#### **TUTORIAL: S32DS APEX VISUAL GRAPH TOOL**

S32 DESIGN STUDIO 3.1 or higher with S32V2xx development package and Vision extension package for S32V2xx







EXTERNAL USE



Interactive

**Tutorial?** 



Welcome to S32 Design Studio for Vision, Version 2.0

GETTING STARTED DOCUMENTATION VIDEO SUPPORT

#### QUICK LINKS

Getting Started 83

-S32DS--VISION-NEW Application Project NEW Library Project NEW Project from Examp

-S32DS--VGT-NEW APEX2 Program Project NEW APEX2 Kernel Project NEW APEX2 Graph Project NEW ISP Data Flow Project

-S32DS- -DDR-NEW S32V DDR Configurati Project INTRODUCTION

Continue training with video. Use training video resources from the Getting Started with the S32DS for Vision 2.0 collection case studies, if you Browse real examples of successful device programming across the Platform of all Products using CONTRACT CONT

 You can view this tutorial as a video under the VIDEO tab of Getting Started page of S32 Design Studio

Create a New APEX2 Project Create an APEX2 Project from Example Debug an APEX2 Project using Emulator Debug an APEX2 Project using Lauterbach TRACE32

Debug an A53 Project using GDB PEMicro Interface Debug an A53 Project using GDB Remote Linux

#### **Prerequisite for tutorial**

- Knowing the S32V234 SoC
- Have an understanding of the APEX architecture and APEX Core Framework(ACF)
  - Refer UG-10267-03-14-ACF\_User\_Guide.pdf to learn about ACF
    - Path:

s32ds\_installation\_directory\S32DS\software\VSDK\_S32V2\_RTM\_1\_3\_0\s32v234\_sdk\docs\ap ex\acf

· Be familiar with the Vision SDK software



#### Agenda

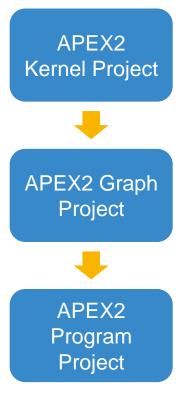
- Tutorial overview
- Make an APEX Graph (project)
- Make an APEX Program (project)
- Make a Linux application project



### WHAT ARE 3 OPTIONS AVAILABLE IN APEX GRAPH TOOL?

- APEX2 Graph Project
- APEX2 Kernel Project
- APEX2 Program Project

- APEX2 Kernel Project:
  - Kernel project is useful to define user function that works as one of the building blocks of vision pipeline
- APEX2 Graph Project:
  - Here, user can define the vision pipeline using multiple kernels
    - Vision SDK provides a broad library of built-in kernels
    - Moreover, user can build custom kernels using APEX2 Kernel Project.
  - The graph minimizes the data transfer between host and APEX, by launching multiple kernels with single data transfer. In short takes advantages of pipelining and data localization.
- APEX2 Program Project:
  - This project actually generates a source code for an APEX application
  - Once vision pipeline is ready, user can
    - Use multiple graphs to create complete APEX program flow (useful in case of dependencies between graphs)
    - Map different graphs to different APEX engines (useful if user want to run multiple graphs in parallel)
    - Choose from different image buffers (useful to transfer data between host and APEX)
- 4 EXTERNAL USE





#### **Tutorial Overview:**

- 1. We will make an APEX graph using APEX2 Graph Project option
  - -Just to make it simple we will use the kernels available in Vision SDK
    - Vision SDK provides many built-in Kernels readily available for user development.
  - -User may use the APEX2 Kernel Project option to create a brand new kernel as well
- 2. Moving forward, we will use the graph built above and make an APEX2 program project
  - As described previously, using the program project we will specify the image buffer type, select APEX engine and generate the source code for APEX engines.
- 3. Lastly, we will use this source code into our Linux application program to accelerate the performance using APEX engines.

Complete application will take a .png image, upscale and downscale it using APEX engines and return processed images



# MAKE AN APEXGRAPH

First of all we will make an APEX graph using Vision SDK kernels



#### Make an APEX2 Graph Project

- We will make a simple graph that
  - Grabs an image >> Up-scales and down-scales it by factor of 2 >> Returns processed images
- Make a new APEX2 graph project named : APEX\_VGT\_test\_graph

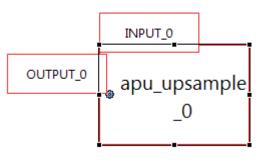
WorkspaceS32DS.3.1 - C/C++ - isp_so	nyimx224_csi_dcu/A53_	src/main.cpp - S32 Design Studio		🔀 New	— 🗆 X	Wew APEX2 Graph Project	– 🗆 X	
File Edit Source Refactor Navigate		gTools Run Creation VDK Debug Window Help		_		Create a new APEX2 Graph project		
New Open File Open Projects from File System	Alt+Shift+N >	C* S32DS Application Project Ctr	rl+Alt+E I+Alt+A rl+Alt+L	Select a wizard Create an APEX2 Graph project		Create a new APEX2 Graph project		
Close Close All Save As	Ctrl+Shift+W	Makefile Project with Existing Code  Makefile Project  C Project  C Project  Project  Project		Wizards: type filter text		Project name (APEX_VGT_test_graph		
Save All     Revert     Move     Rename     Refreth     Convert Line Delimiters To     Print     Switch Workspace     Restart     Import     Properties	F2 F5	Convert to a C/C++ Project (Adds C/C++ Nature) Conver Folder Folder Header File Header File File from Template Class Class Cother	Ctri+N option is		~	Location: C\Users\nxa17459\workspaceS32DS.3.1\A Working sets Add project to working sets Ugrking sets: Ugs default target project Target project APEX_VGT_test_graph	New       Select	Project Explorer ≈ E Outline APEX_VGT_test_graph Vision Graph : APEX_VGT_test_graph ▶ È model
1 main.cpp [isp_sonyimx224_csi_dcu, 2 ISP data flow : ISP_VGT_test_graph 3 ISP data flow : mipi_simple [resour Exit		} #ifndef _STANDALONE fflush(stdout); sleep(1); #endif // ifndefSTANDALONE		? < Back Next > Einist	Cancel	() < <u>B</u> ack <u>Next</u> > Ei	Cancel	

#### Make a graph

- Let's start building a graph...
- From the **Palette** window (right side of the S32DS window):
  - Select Add Kernels block
  - Click in the workspace to drop the block and it will ask to select kernel for that block
  - Select built-in kernel: apu\_upsample (sample\_resizing kernels)

Select an item to open (? = any character, * = any string):  Matching items:  apu_scharr_x (sample_filtering_kernels)  apu_upsample (sample_resizing_kernels)  bgr888_to_grey (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_16s (apexcv_base_image_filters)  box_tx2_08u (apexcv_base_image_filters) box_tx2_08u (apexcv_base_image_filters) box_tx2_08u (apexcv_base_image_filters) box_tx2_08u (apexcv_base_image_filters) bo	🔀 Add Kernels			×
apu_scharr_x (sample_filtering_kernels) apu_scharr_y (sample_filtering_kernels) apu_upsample (sample_resizing_kernels) bgr888_to_grey (apexcv_base_color_conversion) bilateral_Sx5 (apexcv_base_image_filters) box_1x3_16s (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters	Select an item to open (? = any character, * = any string	):		•
apu_scharr_x (sample_filtering_kernels) apu_scharr_y (sample_filtering_kernels) apu_upsample (sample_resizing_kernels) bgr888_to_grey (apexcv_base_color_conversion) bilateral_Sx5 (apexcv_base_image_filters) box_1x3_16s (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters				
apu_scharr_y (sample_filtering_kernels) apu_upsample (sample_resizing_kernels) byr888_to_grey (apexcv_base_color_conversion) bilateral_5x5 (apexcv_base_image_filters) box_1x3_16s (apexcv_base_image_filters) box_1x5_08u (apexcv_base_image_filters) box_1x5_08u (apexcv_base_image_filters) box_1x9_08u (apexcv_base_image_filters) box_1x9_08u (apexcv_base_image_filters) box_3x1 16s (apexcv_base_image_filters	Matching items:			
apu_upsample (sample_resizing_kernels) bgr888_to_grey (apexcv_base_color_conversion) bilateral_5x5 (apexcv_base_image_filters) box_1x3_16s (apexcv_base_image_filters) box_1x5_08u (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters) box_1x1_08u (apexcv_base_image_filters) box_1x1_08u (apexcv_base_image_filters) box_1x1_16s (apexcv_base_image_filters) box_1x1_16s (apexcv_base_image_filters) box_1x1_08u (apexcv_base_image_filters	apu_scharr_x (sample_filtering_kernels)			^
bgr888_to_grey (apexcv_base_color_conversion) bilateral_5x5 (apexcv_base_image_filters) box_1x3_165 (apexcv_base_image_filters) box_1x5_08u (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters) box_1x1_08u (apexcv_base_image_filter	apu_scharr_y (sample_filtering_kernels)			
bilateral_5x5 (apexcv_base_image_filters) box_1x3_16s (apexcv_base_image_filters) box_1x5_08u (apexcv_base_image_filters) box_1x7_08u (apexcv_base_image_filters) box_1x0_08u (apexcv_base_image_filters) box_3v1 16s (apexcv_base_image_filters) V	apu_upsample (sample_resizing_kernels)			
box_1x3_16s (apexv_base_image_filters)     box_1x5_08u (apexv_base_image_filters)     box_1x7_08u (apexv_base_image_filters)     box_1x9_08u (apexv_base_image_filters)     box_1x9_08u (apexv_base_image_filters)     box_11s9_08u (apexv_base_image_filters)	bgr888_to_grey (apexcv_base_color_conversion)			
box_1x5_08u (apexcv_base_image_filters)     box_1x7_08u (apexcv_base_image_filters)     box_1x9_08u (apexcv_base_image_filters)     box_1x9_08u (apexcv_base_image_filters)     box_11s6 (apexcv_base_image_filters)	bilateral_5x5 (apexcv_base_image_filters)			
box_1x7_08u (apexcv_base_image_filters) box_1x9_08u (apexcv_base_image_filters) box_1x9_08u (apexcv_base_image_filters) box_3v1 16s (apexcv_base_image_filters)	box_1x3_16s (apexcv_base_image_filters)			
box_1x9_08u (apexcv_base_image_filters)     box 3v1 1fs (apexcv_base_image_filters)     ✓	box_1x5_08u (apexcv_base_image_filters)			
B hov 3v1 16s (anevov base image filters)	box_1x7_08u (apexcv_base_image_filters)			
hov Rv1 16s (anevcv hase image filters)				
C\NXP\S32DS31\S32DS\config\s32v234_e_resizing_kernels\upsample_apukernel				
enter (b) config (b) c	C:\NXP\S32DS.3.1\S32DS\config\s32v234e_resizing_ke	ernels\upsan	nple.apuk	ernel
	(?)	<	Cance	1







Palette

Input

1uqtuO

Connector

Kernel Definition

Add Kernels

📐 🔍 🔍 📁 • 🛃 •

Graph Definition

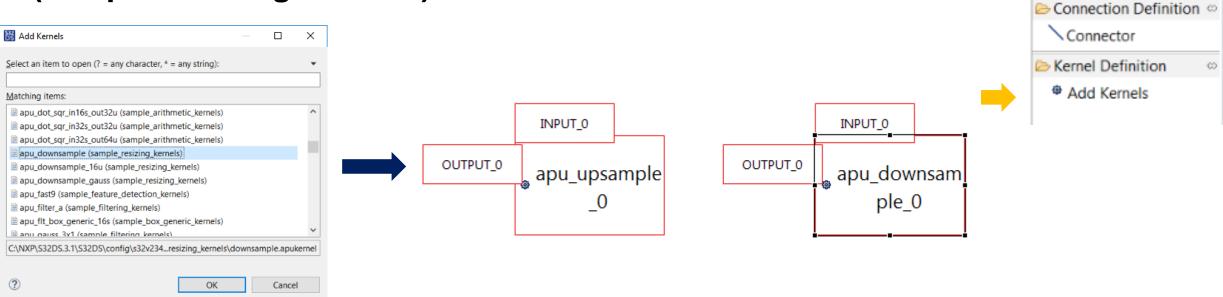
Connection Definition

-60

60

#### Make a graph

 Similarly, create a block for kernel: apu\_downsample (sample\_resizing\_kernels)





Palette

Input

Output

📐 🔍 🔍 📁 • 🛃 •

Graph Definition

Þ

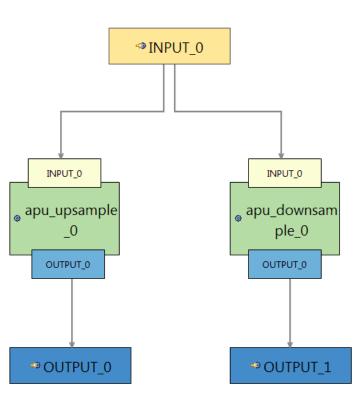
60

#### Make a graph

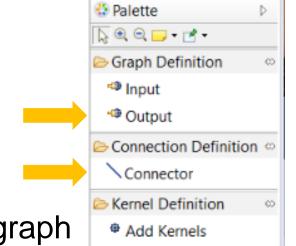
OUTPUT 1

Select Output block and include it into the graph

• Using **Connector**, connect graph blocks to create the following graph







- You do **not** need to configure any properties for the graph!
- APEX Core Framework (ACF) will take care of this!

-e.g. ACF extracts Image size description from image buffer automatically

• It is a generic graph and can be used in any application without any application specific modification such as image size description.



#### **Change the Block Name**

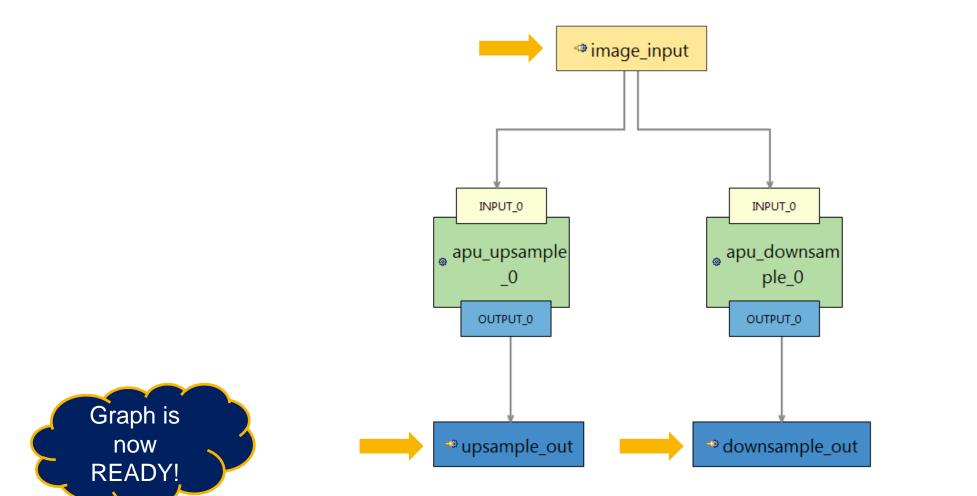
- For better readability, user may want to change the block names
- Select a **Block** from the graph and look at the **Properties** window (on your left)
- Change the Name here, try changing the INPUT block

<b>Input0</b>		
Properties 🔀	<b>1</b>	~ - 8
🖘 Input: Input0		
Properties   Properties		
Appearance Name:	Input0	
Graph:	APEX_VGT_test_graph	
Element Type:	d08u	



#### **Change the Block Name**

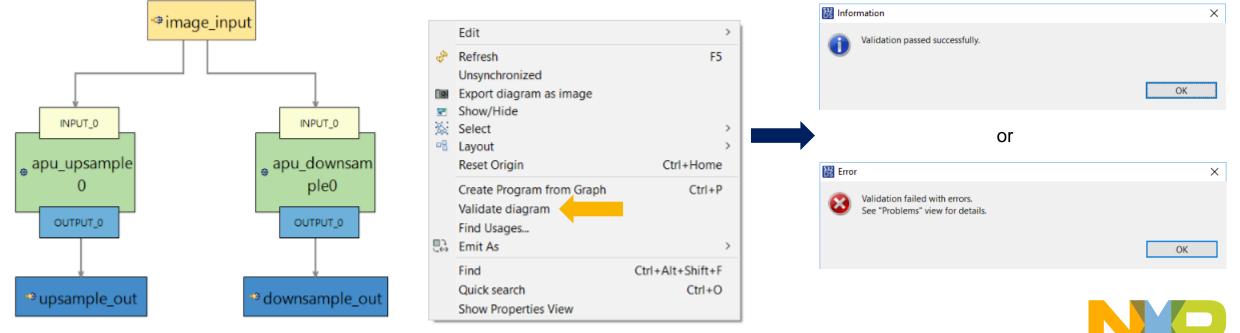
- · Let's do the same for both OUTPUT blocks
- Now graph should look like this...





#### Validate graph for correctness

- Save the graph
- Right Click anywhere in the white part of the graph
- Validate graph
  - -You will see a pop-up window showing status of validation.



#### **Validation Error**

- Error will be indicated by red cross on the block and description can be seen in the **Problems view**
- Since there are no parameters to configure, there are only two type of errors
  - Missing connections
  - Duplicate names
- Block color will remain white in case of errors
- Correct the error, Save the graph and try to Validate again



# MAKE AN APEX PROGRAM

Once APEX graph is ready, we will use this graph to make an APEX program. Here, we will map this graph to one of the APEX engines, define the image buffers and generate source code.



#### Make a Linux application project with APEX program project

- We will make a program project but it requires interaction with Linux application project
- So, we will make a Linux application project
  - the program project will be created automatically

1. Go to File -> New -> S32DS Application Project

File	) Edit	Diagram	Navigate	Search	Project	Run	Processor	Expe	t Window	Help	
	New						Alt+Shift+N	> N		R Configuration Project	Ctrl+Shift+O
۵,	Open Open		om File Syst	em			•		S32DS Ap	opicet from Example oplication Project	Alt+B, Al
	Close Close	All				С	Ctrl+W trl+Shift+W	- III'	APEX2 Gr	aph Project low Project	AIC 0, 1
	Save Save / Save /					(	Ctrl+S Ctrl+Shift+S		APEX2 Ke	ernel Project ogram Project	
	Rever	t							· · ·		
r4	Move. Renar						F2		Other		Ctrl+N
	Refree Conve	sh ert Line Del	imiters To				F5				
۵	Print Page !	Setup					Ctrl+P				
	Switch Restar	h Workspac rt	e					•			
èn Èn	Impor Expor										
	Prope	rties					Alt+Enter				
	2 APE 3 APE	X2 Progran X_VGT_test	_program.a n Diagram : _graph.aird APEX_VGT_t	APEX_VO	GT_test /GT]						
	Exit										

- 2. Type the project name: APEX\_VGT\_test\_application
- 3. Select project type as shown
- 4. Hit Next

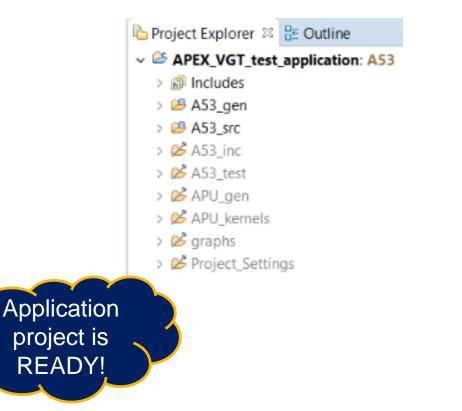
Create a S32 Design Studio Project					
New S32DS Application Project					
Project name:					
APEX_VGT_test_application					
Use default location					
Location: C:\Users\nxa17459\workspaceS32[	DS.3.1\APEX_VG	[_test_applicatio	n	Brows	se
Processors:	ToolChain Se	ection:			
type filter text	Core Kind	Name	Toolchain		
✓ ➢ Family S32V2	A53	Cortex-A53	ARM Linux 64-b	oit Target Binan	y Tc in
S32V232 Cortex-M4					
S32V232 Cortex-A53					
S32V232 Cortex-A53 Linux	<				3
<ul> <li>S32V234 Cortex-A53 APEX2/ISP Lir</li> <li>S32V234 Cortex-A53</li> </ul>	Description:				
S32V234 Cortex-A53	· · · · ·	AE2 project wit	h APEX2/ISP supp	ort for CCC 6 7	,
S32V234 Cortex-M4	toolchain	-A35 project wit	III APEA2/ISP Supp	Joit for GCC 0.3	·
-					

5. Since we are not using ISP, **deselect unnecessary options** as shown

S32DS Application Project	:		
New S32DS Project for S	32V234 Cortex-A53 APEX2/ISP Linux		
Select required cores and pa	rameters for them.		
Project Name	APEX_VGT_test_application		
Core	Cortex-A53		
SDKs	VSDK_MODULE_WIN		
Debugger	SQB Remote Linux Debugger		
APEX2 programming			
ISP programming			
ISP visual modeling			

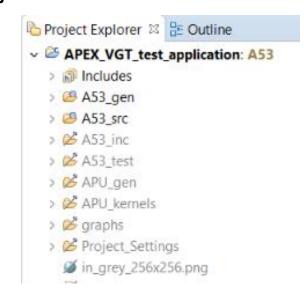
#### Make a Linux application project with APEX program project

 We can now see the project in the Project Explorer



 Copy the picture "in\_grey\_256x256.png" from

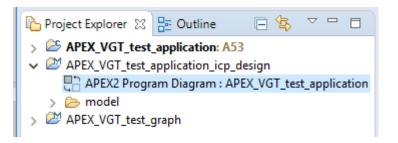
s32ds\_installation\_directory\S32DS\software \VSDK\_S32V2\_RTM\_1\_3\_0\s32v234\_sdk\de mos\data\common to the project folder



We will stop here with application project and get back to "configure block properties"

#### Make an APEX2 Program Project

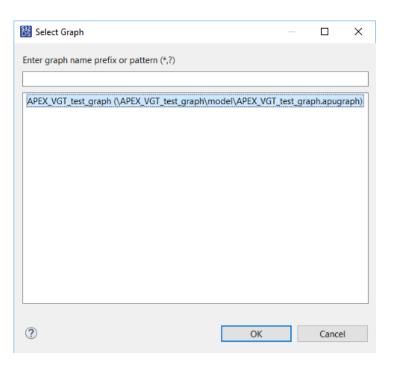
- · We will now make a program project and specify
  - graph we want to use and its mapping on specific APEX engine
  - -image buffer type
- Open the APEX2 Program Diagram



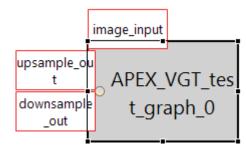


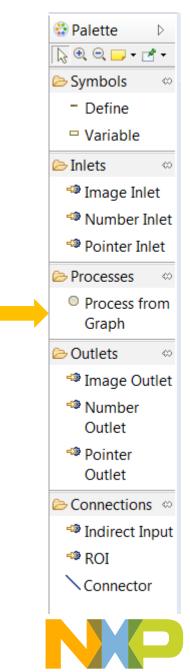
#### Make an APEX program

- From the **Palette** window (right side of the S32DS window):
  - Select Process from Graph block
  - Click in the workspace to drop the block and it will ask to select graph available in the current workspace
  - Select "APEX\_VGT\_test\_graph" that we just created



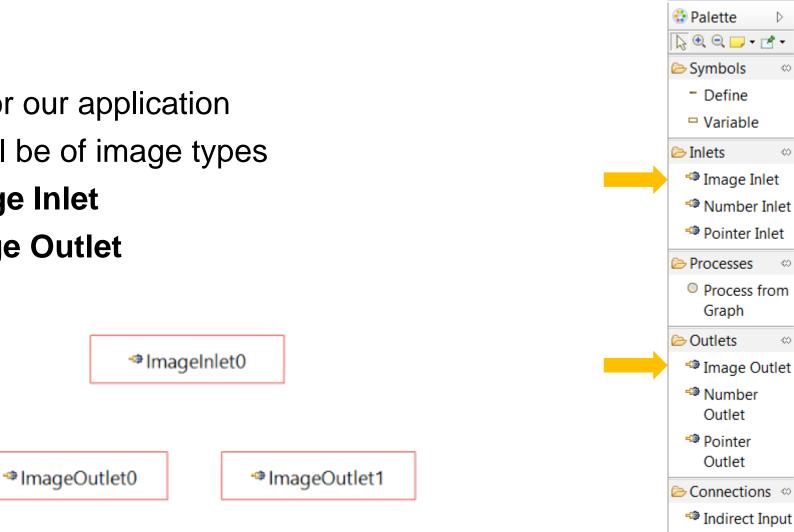






#### Make a program graph

- We will use a .png image for our application
- So, our input and output will be of image types
- Select and create one Image Inlet
- Select and create two Image Outlet



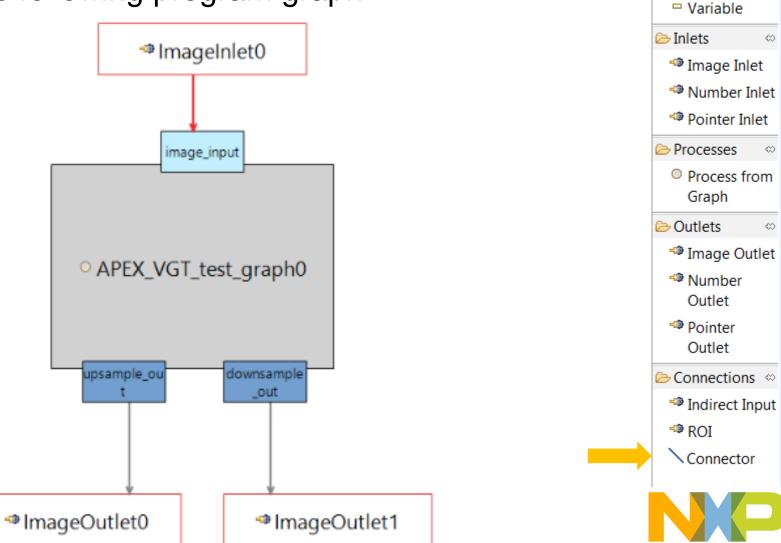
🕫 ROI

Connector



#### Make a program graph

Connect blocks to create the following program graph



Palette

➢ Symbols

Define

📐 🔍 🔍 📁 + 🛃 +

 $\triangleright$ 

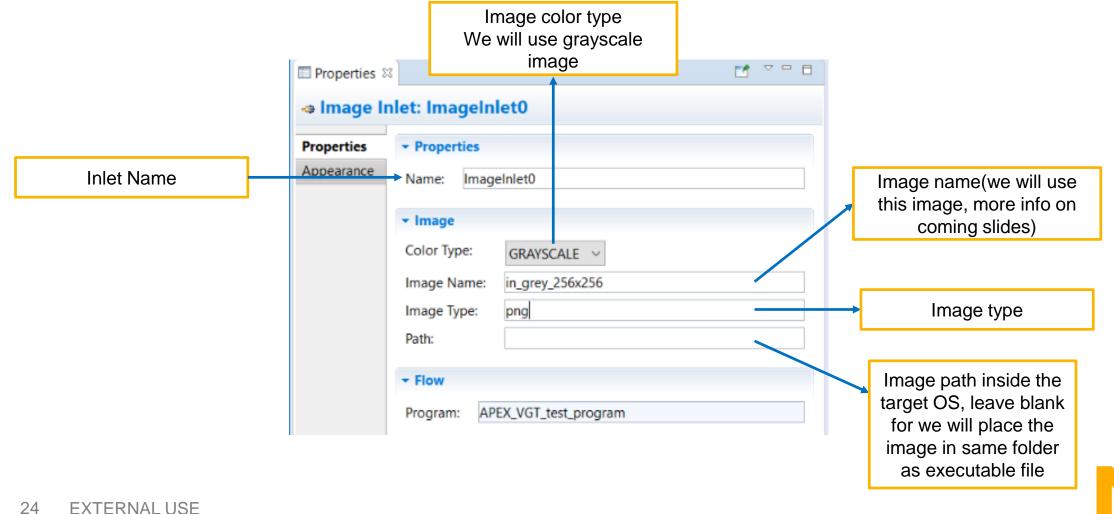
00

22 EXTERNAL USE

- Now we have Linux application project ready. We will get back to configuring block properties of program project
- Select the **ImageInlet0** block
- Look at (open if not visible) the properties window

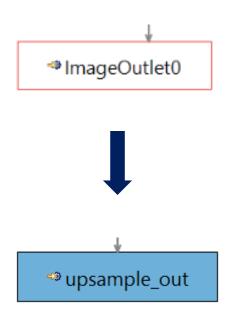
is workspaceS32DS.3.1	Visual Graph Tools - platform:/resource/APEX_VGT_tes	t_progra	ram/model/APEX_VGT_test_program.aird/APEX2 Program Dia	agram : APEX_VGT	_test_program - S32 Desig	gn Studio		- 0	I X
<u>F</u> ile <u>E</u> dit <u>D</u> iagram <u>N</u> a	vigate Se <u>a</u> rch <u>P</u> roject ConfigTools <u>R</u> un <u>C</u> reation	<u>v</u> dk d	Debug <u>W</u> indow <u>H</u> elp						
📑 👻 🔚 🐚 📄 🛩 😒	🗈 🛨 🔚 🐚 🛛 🖗 🏷 🛛 🛷 🕶 🕼 🖓 🕶 🕼 🐨 🗰 💌 🗊 Create APEX2 Projects 🐨 📫 Create ISP Projects 🐨 🖗 🐨 🖓 🐨 🌾 😓 🐨						C	Quick Access	21 🔤 🔛
Project Explorer 😂 🔡	Outline 🖻 😓 🔻	🖻 🄄 🍸 🗖 🗖 🎄 Vision Graph : APEX_VGT_test_graph 🛛 🞄 *APEX2 Program Diagram : APEX_VGT_test_program 🛛 🖻 main.cpp							
> 🐸 APEX_VGT_test_app	lication: A53		=== + 🏭 + 🎮 + 🖃 📧 📾 🐚 🔌 💥 🉀 B I	A - 🖄 🖣	ø ▾ → ▾ 🖪 🕫 🖽   8	8 <b>8</b> 3		😳 Palette	⊳
> APEX_VGT_test_app							~	📡 🔍 🔍 🖵 • 🖻	f -
> APEX_VGT_test_gra								읃 Inlets	0
APEX_VGT_test_pr APEX2 Program	ogram Diagram : APEX_VGT_test_program							Image Inle	et
> 🗁 model	Diagram Arex_vor_cost_program		<ul> <li>ImageInlet0</li> </ul>	I				Number In	nlet
<								Pointer Inf	let
								Processes	00
Properties      S			image_input					Process fro	om Graph
😣 Image Inlet: Im	ageInlet0 The 'Image name shouldn't be empty'							🗁 Outlets	\$
Descention - Desce								Image Out	tlet
Properties - Prope								Number O	
Appearance Name:	ImageInlet0		<ul> <li>APEX_VGT_test_g</li> </ul>	raph0				Pointer Outlet	
✓ Imag								Connections	
								Indirect In	put
Color T	GRAYSCALE ~		upsample_ou dow	nsample				ROI	
Image	Name:			_out				Connector	7
Image	Type:								
Path:			*	*					
			ImageOutlet0	ImageOutlet	1				
✓ Flow							~		
Program	n: APEX_VGT_test_program		<				>		
✓ Input	s		Problems 🛙						~
			0 items						
49 Pro	ocess Input: APEX_VGT_test_graph0.image_input		Description	Resource	Path	Location	Туре		
		÷.							
		÷-							
ono simulation	Synchronized diagram					1			

• ImageInlet0 block : Configure the properties like follow:





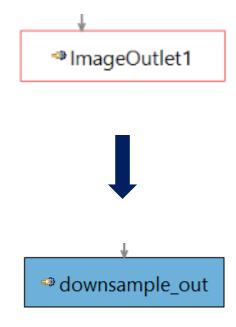
Select ImageOutlet0 block



		Outlet	Name			
Properties 🛛	3		t	3 ~ - 8		
🗠 Image O	utlet: upsamp	ole_out				
Properties	▼ Properties					
Appearance	Name: upsar	nple_out	l			Up-sampled image will take this name
	▼ Image					
	lmage Name: Image Type: Path:	out_grey_512 png	2x512		, ,	Image type
	Flow					
	<ul> <li>Program:</li> <li>Process Output:</li> </ul>		_test_applicatio _test_graph0.up			Image path inside the workspace, again, we will leave it blank

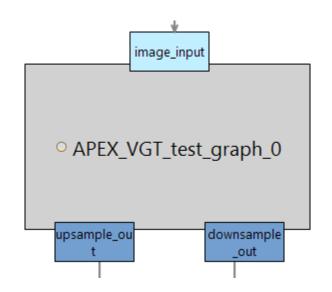


#### Similarly, select IMAGE\_OUTLET\_1 block



		Outlet Name	]		
🔲 Properties 🖇	3				
🔹 Image O	utlet: downsa	ample_out			
Properties	▼ Properties				
Appearance	Name: dowr	nsample_out			Down-sampled image will take this name
	▼ Image			/ -	
	lmage Name: Image Type:	out_gray_128x128 png			Image type
	Path:			L	
	▼ Flow			$\backslash \Gamma$	
	Program: Process Output:	APEX_VGT_test_application APEX_VGT_test_graph0.dow	nsample_out		Image path inside the workspace, leave it blank

- Now, we will select on which APEX we want to run our graph
- Select APEX\_VGT\_test\_graph\_0 block



Properties			
Appearance	Unit: APEX Graph: APEX Program: APEX <pre> • Inputs:</pre>	_VGT_test_graph0 CVGT_test_graph CVGT_test_program : APEX_VGT_test_graph0.image_input	Select the APEX on which you want to run the graph
		ut: APEX_VGT_test_graph0.upsample_ou ut: APEX_VGT_test_graph0.downsample	

- You do not need to configure the image\_input properties
  - But, if you want to optimize graph performance, you can hand-tune chunk size of the input. (See APEX documents for more information)
  - Otherwise, leave chunk size to "zero" and ACF will decide chunk size automatically

	Properties 🛛				
	Process	Input: APEX_VGT_test_graph0.image_input			
	Properties	▼ Properties			
image_input	Appearance	Name: image_input			
		▼ Data			
		Columns: 1			
		Rows: 1			
		▼ Umat			
<ul> <li>APEX_VGT_test_graph0</li> </ul>		UMat Type: VSDK_CV_8UC1 ~			
		* Flow			-
		Process: APEX_VGT_test_graph0	*	Chunk Size X:	0
		Graph Source: image_input E0 Data Type: d08u		Chunk Size Y:	0
downsample_o		Memory Part: VEC			
upsample_out ut		Program Inlet: IMAGE_INLET_0			
		Process Output:			
1 1		✓ Chunks			
		Chunk Size X: 0			
		Chunk Size Y: 0			
	<b>– –</b>				



- We will now specify the output image buffers size
- Select upsample\_out and downsample\_out of APEX\_VGT\_test\_graph\_0 block
- Modify its properties as shown below

Properties	E Properties ∞	3
		Output: APEX_VGT_test_graph0.downsample_out
Properties         Properties           Appearance         Name: upsample_out	Appearance	Properties     Name: downsample_out
✓ Data Columns: 512		✓ Data Columns: 128
Rows: 512		Rows: 128
progra	im 🗸	
	Process Output: APEX_VGT_test_grap Properties Appearance • Properties Name: upsample_out • Data Columns: 512 Rows: 512 Columns: 512 Columns: 512	<ul> <li>Process Output: APEX_VGT_test_graph0.upsample_out</li> <li>Properties</li> <li>Appearance</li> <li>Name: upsample_out</li> <li>Data</li> <li>Columns: 512</li> </ul>

now READY!

29

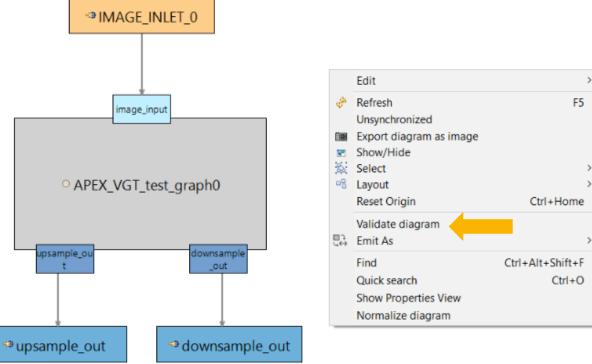
**EXTERNAL USE** 



6 of 6

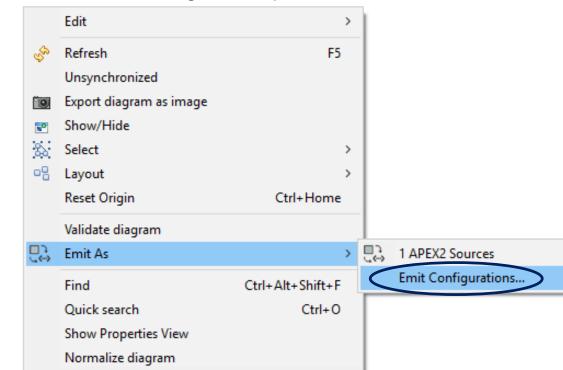
#### Validate graph for correctness

- Save the graph
- Right Click anywhere in the white part of the graph window
- Validate graph
- Validation errors are reported and can be taken care of in a same way as explained before



#### Select the destination of autogenerated source code 1 of 3

- By default all source code will be generated inside the APEX program project itself
- We can reconfigure the destination of source code to any other open projects.
  - -We will use this feature and generate the source code in Linux application project.







#### Select the destination of autogenerated source code 2 of 3

- Define a new configuration and specify where we want to generate our source code.
  - 1. Change the configuration as shown in the picture
  - 2. Click on Apply to save the changes

	Name: emit_to_application
type filter text	🖳 Main 🔗 Refresh 🔠 Common
<ul> <li>APEX2 Emitter</li> <li>APEX2 sources APEX_VGT_test_application_icp_design - APEX_VGT_test_application</li> <li>New_configuration</li> <li>ISP Emitter</li> </ul>	Source type
	/APEX_VGT_test_application_icp_design/model/APEX_VGT_test_application.apuprogram Browse Workspace
	Output
	S{workspace_loc:/APEX_VGT_test_application}
	Browse Workspace Browse File System Varia <u>b</u> les
	Emit kernels
	APU_kernels
	Emit graphs
	graphs
	Emit host
	A53_gen
	Emit test
	A53_test



#### Select the destination of autogenerated source code

#### • Edit some more configuration

- 3. Go to Common tab.
- 4. Select the \APEX\_VGT\_test\_application\.launches folder under "Shared files" option here.
- 5. Apply the settings and Hit Emit button to generate a source code at the designated location

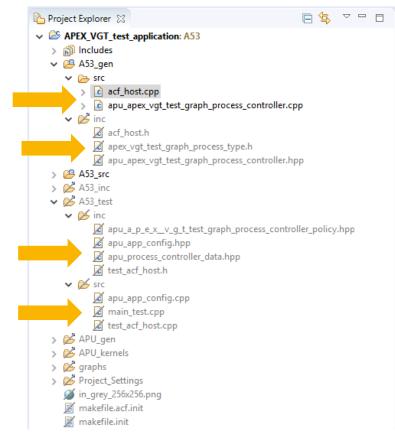
Emit Configurations	×
Create, manage, and run configurations	
Create a configuration to emit APEX2 sources	للم الم الم الم الم الم الم الم الم الم
Image: Second	Name:       emit_to_application         h       Common         Save as       O Local file         Image: Shared file:       \APEX_VGT_test_application\.launches         Browse       Browse
	Display in favo <u>r</u> ites menu Encoding
	Console (necessary for input)
	Input Eile:
	Workspace     File System     Variables
	Workspace File System Variables
	☐ Launch in bac <u>k</u> ground
Filter matched 4 of 4 items	Re <u>v</u> ert Apply
?	Emit Close



3 of 3

#### **Emit the source code**

- Auto generated code can be seen inside the project folder
  - Note: If you can not see source code, please right click on the project and click on **Refresh** from the menu.





# LINUX APPLICATION PROJECT FOR APEX

Once APEX code is ready, we will now build our Linux application



#### **Application Code for APEX**

- Basic, auto generated, application code template for APEX can be found in A53\_test/src/test\_acf\_host.cpp
- ACF\_APP\_CALL() inside the A53/src/main.cpp is just a place holder.
- User should copy/add/change code inside the A53\_gen/acf\_host.cpp and A53/src/main.cpp according to his/her application needs or structure
- User should use build config TEST\_A53 for this tutorial example, and during early APEX graph development. Later, once host code is developed and added to A53\_gen/acf\_host.cpp, the user should switch to build config A53. If you build A53 without the host code, the project may build and execute on target without error, but the application will not be run.
  - -Note: In this tutorial we will not change default structure as it is not necessary



#### **Application code for APEX: Compile**

- In this tutorial, we do not need to add/change any default code
- Go to C/C++ perspective and compile the application for TEST\_A53





### Execute your APEX\_VGT\_test\_application.elf binary

### on the target!

#### **Connect and Observe**

- Do not forget to copy *in\_grey\_256x256.png* to the same directory where you copied binary
- Run the application

38

**EXTERNAL USE** 

- You can find 2 new image files generated in the same directory
  - One will be doubled the size of the original image and the
  - other will be half the size of the original image





#### SECURE CONNECTIONS FOR A SMARTER WORLD