOC_CLI/Debug/RFSOC_CLI.elf
Info: Cortex-A53 #0 (target 10) Running
source download_waveform.tcl
Downloading player waveform

  0%    0MB   0.0MB/s  ??:?? ETA
 75%    0MB   0.2MB/s  ??:?? ETA
100%    0MB   0.2MB/s  00:00
Successfully downloaded C:/rfsoc/ex_des/zcu216/mts/2p4g/v1/out/dac_output.bin
xsct%

xsct%
```

```
# Download player waveform to uram
puts "Downloading player waveform"
targets -set -filter {name =~ "PSU"};
# targets -target-properties to find JTAG ID's - search for  "level 0 name PSU"
#targets -set 9;
# 0x40001000
dow -force -data ./out/dac_output.bin 0xa0080000
```

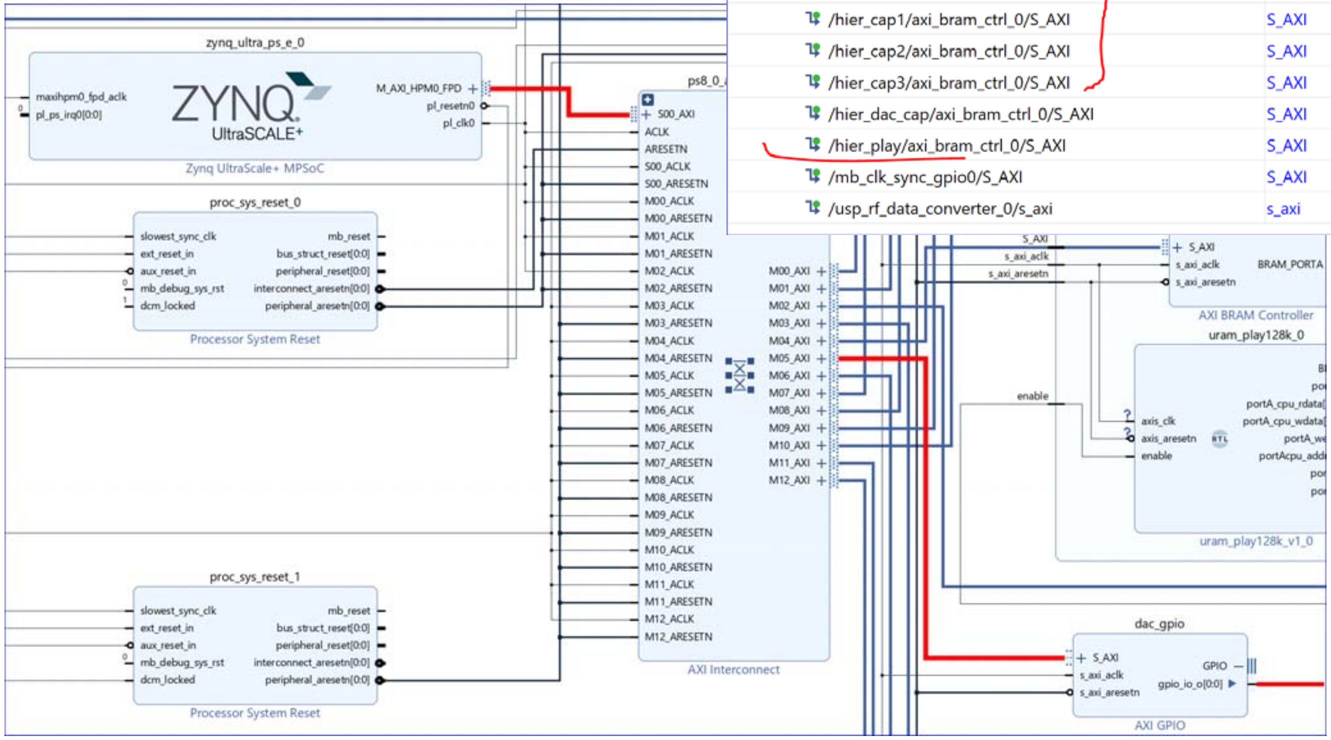
source download_waveform.tcl

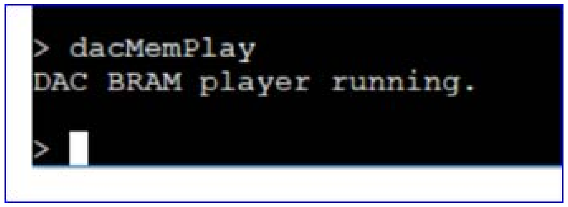


/zynq_ultra_ps_e_0

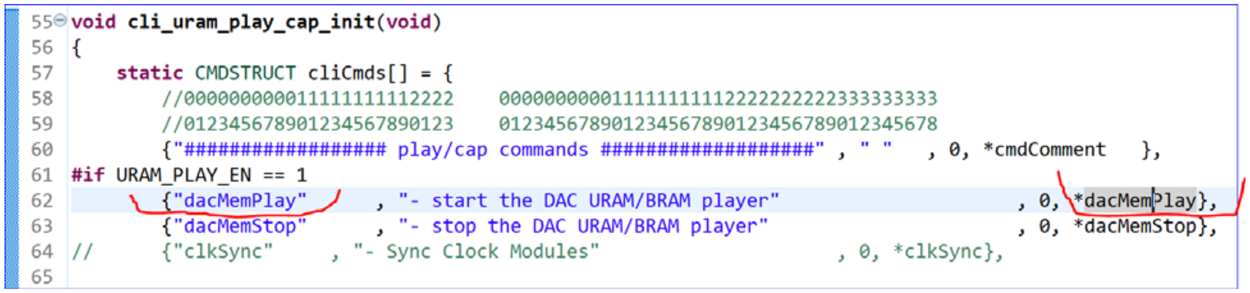
/zynq_ultra_ps_e_0/Data (39 address bits : 0x00A0000000 [256M] ,0x0400000000 [4G] ,0x1000000000 [224G])

/adc_gpio_0/S_AXI	S_AXI	Reg	0x00_A006_0000	64K	-	0x00_A006_FFFF
/axi_gpio_spi_mux/S_AXI	S_AXI	Reg	0x00_A004_0000	64K	-	0x00_A004_FFFF
/axi_uartlite_0/S_AXI	S_AXI	Reg	0x00_A005_0000	64K	-	0x00_A005_FFFF
/clk_phase_sync/S_AXI	S_AXI	Reg	0x00_A014_0000	64K	-	0x00_A014_FFFF
/dac_gpio/S_AXI	S_AXI	Reg	0x00_A007_0000	64K	-	0x00_A007_FFFF
/hier_cap0/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A00A_0000	128K	-	0x00_A00B_FFFF
/hier_cap1/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A00C_0000	128K	-	0x00_A00D_FFFF
/hier_cap2/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A00E_0000	128K	-	0x00_A00F_FFFF
/hier_cap3/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A010_0000	128K	-	0x00_A011_FFFF
/hier_dac_cap/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A012_0000	128K	-	0x00_A013_FFFF
/hier_play/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A008_0000	128K	-	0x00_A009_FFFF
/mb_clk_sync_gpio0/S_AXI	S_AXI	Reg	0x00_A015_0000	64K	-	0x00_A015_FFFF
/usp_rf_data_converter_0/s_axi	s_axi	Reg	0x00_A000_0000	256K	-	0x00_A003_FFFF

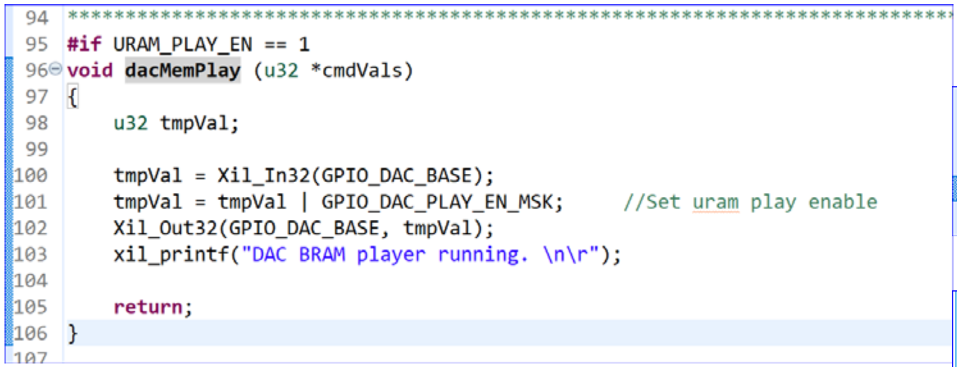




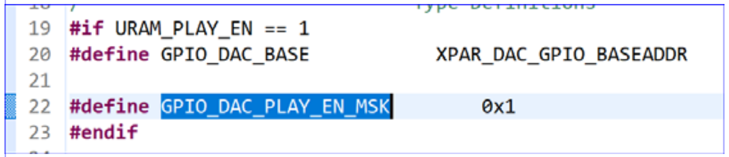
```
> dacMemPlay
DAC BRAM player running.

>
```



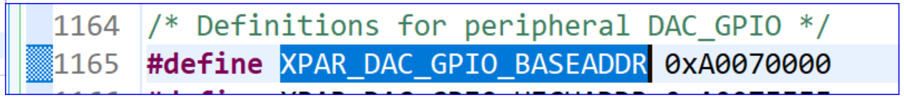
```c
void cli_uram_play_cap_init(void)
{
    static CMDSTRUCT cliCmds[] = {
        //000000000011111111112222    0000000000111111111122222222223333333333
        //012345678901234567890123    0123456789012345678901234567890123456789
        {"################## play/cap commands ###################" , " "    , 0, *cmdComment   },
#if URAM_PLAY_EN == 1
        {"dacMemPlay"      , "- start the DAC URAM/BRAM player"                     , 0, *dacMemPlay},
        {"dacMemStop"      , "- stop the DAC URAM/BRAM player"                      , 0, *dacMemStop},
//      {"clkSync"     , "- Sync Clock Modules"                         , 0, *clkSync},

```



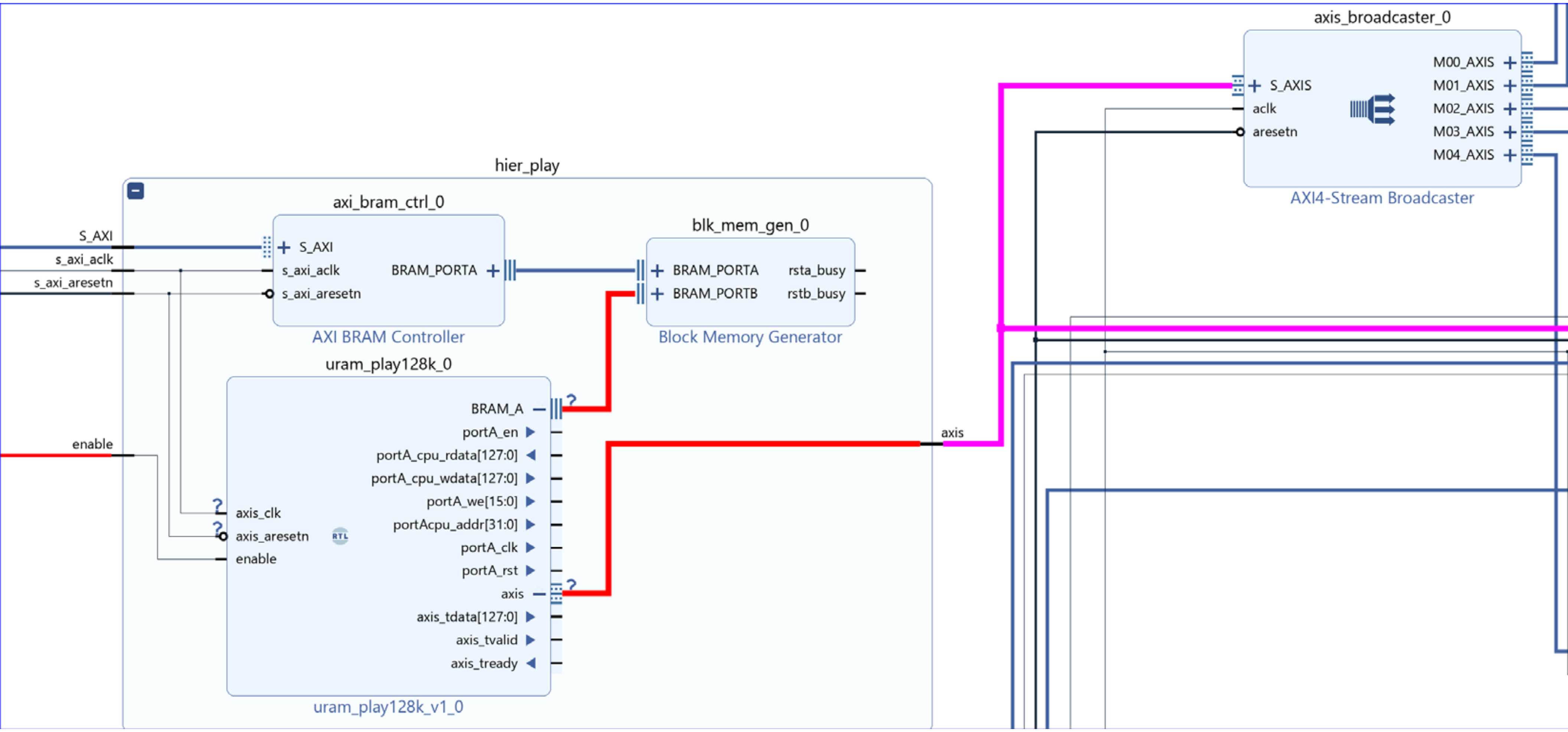
```c
*****************************************************************************
#if URAM_PLAY_EN == 1
void dacMemPlay (u32 *cmdVals)
{
    u32 tmpVal;

    tmpVal = Xil_In32(GPIO_DAC_BASE);
    tmpVal = tmpVal | GPIO_DAC_PLAY_EN_MSK;     //Set uram play enable
    Xil_Out32(GPIO_DAC_BASE, tmpVal);
    xil_printf("DAC BRAM player running. \n\r");

    return;
}
```

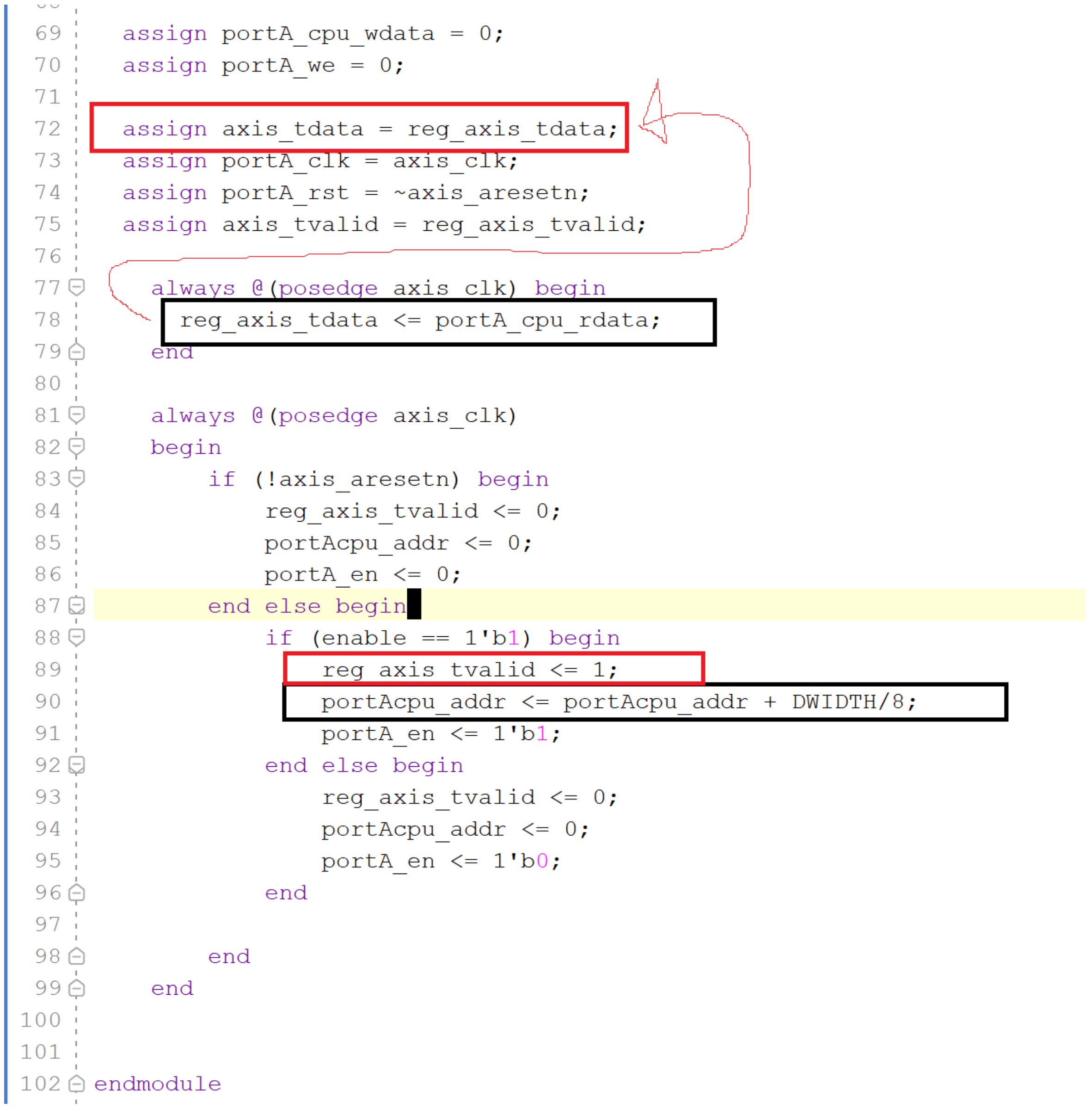


```c
#if URAM_PLAY_EN == 1
#define GPIO_DAC_BASE              XPAR_DAC_GPIO_BASEADDR

#define GPIO_DAC_PLAY_EN_MSK          0x1
#endif
```



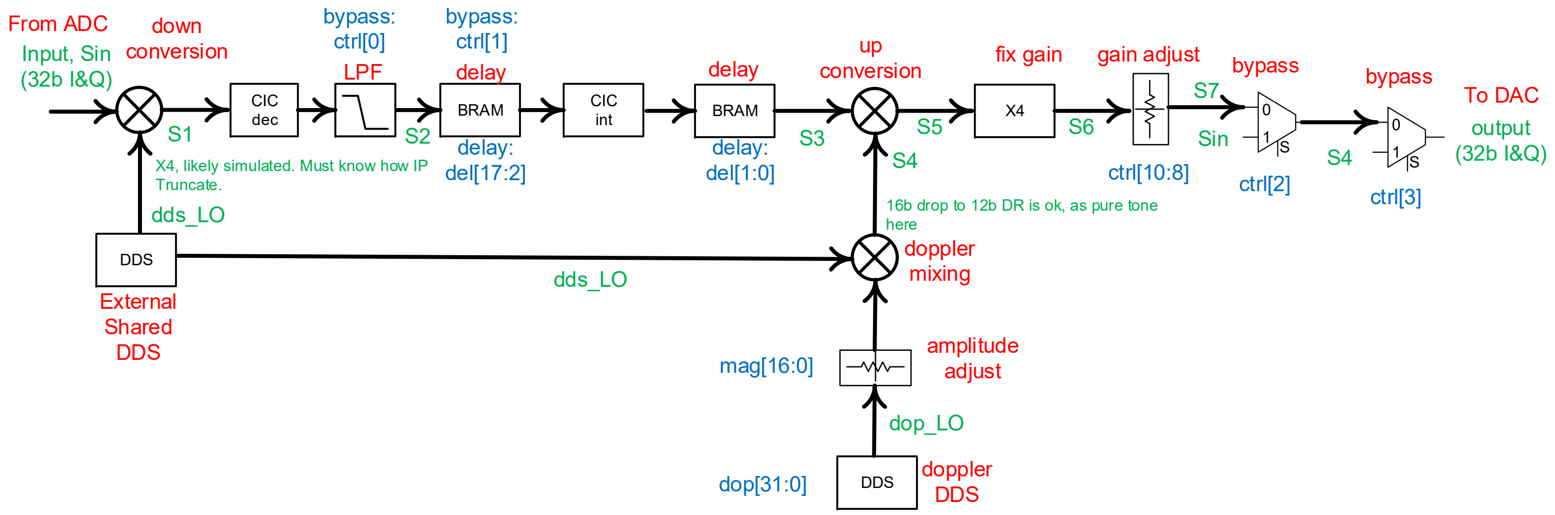
```c
/* Definitions for peripheral DAC_GPIO */
#define XPAR_DAC_GPIO_BASEADDR 0xA0070000
```


axis_broadcaster_0
S_AXIS
aclk
aresetn
M00_AXIS
M01_AXIS
M02_AXIS
M03_AXIS
M04_AXIS
AXI4-Stream Broadcaster
hier_play
axi_bram_ctrl_0
S_AXI
s_axi_aclk
s_axi_aresetn
BRAM_PORTA
AXI BRAM Controller
blk_mem_gen_0
BRAM_PORTA
BRAM_PORTB
rsta_busy
rstb_busy
Block Memory Generator
S_AXI
s_axi_aclk
s_axi_aresetn
uram_play128k_0
BRAM_A
portA_en
portA_cpu_rdata[127:0]
portA_cpu_wdata[127:0]
portA_we[15:0]
portAcpu_addr[31:0]
portA_clk
portA_rst
axis
axis_tdata[127:0]
axis_tvalid
axis_tready
axis_clk
axis_aresetn
enable
RTL
uram_play128k_v1_0
enable
axis



```verilog
  assign portA_cpu_wdata = 0;
  assign portA_we = 0;

  assign axis_tdata = reg_axis_tdata;
  assign portA_clk = axis_clk;
  assign portA_rst = ~axis_aresetn;
  assign axis_tvalid = reg_axis_tvalid;

    always @(posedge axis_clk) begin
     reg_axis_tdata <= portA_cpu_rdata;
    end

    always @(posedge axis_clk)
    begin
        if (!axis_aresetn) begin
            reg_axis_tvalid <= 0;
            portAcpu_addr <= 0;
            portA_en <= 0;
        end else begin
            if (enable == 1'b1) begin
                reg_axis_tvalid <= 1;
                portAcpu_addr <= portAcpu_addr + DWIDTH/8;
                portA_en <= 1'b1;
            end else begin
                reg_axis_tvalid <= 0;
                portAcpu_addr <= 0;
                portA_en <= 1'b0;
            end

        end
    end


endmodule
```

❑ Modify and/or insert own application



Thank you