



**FTF 2016**  
TECHNOLOGY FORUM

# HOW TO USE AUTOMOTIVE MCU VIRTUAL MODELS

**FTF-AUT-N1799**

MANFRED THANNER, NXP SYSTEMS MODELING EXPERT

CURT HILLIER, NXP APPLICATIONS ENGINEER

MOJIN KOTTARATHIL, SYNOPSISYS

ANEESH BHASIN, SYNOPSISYS

MAY 18, 2016

PUBLIC USE



# AGENDA

- Course Overview
- Introduction to Virtual Prototypes
- Demos:
  - Linux boot
  - A53 Test Software
- Hands on labs:
  - Modifying source code
  - Using debugger run control
- Conclusion



# COURSE OVERVIEW



# Course Overview

- Introduction to Virtual Prototypes
  - What are Virtual Prototypes?
  - Why are they useful?
  - Explain the NXP and Synopsys COE
- Demos
  - See next slide
- Hands-on Labs
  - See next slide

# Today's Demo and Hands-on Lab Sessions

- **DEMO: Startup** the S32V234 Virtual Prototype in the Synopsys Virtual Development Kit (VDK) environment
- **DEMO: Boot Linux** on the S32V234 based on ARM® Cortex®-A53. Observe boot process in the Linux console window
- **DEMO: Execute Test Software** on the S32V234 processor based on Cortex-M4.
  
- **HANDS-ON:** Configure Function and Context tracing, display in the VP Explorer
- **HANDS-ON: Bring up** a Lauterbach debugger and control the run-time operation of the virtual model – just like you would control a real Hardware EVB
- **HANDS-ON: Modify code**, recompile, and see your changes in action

# INTRODUCTION TO VIRTUAL PROTOTYPES



# NXP and Synopsys – Virtual Prototyping

Virtual Prototyping

**SYNOPSYS**<sup>®</sup>  
Silicon to Software<sup>™</sup>



**Accelerate Innovation with Earlier and Faster Software Development**

<TO DO: Insert slide on the Synopsys + NXP COE agreement.>



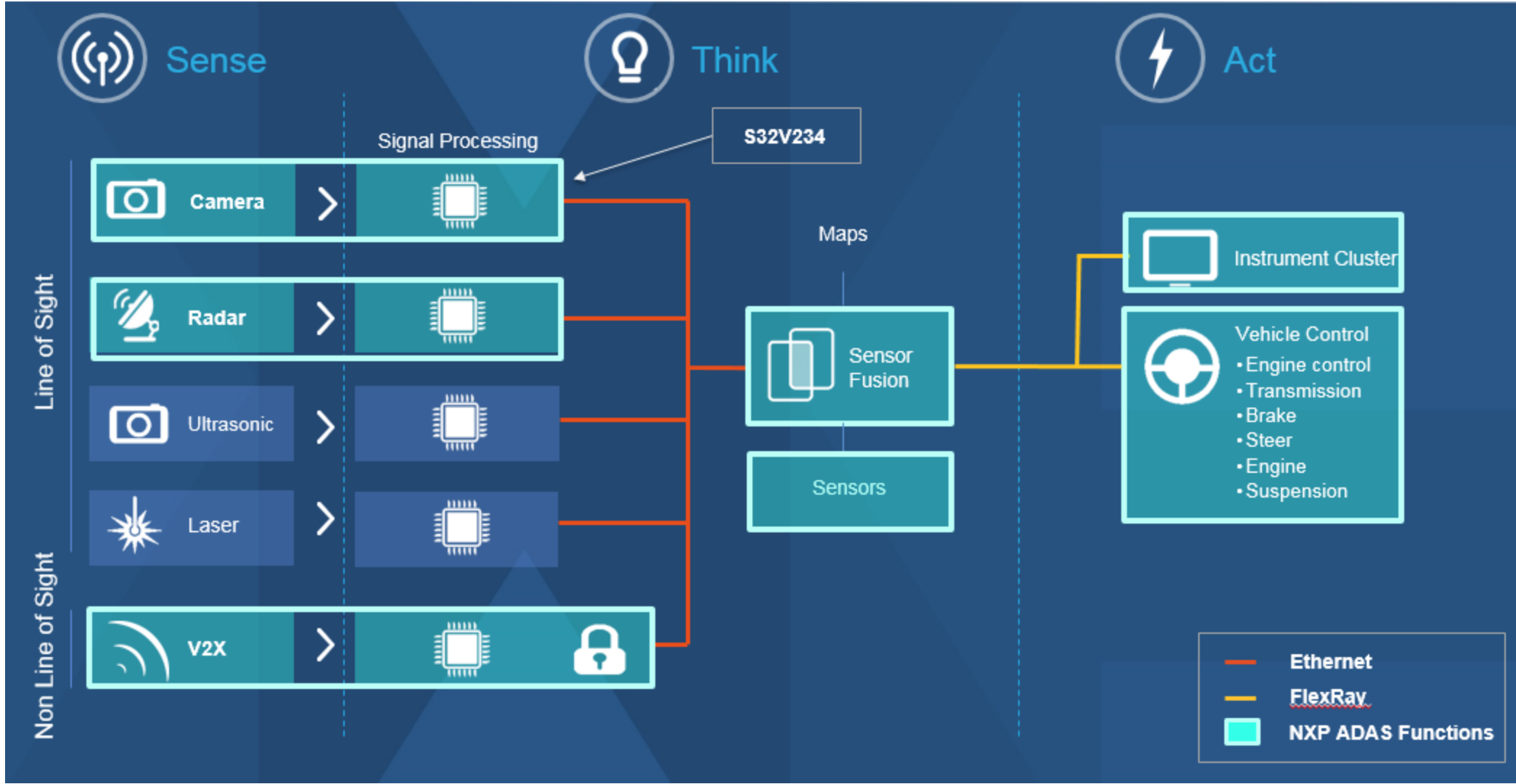
# Introduction

- What is a virtual prototype?
  - “Virtual Prototyping is the use of computer models to develop and test a process or component without having to physically build or run it.” FLUXTROL, Inc.
- Why do I need to know?
  - Virtual Prototyping is becoming more commonplace in engineering environments as the demands for performance modeling and early software development and testing continue to increase.
- What can I do with a virtual prototype?
  - “Virtual prototyping results in faster time-to-market through earlier and faster software development and improved communication throughout the supply chain. They enable software engineers to start development months before the hardware design is complete, enabling full system bring-up to occur within days of silicon availability. Virtual prototypes are fast, fully functional software models of complete systems that execute unmodified production code and provide unparalleled debug efficiency.
  - See more at: <http://www.synopsys.com/prototyping/virtualprototyping/Pages/default.aspx#sthash.5y7mdELS.dpuf>

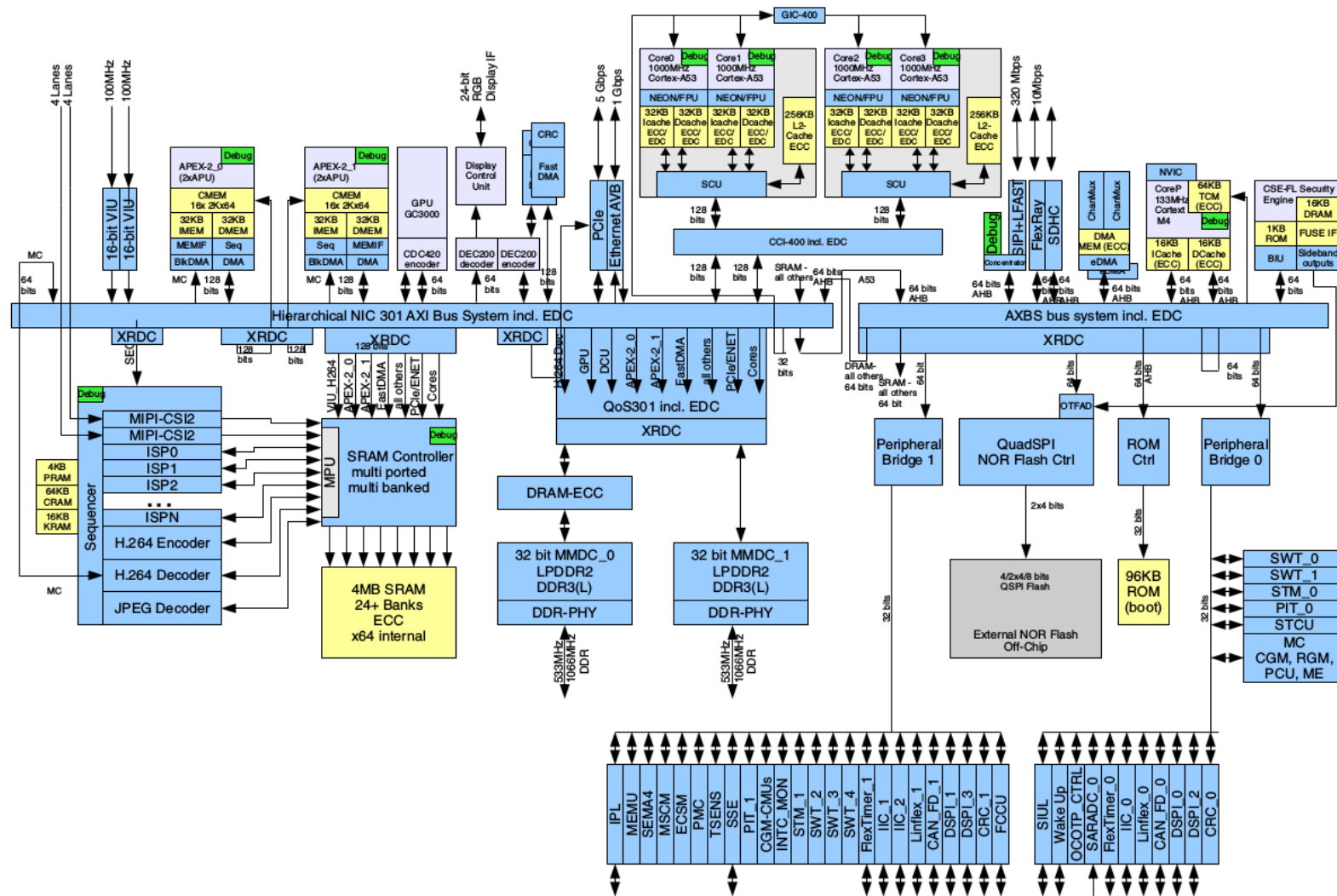
# Introduction to the Class

We will analyze the S32V234 vision processor shown on the next slides.

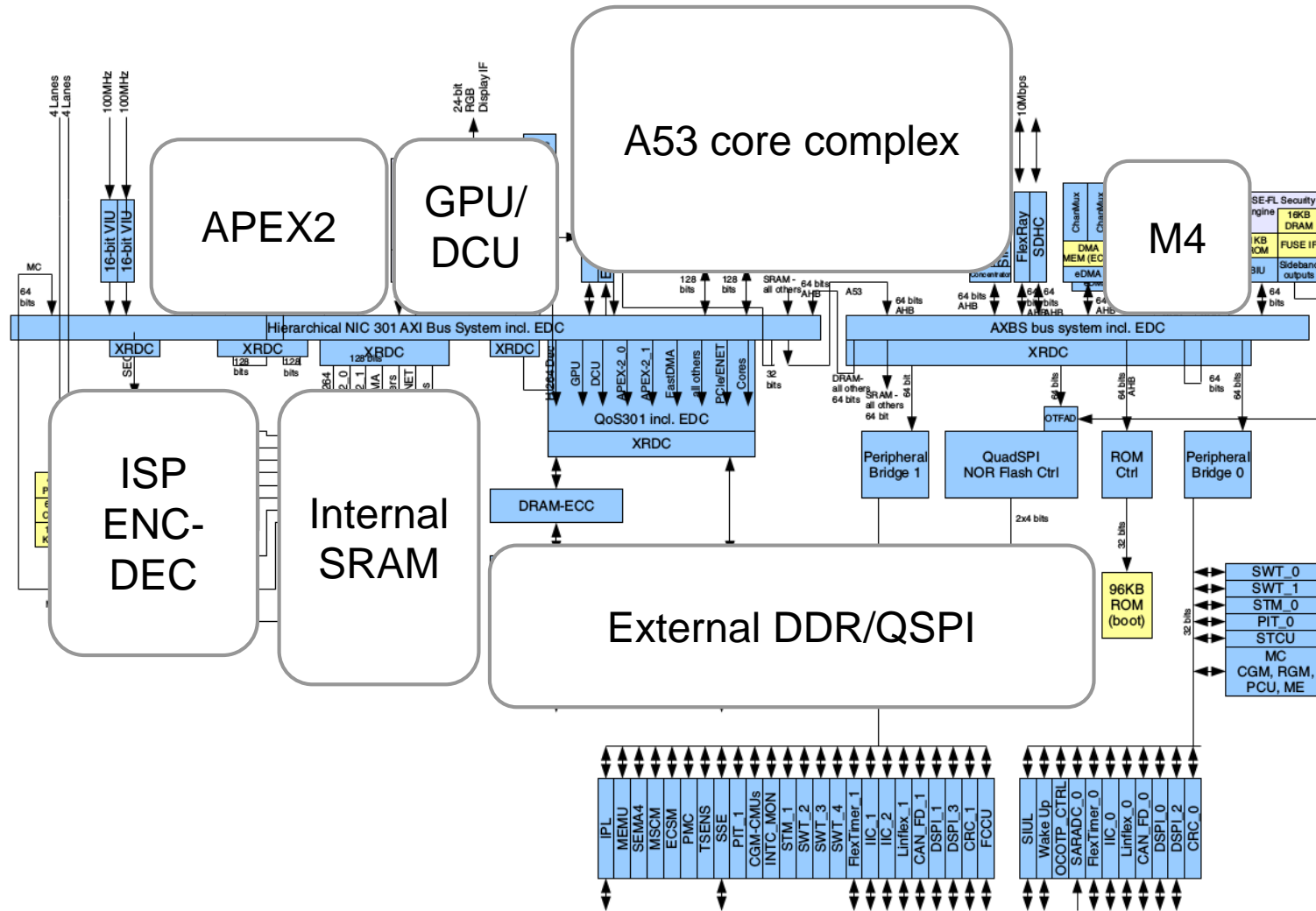
# ADAS System Solution



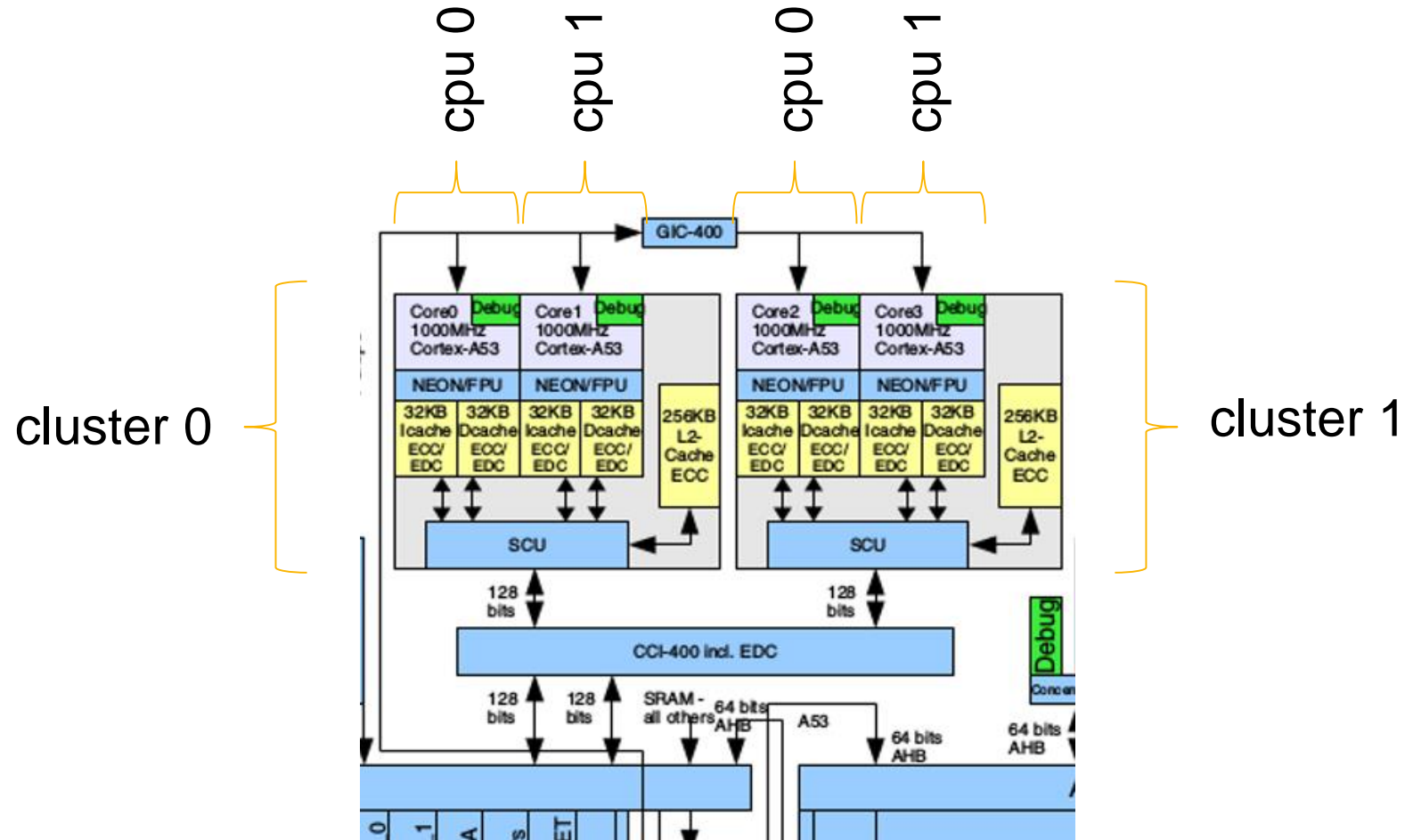
# S32V234 Block Diagram



# S32V234 Block Diagram



# S32V234 based on -A53 Cluster Definition



# DEMOS AND HANDS ON LABS





Start synopsis VP Explorer  
PPT FILES

# GETTING STARTED WITH YOUR S32V234 VIRTUAL PROTOTYPE



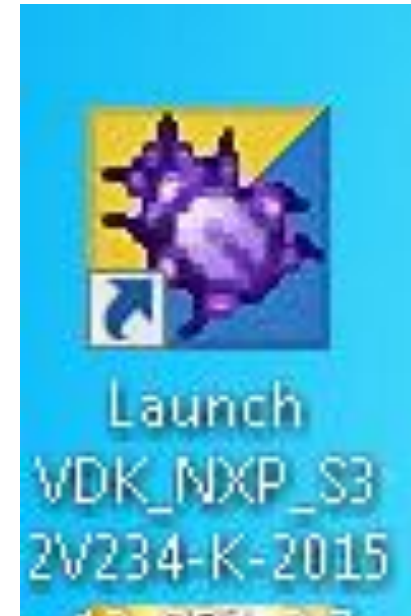
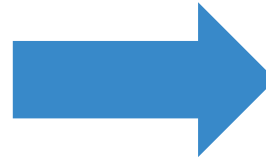
SECURE CONNECTIONS  
FOR A SMARTER WORLD



**<TO DO: Get slides from Mojin and Aneesh for Linux boot>**

# Launch VP Explorer

- Double click this icon

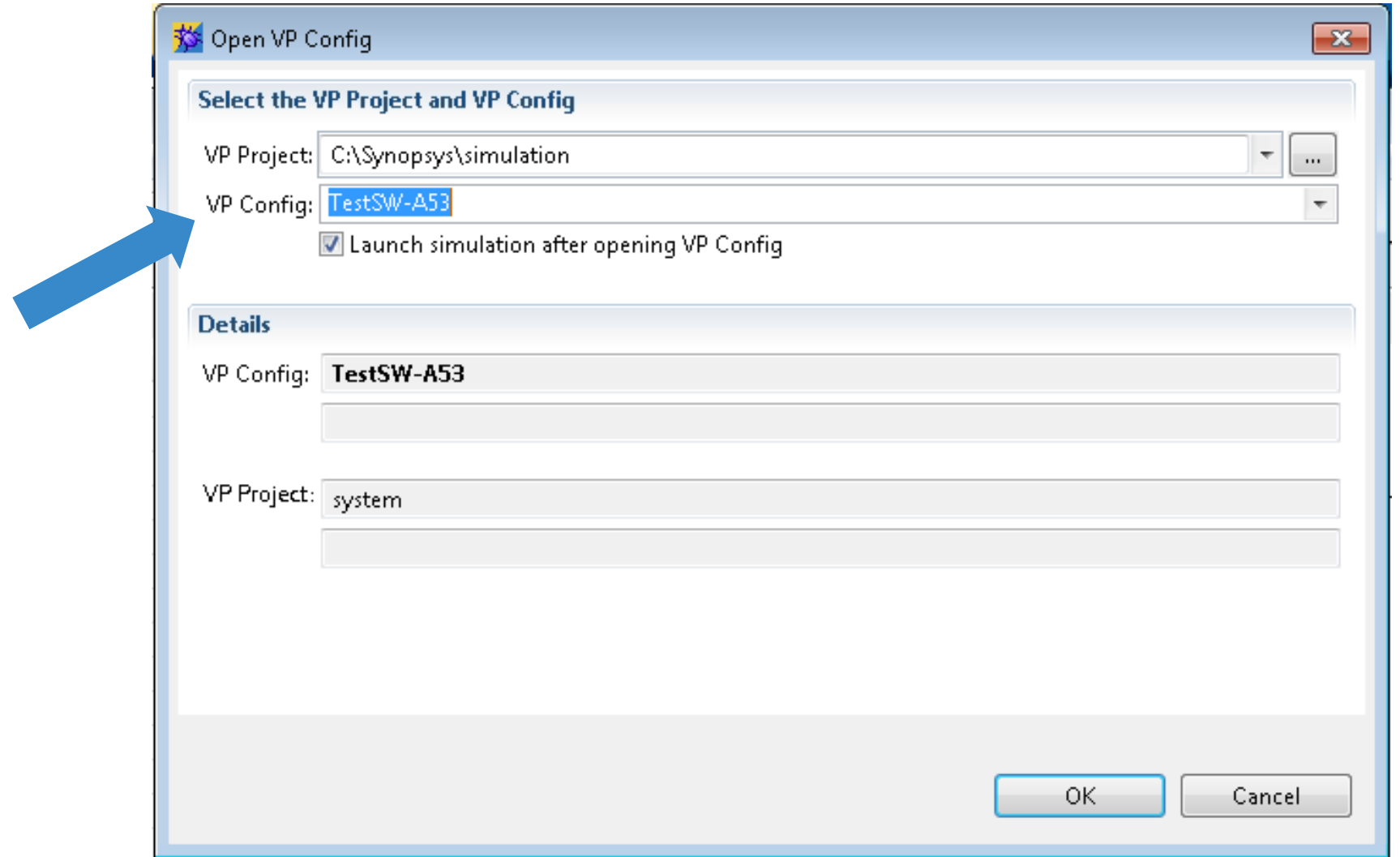


# VP Explorer

- VP Explorer (vpx) is launched by default and starts the virtual prototype simulation.
- VP Explorer is primarily intended to:
  - Run and debug SystemC based simulations in depth.
  - Control simulation execution.
  - Trace and analyze simulation output during or after simulation.
- The VP Explorer controls the execution of the virtual prototype, like, suspending the entire virtual prototype. In this case, the entire platform state is frozen, including all timers and clocks. On resuming the execution, the virtual prototype continues from that exact point. As suspending the simulation has no impact on the platform state, it is called non-intrusive.

# A53 Test SW

- Select the TestSW-A53 skin from the VP Config: drop down box
- Click OK







# Check Your System

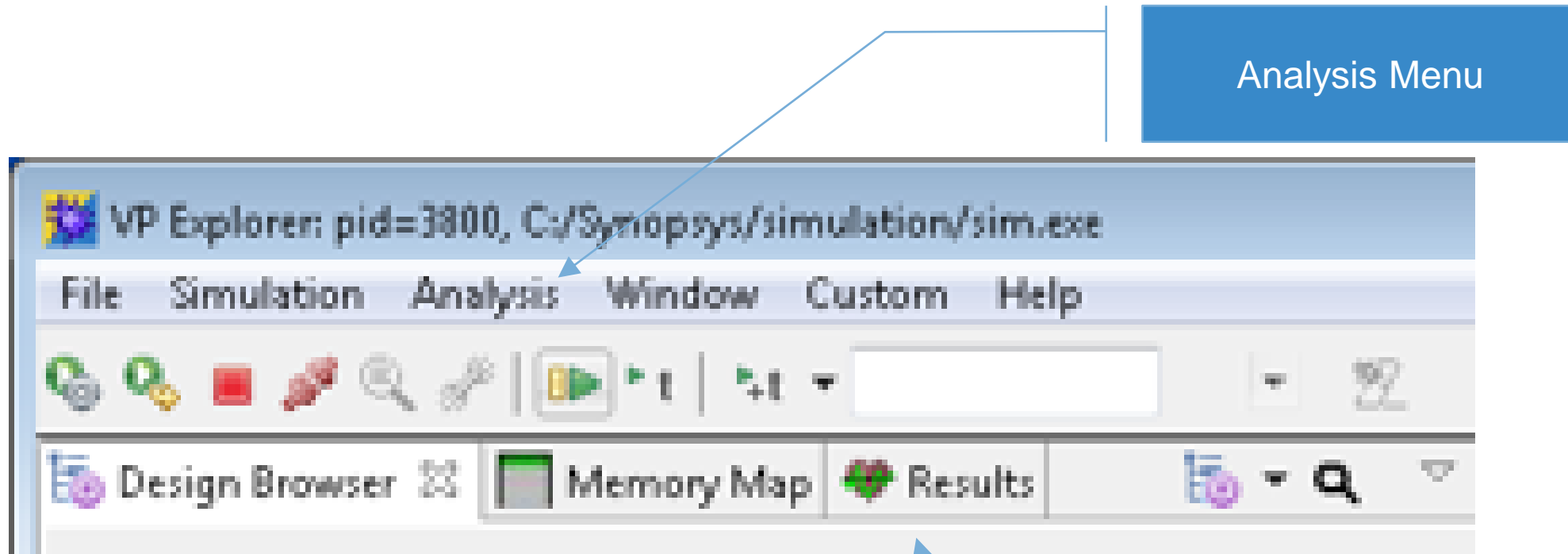
The screenshot displays the VP Explorer application window. The top menu bar includes File, Simulation, Analysis, Window, Custom, and Help. The main interface is divided into several panes:

- Design Browser:** Shows a design hierarchy with 'Current Simulation (suspended at)' expanded to 'Design Hierarchy' > 'SYSTEM' > 'ECU'.
- Breakpoints:** A table listing breakpoints with columns for State, Type, and Location. The first entry, 'initial\_crunch /', is highlighted in yellow and circled in green.
- Overview:** Shows 'VP Config: TestSW-A53' and 'VP Project: system'. There are 'Build' and 'Release' buttons.
- Console:** Displays the Tcl Console output, including 'INFO: Simulation initialization complete', which is circled in green.
- Bottom Status Bar:** Shows 'initial\_crunch /' and a timer '0:00:00.000 000 000 000', both circled in green.

# VP Explorer: Important Tool Buttons

Toolbar icons		Description
	Resume	Resumes suspended simulation.
	Pause	Pauses a running simulation.
	Stop	Stops a simulation. All cores and all peripherals are frozen. Simulation resets
	Restart	Restarts the simulation. In this virtual prototype skin, all images are also reloaded.

# Description and Location of Tool Bar Buttons



Exercise 1: Configure and Display Function Trace

# Set Up Function Trace

**Select ...cluster0.cpu0**

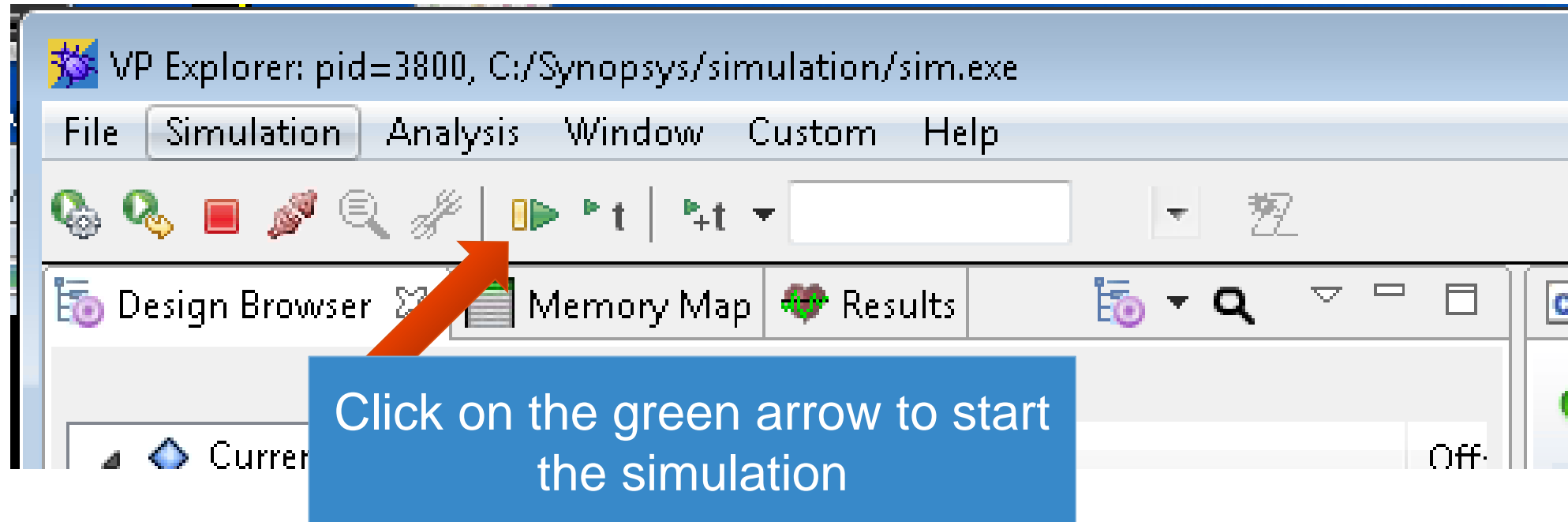
**Select Function Trace**

**Click Apply**

The screenshot shows the 'Configure Analysis' dialog box. The left pane shows a tree view under 'Software Analysis' with 'SYSTEM.ECU.S32V234.CPU\_BOARD.CPU.cluster0.cpu0' selected. The right pane shows a list of analysis options, with 'Function Trace' checked. The 'Apply' button is highlighted at the bottom.



# Run the Simulation



# Check Your System

The screenshot displays the VP Explorer simulation environment. The main window shows a 'Tcl Console' with the following output:

```
Frame Per Second set to: 29  
Data Type set to: 32  
LineStart/End packets turned on.  
Data transmission started  
Data transmission stopped  
Test Passedbreak
```

The text 'Test Passedbreak' is circled in green. Below the console, a timer displays '0:00:15.935 575 081 042' and a delta value of 'Δ 8480958', also circled in green. An inset window at the bottom shows a list of breakpoints, with the 'sw\_breakpoint' at '/SYSTEM/ECU/S32V234/CM4' highlighted in yellow.

State	Type	Location
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/LIN_MONITOR_1/msg_recv_t...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_0/msg_recv_trigger
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_1/msg_recv_trigger
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_2/msg_recv_trigger
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_1/msg_recv_trigger
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_3/msg_recv_trigger
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_1/msg_recv_trigger
<input checked="" type="checkbox"/>	sw_breakpoint	/SYSTEM/ECU/S32V234/CM4



# Next Steps: Hands-on

- **Exercise 2:** Configure the Results window
- Send results to Function Trace
- Filter function trace results for Programmable Interval Timer (PIT\_0) and PIT\_1 functions:
  - PIT\_0\_ISR
  - PIT\_1\_ISR
  - pit\_0\_interrupt\_test
  - pit\_1\_interrupt\_test
- Measure PIT\_0 and PIT\_1 elapsed time
  - PIT\_0 elapsed time = \_\_\_\_\_
  - PIT\_1 elapsed time = \_\_\_\_\_
  - Show how to reduce PIT\_1 timer from 12 seconds to 700 ms

# Check Function Trace

The image displays two screenshots of the VP Explorer software interface, illustrating the process of checking function traces.

**Top Screenshot:** Shows the main interface with a table of Design Object/Monitor Name, Monitor Type, and Data Item Name. The table is currently empty. Below the table is a Breakpoints window with a list of breakpoints. The bottom right shows a function trace chart with a time axis from 0 to 15 seconds. The chart displays several function traces, including PIT\_0\_ISR, PIT\_1\_ISR, pit\_0\_interrupt, and pit\_1\_interrupt. A cursor is positioned at 7223087886 ps.

**Bottom Screenshot:** Shows a detailed view of the function trace chart. The chart displays the same function traces as the top screenshot. A cursor is positioned at 7223087886 ps. The chart shows a significant delay in the execution of the pit\_0\_interrupt and pit\_1\_interrupt functions, which is highlighted by a red box.

State	Type	Location
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/LIN_MONITOR_1/msg_recv...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_0/msg_recv_trigge...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_1/msg_recv_trigge...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_2/msg_recv_trigge...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_1/msg_recv_trigge...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_3/msg_recv_trigge...
<input checked="" type="checkbox"/>	after_write_time	/SYSTEM/ECU/dspi_bus_1/msg_recv_trigge...
<input checked="" type="checkbox"/>	sw_breakpoint	/SYSTEM/ECU/S32V234/CM4

# HANDS-ON LAB: STARTING LAUTERBACH TRACE32 DEBUGGER

External debugger run-control



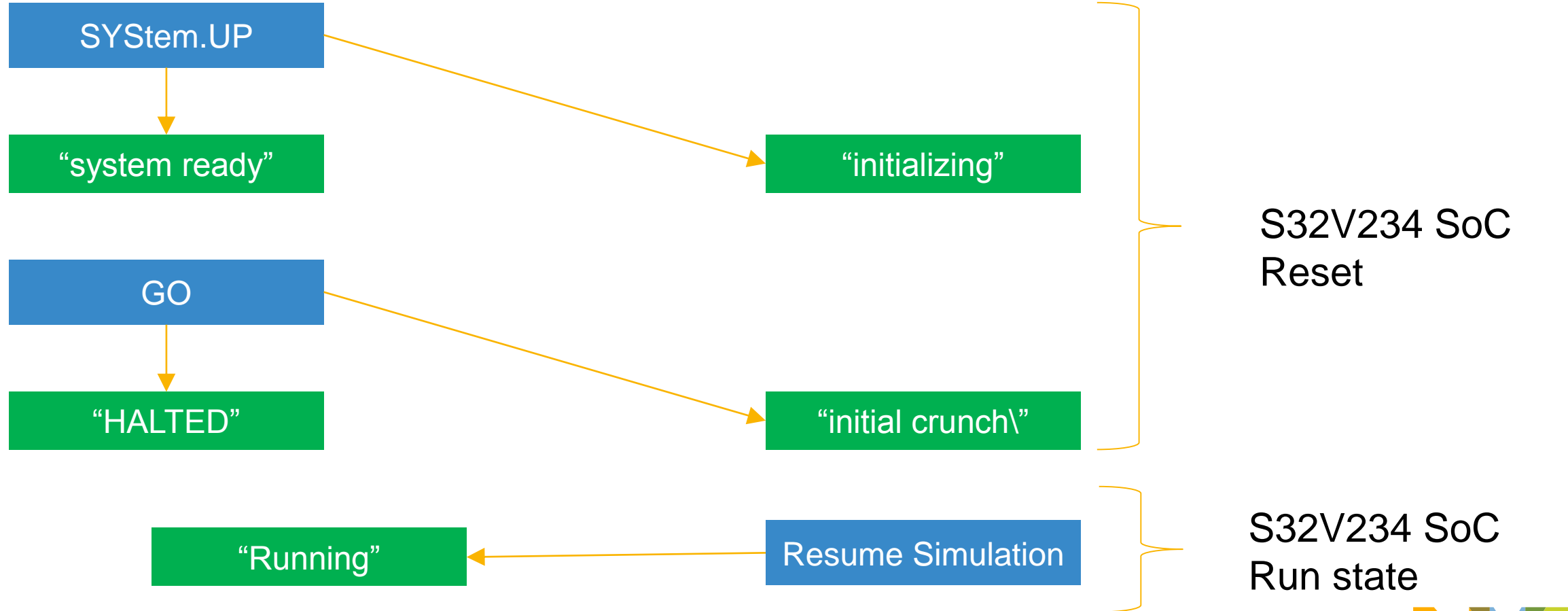
SECURE CONNECTIONS  
FOR A SMARTER WORLD

# Debugger Operation: Goals

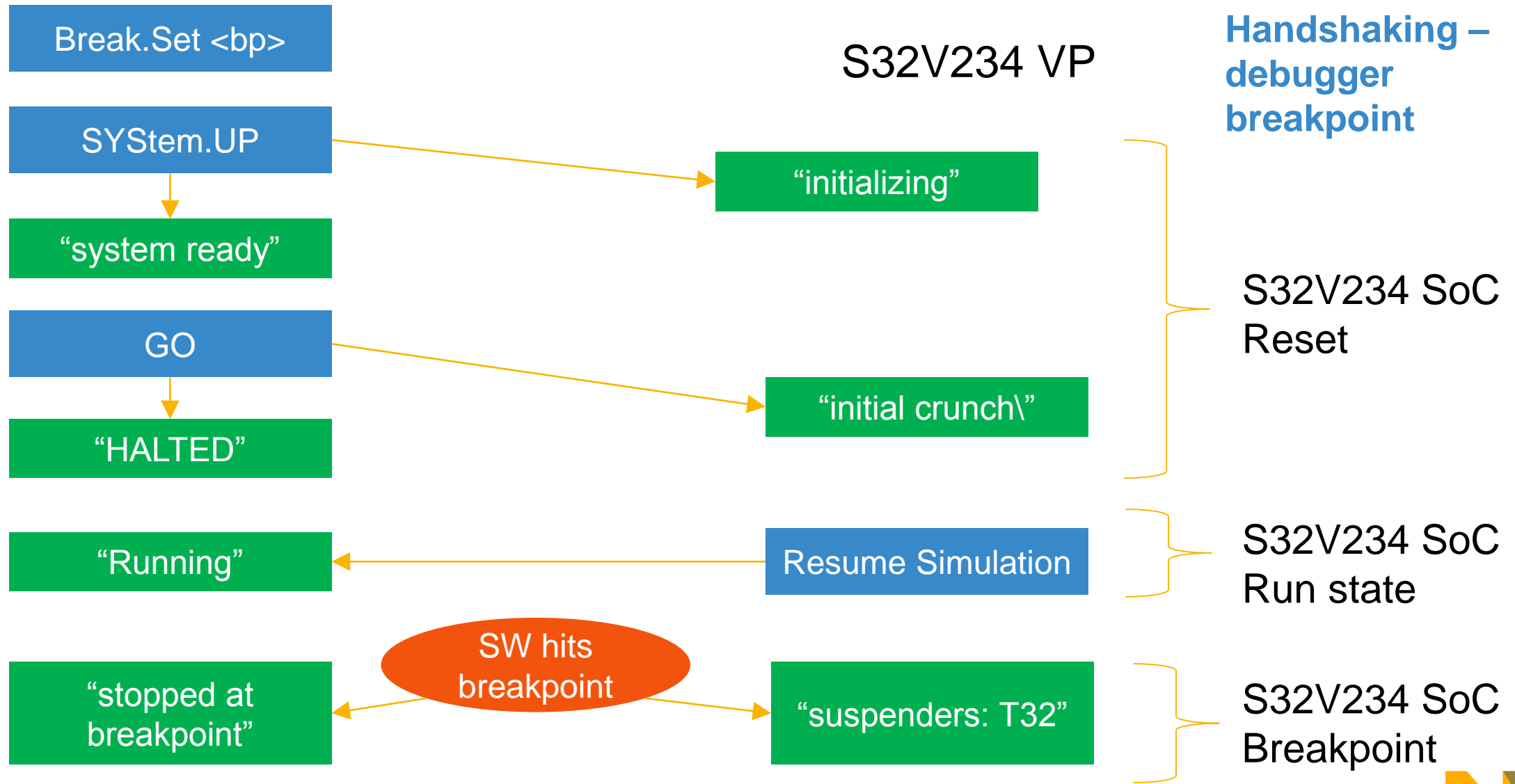
- Startup Lauterbach TRACE32 debugger
- Operate debugger and Virtual Prototype in tandem
- Set a breakpoint
- Run to breakpoint
- Modify S32V234 Vision Processor values from the debugger

# Handshaking

- TRACE32 Debugger



# TRACE32 Debugger





## Exercise 3:

- Break.set <function name> - run to the pit\_0\_timer and pit\_1\_timer interrupt routines, break at the start of the routines
- Inspect the code – how is it configuring the load value?

## Exercise 4

- Reduce test time from 12 seconds to 2 seconds
  - Option 1: Debugger run-time control – halt, modify, resume
  - Option 2: Debugger patch of code at start-time
  - Option 3: Use a precompiled \*.elf file with the change from 0x0FFF\_FFFF to 0x00FF\_FFFF. Rerun the test with the new elf file and confirm PIT\_1 elapsed time changes from 12 seconds to 700 ms

# Exercise XYZ: Create a New Skin & Run APEX-CV and/or ISP Demo

- Startup APEX-CV or ISP demo.
- Need Aneesh to port it over to VDK.
- Need Mojin to show steps for creating new VDK skin
  - Run sobel filter
  - Run test of all filters
- <TO DO: Add slides for creating a new skin>



# APEX-CV Example Code

```
/******  
* Main function  
*****/  
  
int main(int, char**)  
{  
  
//...  
  
test_apexcv_filter();  
test_apexcv_color_conversion();  
test_apexcv_arithmetic();  
test_apexcv_interpolation();  
test_apexcv_histogram();  
test_apexcv_integral_image();  
  
return 0;  
}
```



# APEX-CV Code Example

```
#include <apexcv_base_integral_image.h> //header for apexcv integral image

using namespace apexcv; //optional if not using apexcv::

int test_integral_image(const unsigned char *src, int w, int h)
{
    int lRetVal = 0;
    apexcv::IntegralImage i; //create class object

    DataDescriptor srcImg(w, h, DATATYPE_08U); //create src buffer
    memcpy(srcImg.GetDataPtr(), src, w*h); //copy external content into src buffer
    DataDescriptor dstImg(w, h, DATATYPE_32U); //create dst buffer

    if ( srcImg.IsOK() && dstImg.IsOK() ) {
        lRetVal |= i.exec(srcImg, dstImg); //run implementation
    } else {
        lRetVal |= 1;
    }

    //TO SOMETHING WITH dstImg

    srcImg.SetFreeOnExit(true); //free buffers on exit
    dstImg.SetFreeOnExit(true); //free buffers on exit

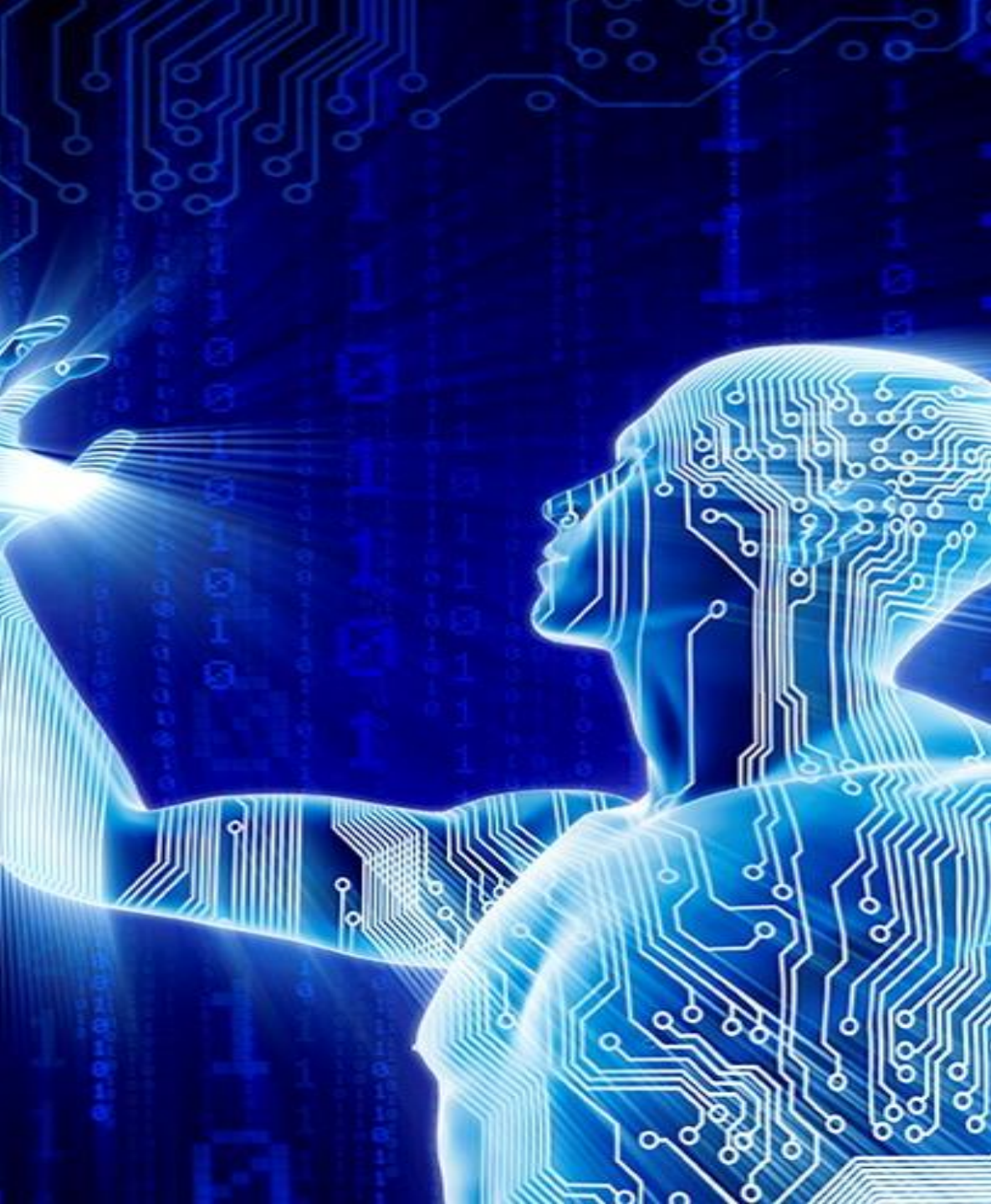
    return lRetVal;
}
```

# CONCLUSION

# Conclusion

- In today's class, you learned:
  - Definition and benefits of a Virtual Prototype
  - How to start the Synopsys VDK and configure / view function trace
  - How to run VDK demos
    - Linux boot
    - A53
    - ISP/APEX
  - How to modify code and re-test
  - How to create a new VDK skin for an existing Vision SDK demo





## Conclusion

- The S32V234 Virtual Development Kit speeds your time to market for vision processing solutions:
  - 1) Developers can start early, develop full chip software
  - 2) In tandem, programmers can develop comprehensive regression test suites
  - 3) This results in very rapid porting of high-quality code once hardware evaluation systems become available





**NXP: FUELING  
AUTOMOTIVE  
INNOVATION**



SECURE CONNECTIONS  
FOR A SMARTER WORLD



SECURE CONNECTIONS  
FOR A SMARTER WORLD

## ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE Flex, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and  $\mu$ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. © 2015–2016 NXP B.V.

