2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Estimating Power Dissipation of End-User Application on RTL

Magdy El-Moursy, Ph.D.
Senior Engineering Manager

SIEMENS

accellera SYSTEMS INITIATIVE

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

Changing the platform

Summary

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

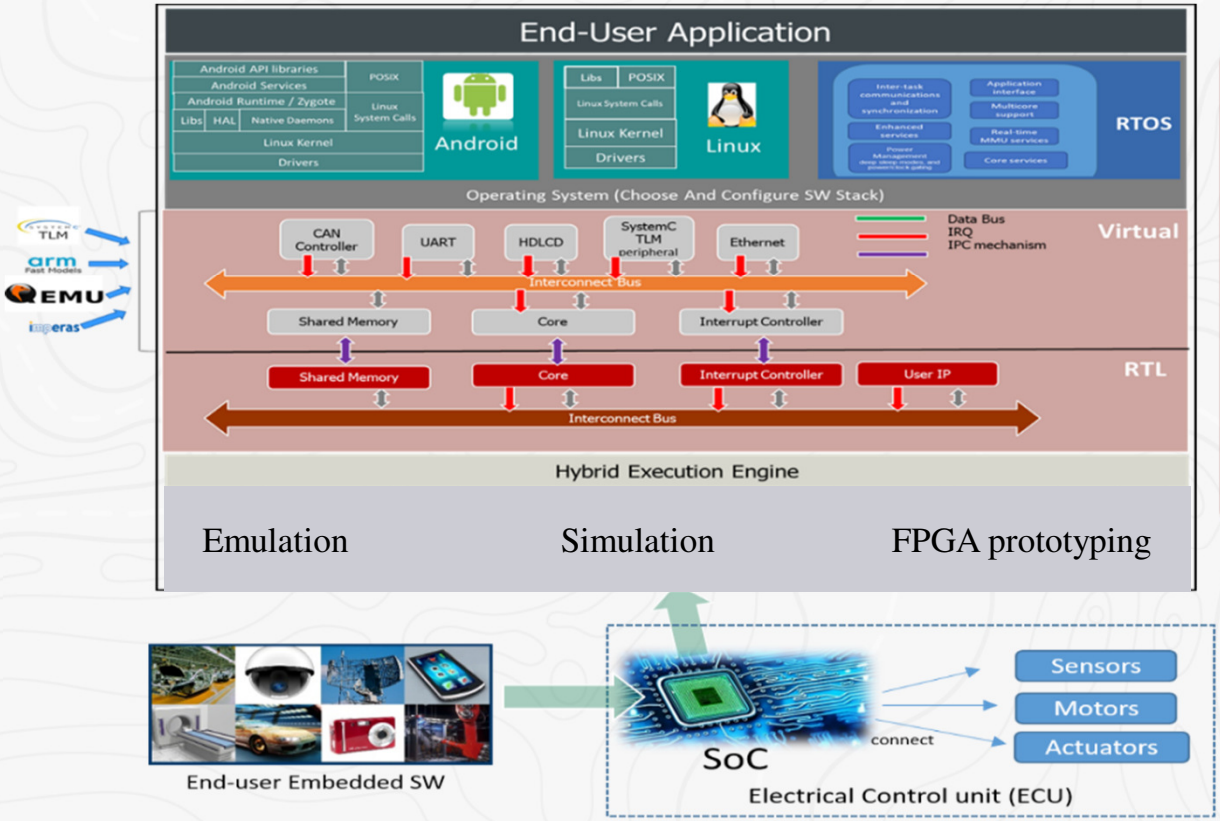
Viewing the results

Changing the platform

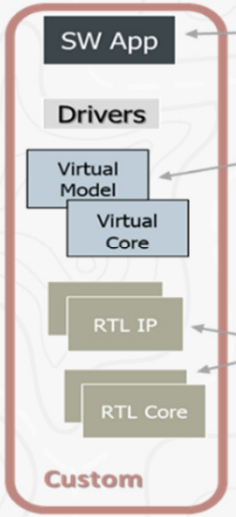
Summary

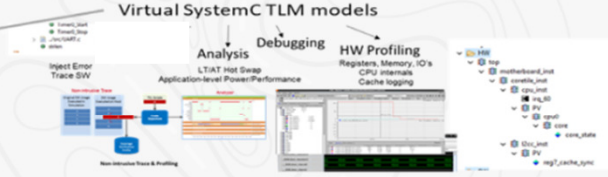
The following slides are an introduction to the tools referenced in the workshop

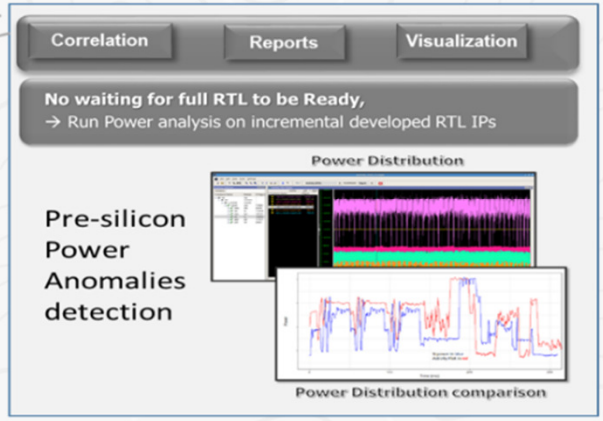
Software/Hardware Co-Development Methodology



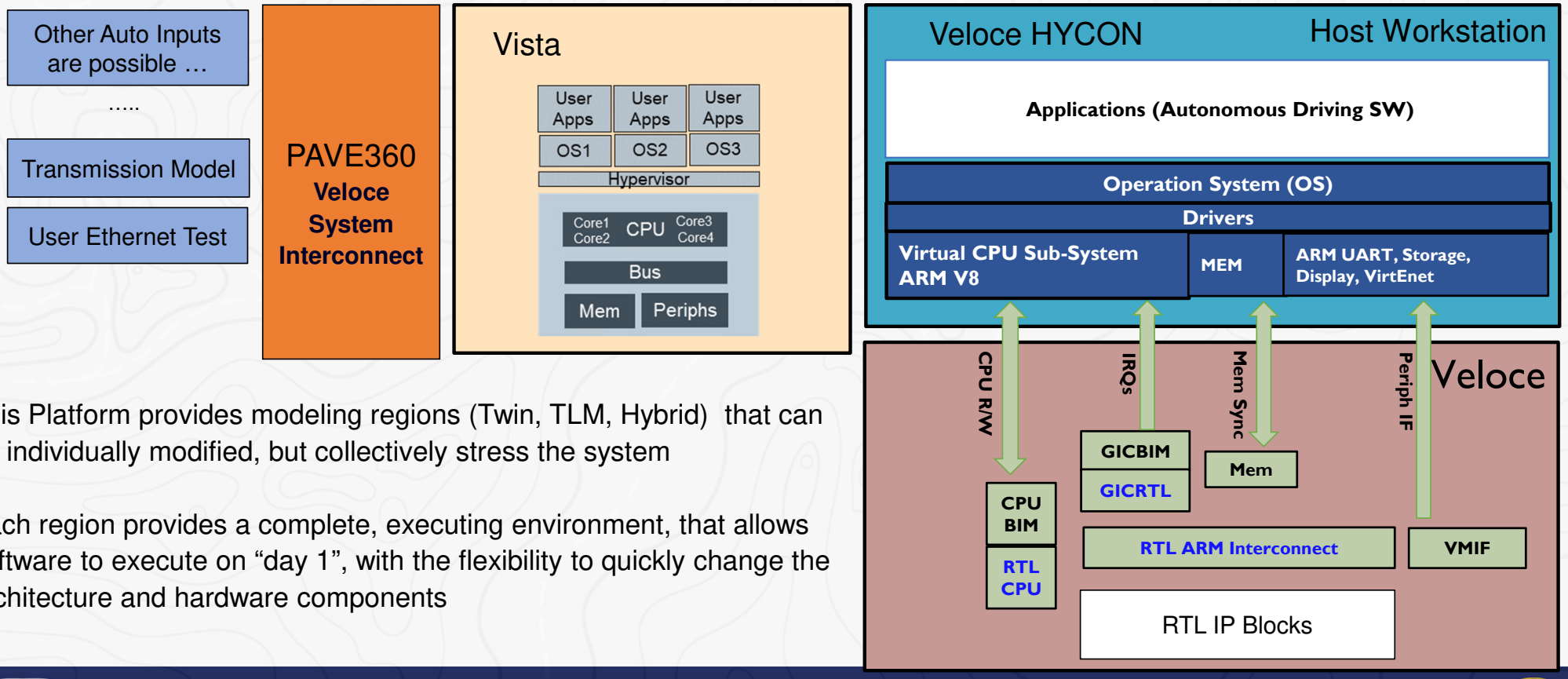
End-user Embedded SW







Hybrid Reference Platform Allows Flexibility

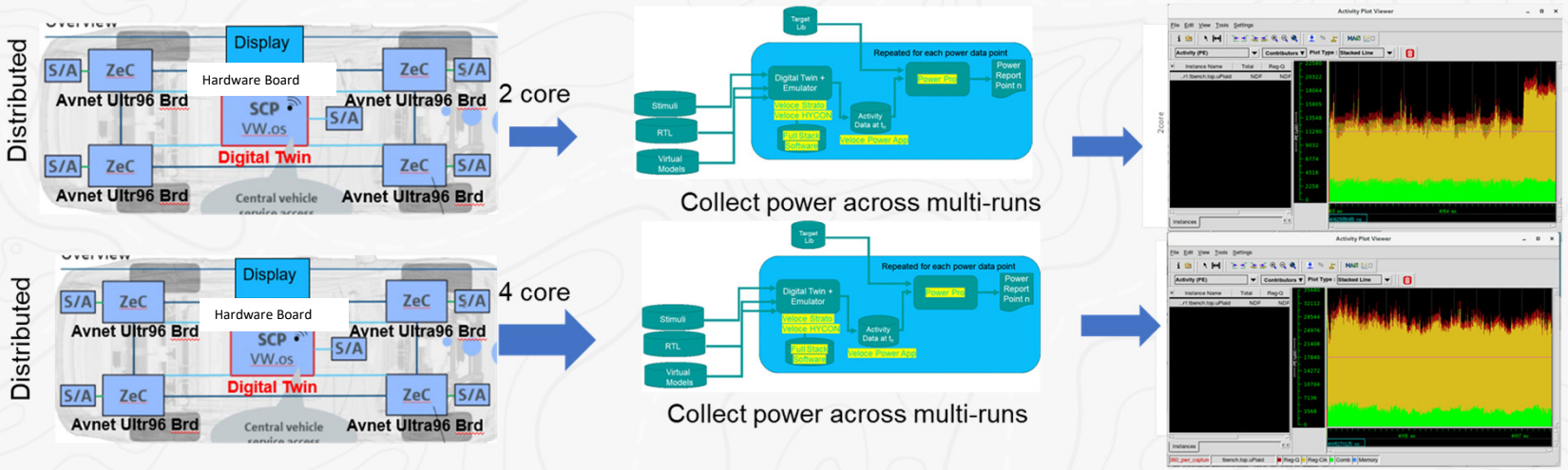


This Platform provides modeling regions (Twin, TLM, Hybrid) that can be individually modified, but collectively stress the system

Each region provides a complete, executing environment, that allows software to execute on “day 1”, with the flexibility to quickly change the architecture and hardware components

Scalable Verification Power Use-Case

- Identify power consumption of end-user applications, across 10's of seconds, pre-silicon
  - Execute end-user app; capture/calibrate/analyze power profiles; identify "areas of interest"



  - End-User knowledge needed: how to run apps; how to extract data from OS's and drivers; etc.
  - Multiple methods needed: Execution of Apps; identification of "areas of interest"; sampling; calibration
  - Multiple products needed: Veloce HYCON (SW), Veloce Strato+ (HW), Veloce Power App (data), PowerPro (get power)

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

Changing the platform

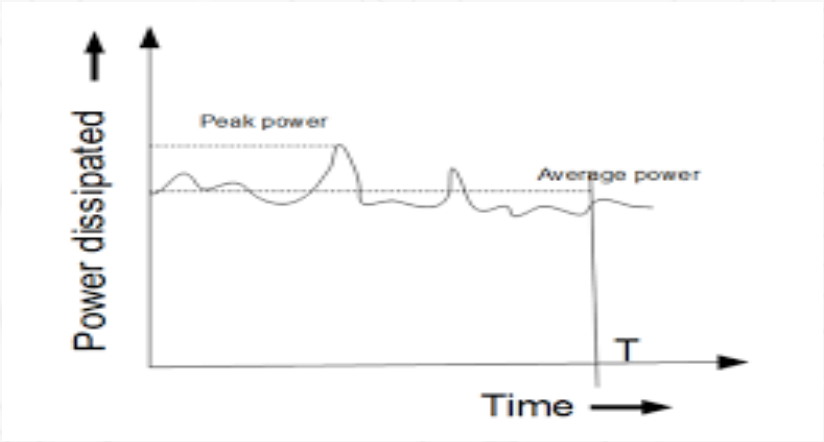
Summary

Power Dissipation of SoC

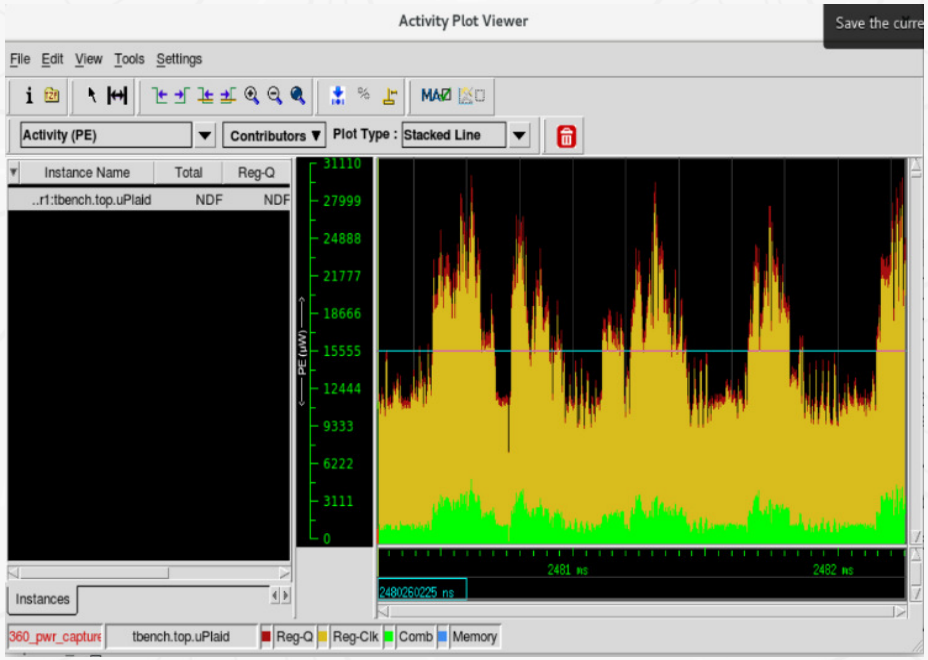
- **Power dissipation could be determined by multiplying the current running from the power supply source and the supply voltage**
  - Supply voltage is highly stable and usually assumed to be constant
  - Instantaneous power is usually not of an interest
  - Average power is used to indicate the cost of running an application on the HW
  - Current is determined over short periods of time to determine the average power over those periods
  - For digital circuits, a power equation could represent the components to determine the power dissipation

$$P_{dyn} = \alpha C_{load} V_{dd}^2 f$$

  - All components of this equation are HW oriented except the activity factor which completely depends on the running application
  - We need to know information about the HW (RTL and beyond) and the running application to get the power

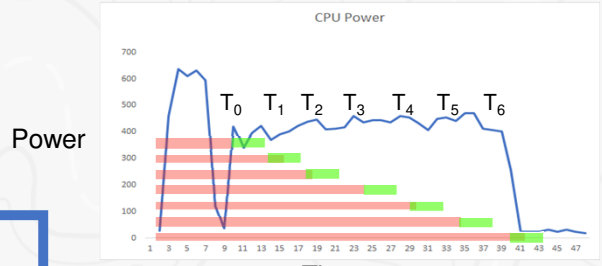


Software to Systems – How do we do it?



Power Flow Methodology

Run-Fast to time t, then Run-Accurate to Calculate Point n



TSMC Libs

Target Lib

Sensitivity Analysis Runs

Repeat for each power data point

Digital Twin + Emulator

Power Pro

Power Report Point n

Stimuli

RTL

Virtual Models

Veloce Strato+, Veloce HYCON

Full Stack Software

Activity Data at t_n

Veloce Power App

Sampling Power Report for Point N

```
Power Summary Report
-------------------------------------------------------------------------------------------------------------------
Power Group     Count       Leakage Power(uW)     Internal Power(uW)     Switching Power(uW)     Total Power(uW)     Percentage(%)
-------------------------------------------------------------------------------------------------------------------
io_pad          0           0                     0                      0                       0                   0%
memory          0           0                     0                      0                       0                   0%
black_box       15436       0                     0                      59.8304                 59.8304             0.11%
register        1163476     3153.01               22954.8                1347.59                 27455.4             49.72%
combinational   9830756     4800.58               4655.89                5029.58                 14486               26.23%
sequential      0           0                     0                      0                       0                   0%
clock_network   142709      510.792               6969.5                 5742.36                 13222.6             23.94%
total           11152377    8464.38               34580.2                12179.4                 55223.9             100%

    [UI-STATS]   PowerPro process is using 28873 MB, peak 31904 MB, and 14296 seconds, 3:58:16 [+0MB, +0MB, +0s] @ Tue Sep 14 04:11:44 2021
```

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

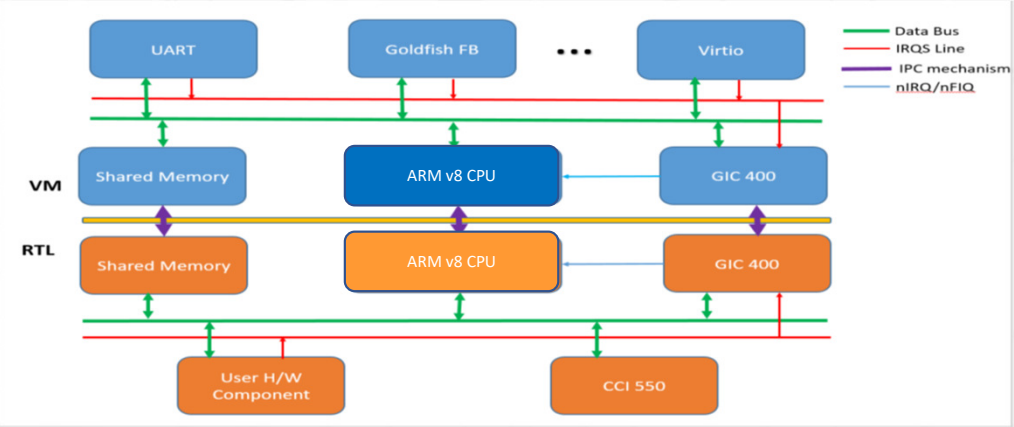
Viewing the results

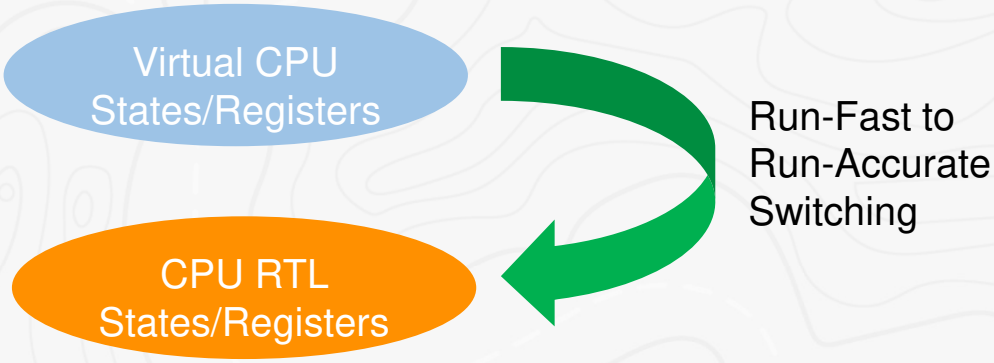
Changing the platform

Summary

Run-Fast Run-Accurate Concept

- **Hybrid solution introduces the concept of being able to run the end-user application in two modes - Run-Fast mode or Run-Accurate mode**
  - 1msec of emulation (with HW acceleration) takes 30min of execution [**30sec would take around 2 years**]
  - Switching between the two modes is done at run time
  - The platform needs to be configured to allow the switching at run-time
  - With Run-Fast (RF) mode, the main processing element of the SoC, the CPU, is virtually modeled at the instruction level
  - Switching from RF to RA takes less than a minute of execution





Full SW Stack with RFRA

1. Run with Virtual CPU in Veloce Hycon RF
2. Use **Veloce Hycon RFRA** to Inject State in RTL

3. Use RTL CPU to extract power data using **Veloce Power App**
4. Use **PowerPro** to extract power

Table B2-1 Registers with IMPLEMENTATION DEFINED bit fields

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
ACTLR_EL1	3	c1	0	c0	1	64	B2.5 ACTLR_EL1, Auxiliary Control Register, EL1 on page B2-146
ACTLR_EL2	3	c1	4	c0	1	64	B2.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page B2-147
ACTLR_EL3	3	c1	6	c0	1	64	B2.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page B2-149

Veloce HYCON

AUTOMOTIVE Applications

Object Detection SW Ext.

ANDROID / LINUX

CAN Socket

Linux Network Stack

Linux Ethernet Driver

Camera Stereo Images

Read JPG Images

Boot Code

RCAR CAN

Display

Ethernet Controller

Virtual A76AE

Virtual HW Platform

Virtual GPU

Memory

UART

Hybrid Execution Engine

RTL A76AE

RTL Emulation Engine

Bus

Full Stack SW

- Boot Code
- Linux
- Android
- Auto Apps

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

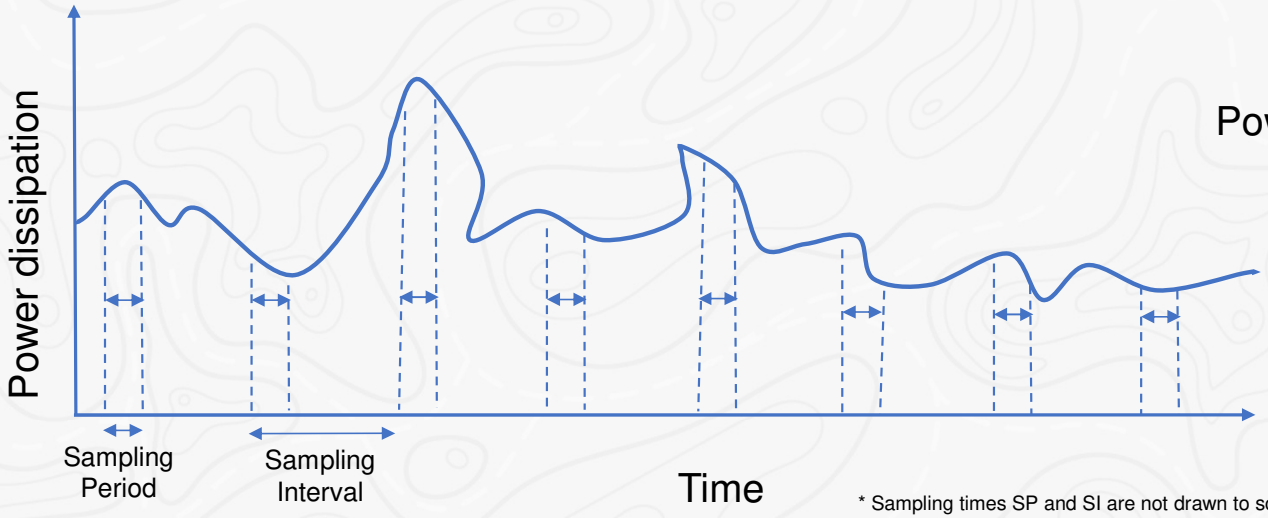
Execute the platform for performance

Viewing the results

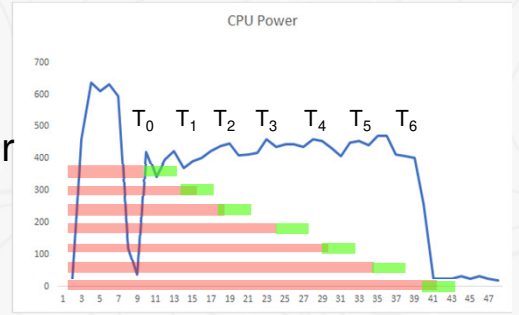
Changing the platform

Summary

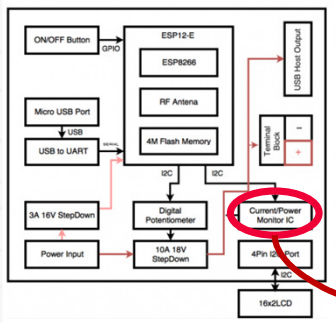
Power Dissipation Sampling Mechanism on Real HW

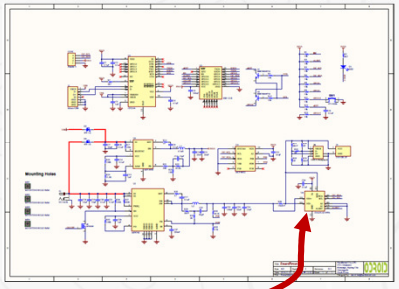


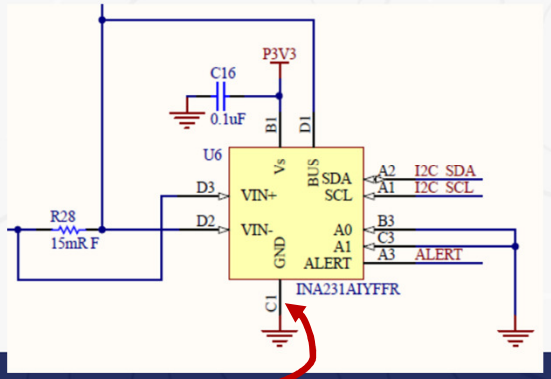
Power

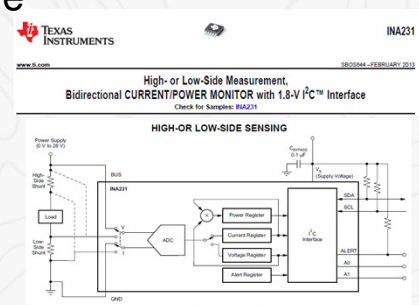


* Sampling times SP and SI are not drawn to scale for demonstration
- By default SP is 1.1msec and SI is 1 sec









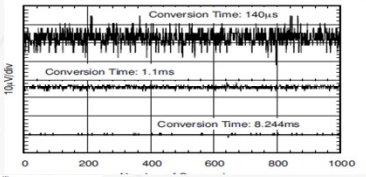


Table 4. V_{BUS} CT Bit Settings [8:6]

V_{BUS} CT2 (D8)	V_{BUS} CT1 (D7)	V_{BUS} CT0 (D6)	CONVERSION TIME
0	0	0	140 µs
0	0	1	204 µs
0	1	0	332 µs
0	1	1	588 µs
1	0	0	1.1 ms
1	0	1	2.116 ms
1	1	0	4.156 ms
1	1	1	8.244 ms

Power Activity Capture Ratio

- **Veloce provides flexibility in capturing the power profile for different capture ratios**
  - HW board measurement determines the average power during the sampling periods
  - For accurate results it is recommended to use CR-1 (Capture Ratio-1, not skipping any cycle)
  - Determining the power outside of the Sampling Period is useless and would not match real HW
  - We created scripts to determine the power only during the Sampling Periods
  - We use the unique RF-RA mechanism of Veloce HYCON
  - We run the platform in RF mode to the start of each Sampling Interval (SI)
  - We switch to RA mode and Emulate for the Sampling Period (SP)
  - The generated activity is then imported to PowerPro to determine the average power dissipation
  - The average power is plotted and compared to the power of the real HW board

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

Changing the platform

Summary

Automating the Execution Flow

- **The flow is fully automated to get the average power numbers**
  - A shell script to run the platform and a Veloce script (*run.do*) to execute the following:
    - Boot OS on the target
    - Mount the target file system to the host file system
    - Start the application on target
    - Run the emulation in RF mode till the needed Sampling Interval
    - Switch to accurate mode RA
    - Enable power tracing of the emulation with CR-1 to generate activity database
    - Run the emulation for the Sampling Period
    - Close the emulation and quit

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

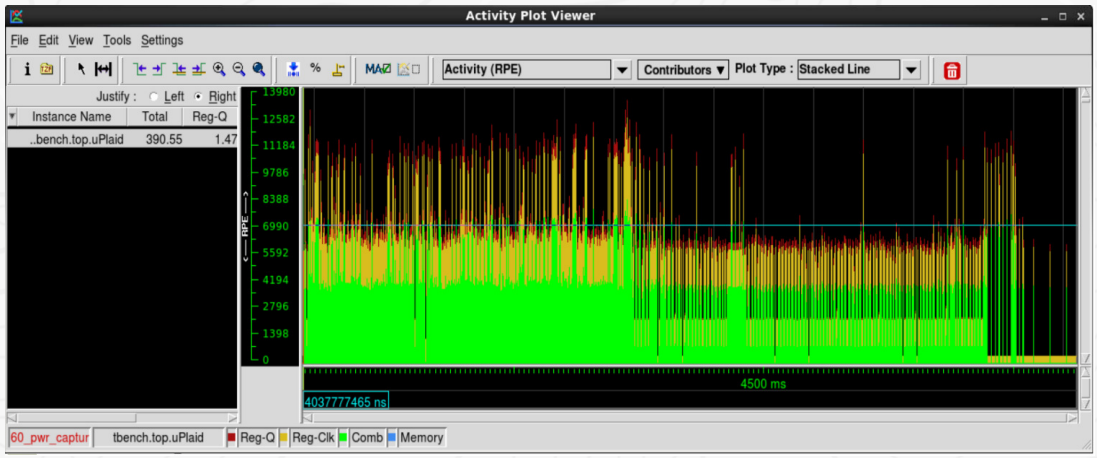
Viewing the results

Changing the platform

Summary

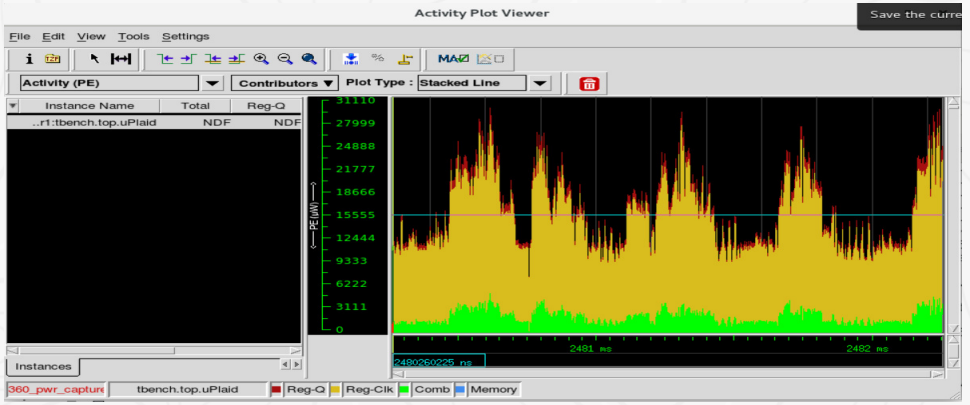
Veloce Power Profile

- **This flow generates a power profile for the design**
  - Generating this database runs in parallel to the emulation run

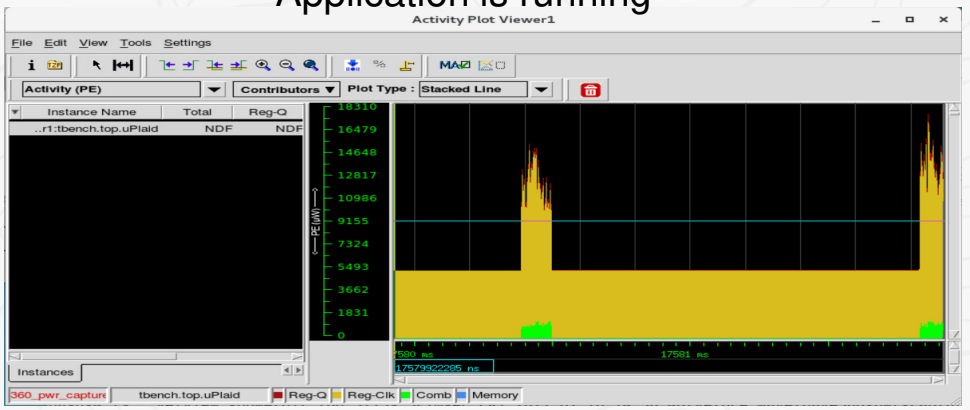


Power Profile of an Automotive Application

- **We use the power profile to monitor the activity of the running application on the platform**
  - For this example:
    - We run in RF mode till the point at which we need to get the average power or check the activity
    - We switch to RA mode
    - We set the capture ratio to 1 and enable tracing
    - We run it for short period 1msec (Sampling Period) in RA
    - We capture power profile
  - In RF, the application takes 25sec of Simulation, 30min of Execution
  - Here we capture in the middle of the application run and after it finishes (we can see the difference in the activity)



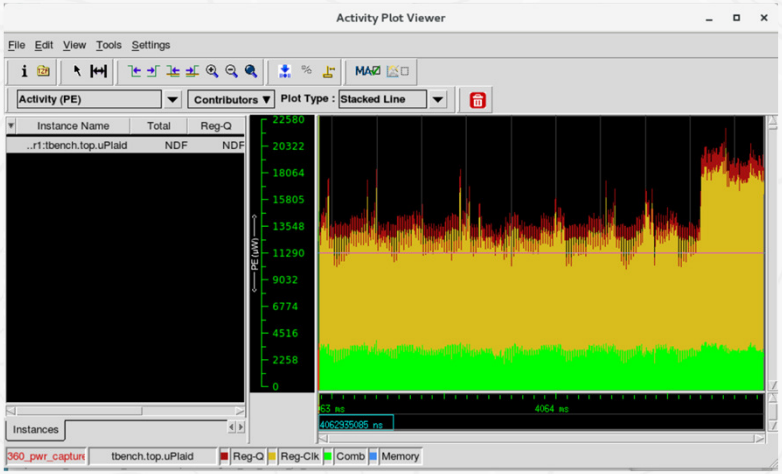
Application is running



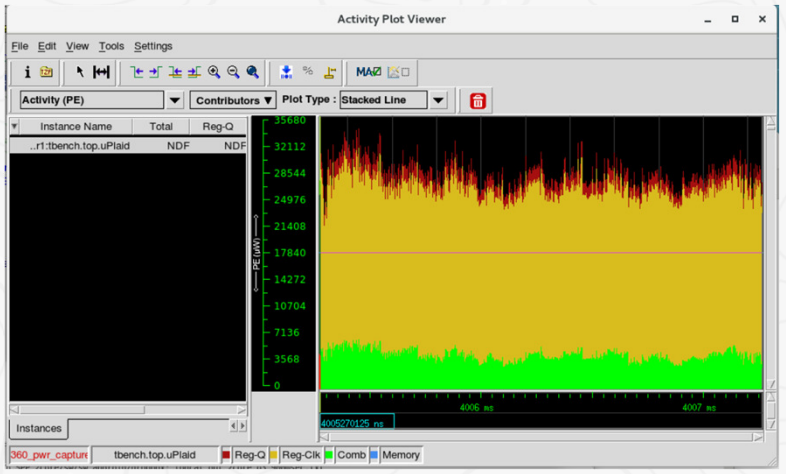
Application is finished

Automotive Application using Different HW Designs

- **We can quickly re-configure the platform and see the effect on the power profile**
  - We run 2core and 4core designs.
  - We notice 50% increase in the peak activity with 4core as compared to the 2core.



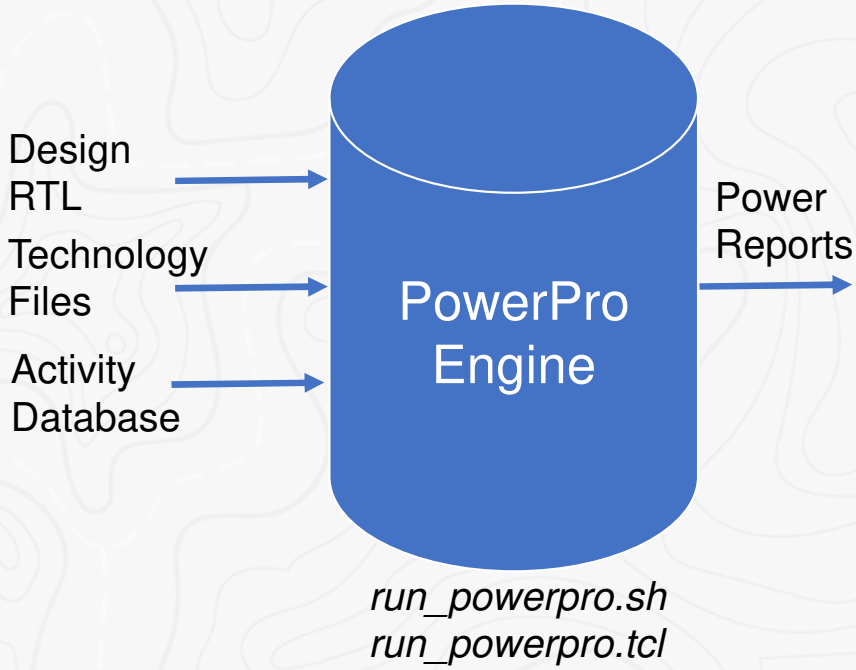
2core design



4core design

PowerPro Flow

- **Two scripts are provided to run the flow of the power calculation tool *PowerPro***
  - The shell script *run_powerpro.sh*
    - Sets the needed environment variables and license.
    - Executes the tcl script *run_powerpro.tcl*
  - The shell script *run_powerpro.tcl*
    - Sets the needed global variables for the flow
    - Reads the technology library (to determine node capacitance and supply voltage)
    - Reads the design RTL and builds design internal database
    - Reads the activity database which is generated by Veloce in the previous steps
    - Injects the activity database for all nodes and calculates the power of all nodes
    - Reports the average power in output text file which can be plotted



Sample of PowerPro Flow Output

Power Group	Description
io_pad	Specifies power consumption for all input-output pads in the design.
memory	Specifies power consumption for all identified memory objects.
black_box	Specifies power consumption for all black box objects.
register	Specifies power consumption for all flop and data latch objects.
combinational	Specifies power consumption for all combinational gate objects which are not part of clock network.
sequential	Specifies power consumption for all CGIC and clock network latch objects (only if global pa_clock_network_include_cgics is set as 0).
clock_network	Specifies power consumption for all objects which are part of clock network. CGIC and latch power is excluded if global pa_clock_network_include_cgics is set as 0.
total	Specifies the total power of the design.

Power Summary Report

Power Group	Count	Leakage Power(uW)	Internal Power(uW)	Switching Power(uW)	Total Power(uW)	Percentage(%)
io_pad	0	0	0	0	0	0%
memory	0	0	0	0	0	0%
black_box	15400	0	0	2466.85	2466.85	0.08%
register	1132068	837.503	327492	28593	356922	11.5%
combinational	8891559	20255.7	1.30165e+06	1.17945e+06	2.50136e+06	80.63%
sequential	0	0	0	0	0	0%
clock_network	138886	148.001	143868	97693.5	241710	7.79%
total	10177913	21241.2	1.77301e+06	1.3082e+06	3.10246e+06	100%

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

Changing the platform

Summary

Board Measurements Setup

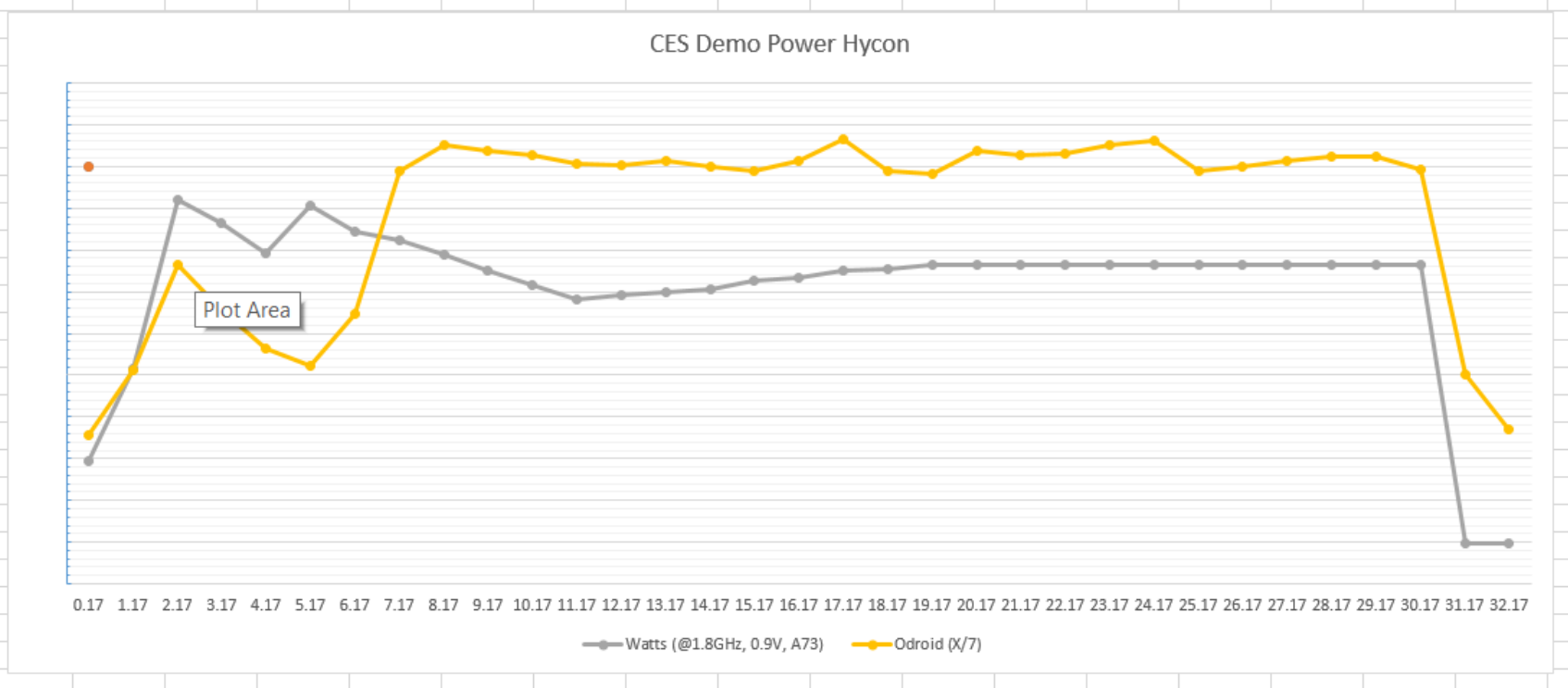
- **For power calculations, current flow on the power supply grid of the whole Odroid board is measured with Smart Power2 device**
  - Current is measured for a Sampling Period (SP) every Sampling Interval (SI) on the power grid
  - Default SP of 1.1msec and SI of 1sec are used to get the power of the board
  - Odroid-N2 SBC board contains 4-core ARM v8 based Amlogic S922X SoC
  - OS is running on the board for around 1 minute
  - ADAS (Advanced Driver-Assistance Systems) application with PAVE-360 object detection and AI object classification are running on the board to measure the power dissipation
  - ADAS (Advanced Driver-Assistance Systems) application runs for around 30 seconds (within the 1min)
  - Maximum power defined by the board specifications for all components under stressful conditions is determined to be around 11.5W
  - Maximum power defined by the board specifications for 4-core ARM v8 CPU under stressful conditions is determined to be around 5.5W (which represents around 50% of the board power)

Veloce HYCON Setup

- The following is used/assumed to get the power graphs for the Digital Twin:
  - RF mode is used to boot the OS [in 2000msec RF Simulation Time (SimT)]
  - ADAS application for PAVE-360 object detection and AI object classification is running on Veloce HYCON platform
  - ADAS application is left to run for multiples of 300msec RF SimT then the emulation is switched to RA mode
  - Power is determined for a SP of 2msec in RA mode
  - Activity Capture Ratio (CR) is set to 1 to be able to do power estimates

Comparing Emulation Results with HW

- **Here is the power of the Odroid HW board versus the Emulated values scaled based on circuit size**



Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

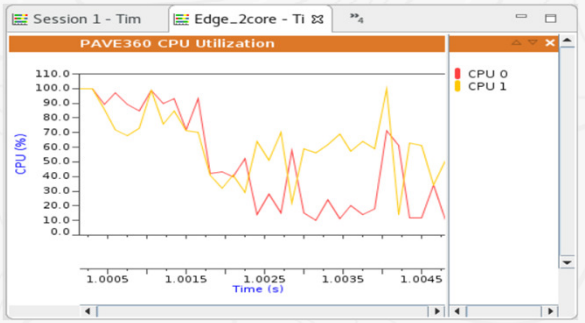
Changing the platform

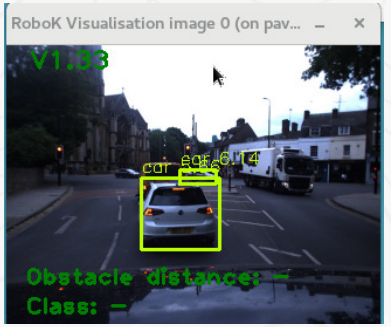
Summary

Areas to Explore

Areas to Explore during the Workshop

- ❑ CPU Utilization for: 2-core, 4-core, 8-core
- ❑ Execution of the ADAS software
- ❑ Change execution time
- ❑ Change number of cores





Running the Platform with CPU Utilization Flow

- **We created a script to run the whole flow on the platform. It does the following:**
  - Runs the Emulation (using Veloce Strato+)
  - Boots the OS on the platform
  - Runs the application
  - Dumps the CPU utilization raw information every 1 sec of execution
  - Converts the raw data to a format that can be displayed using Siemens Sourcery Analyzer tool

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

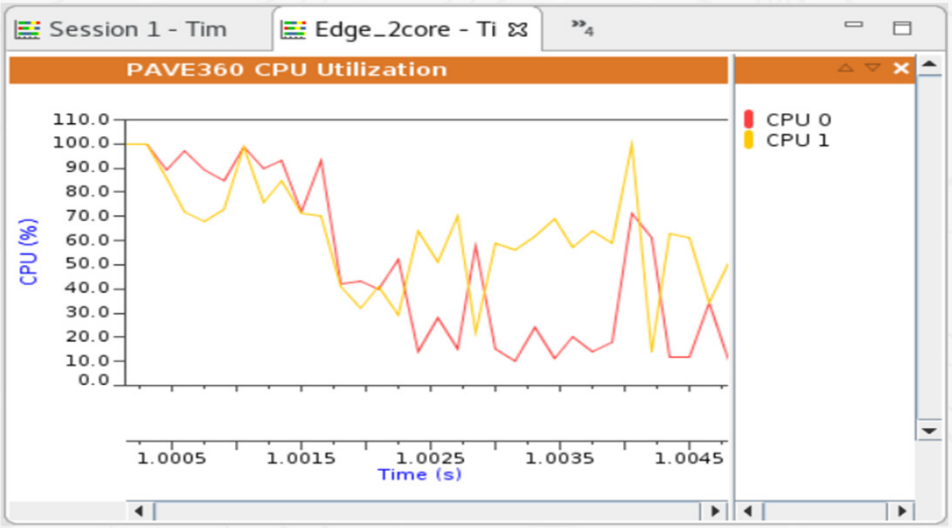
Execute the platform for performance

Viewing the results

Changing the platform

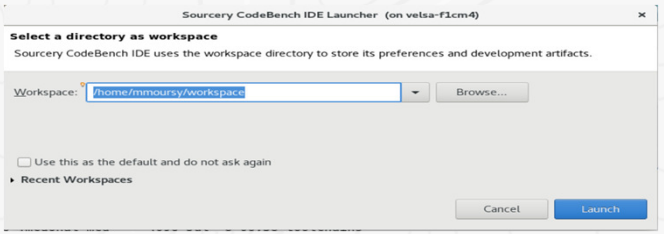
Summary

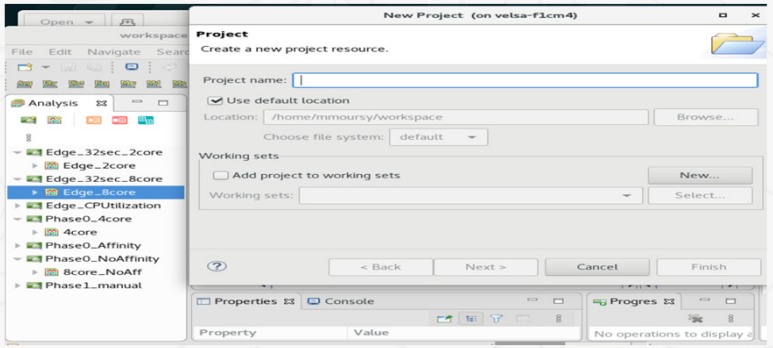
CPU utilization, during the application execution, will be visualized to allow analysis.

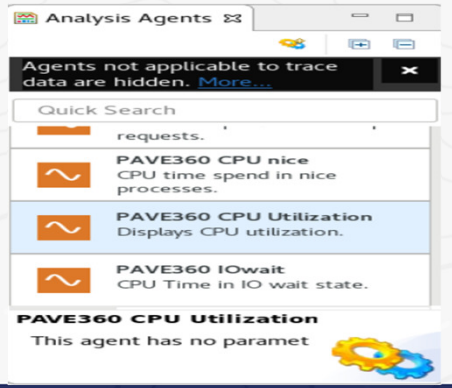


Steps to Visualize Data [running SA]

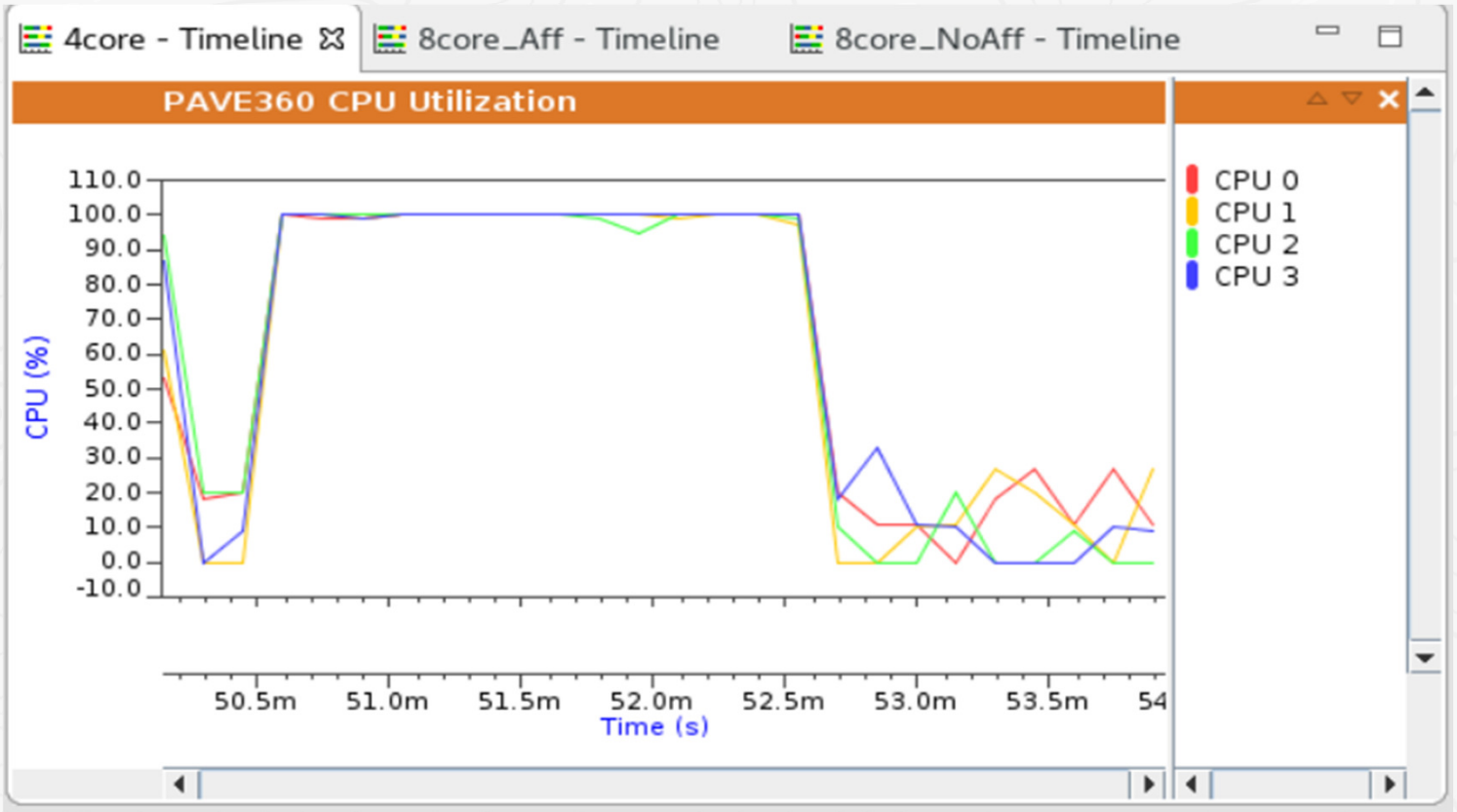
- **Sourcery Analyzer (SA) is a plugin for Codebench Siemens tool**
  - Launch SA using
  - Once SA is up and running, create new Analysis project
  - Import the database to SA
  - Click on the created Session → Agents
  - Double click on the “PAVE360 CPU Utilization” under the “Analysis Agents” Tab
  - Note, you can access pre-captured results





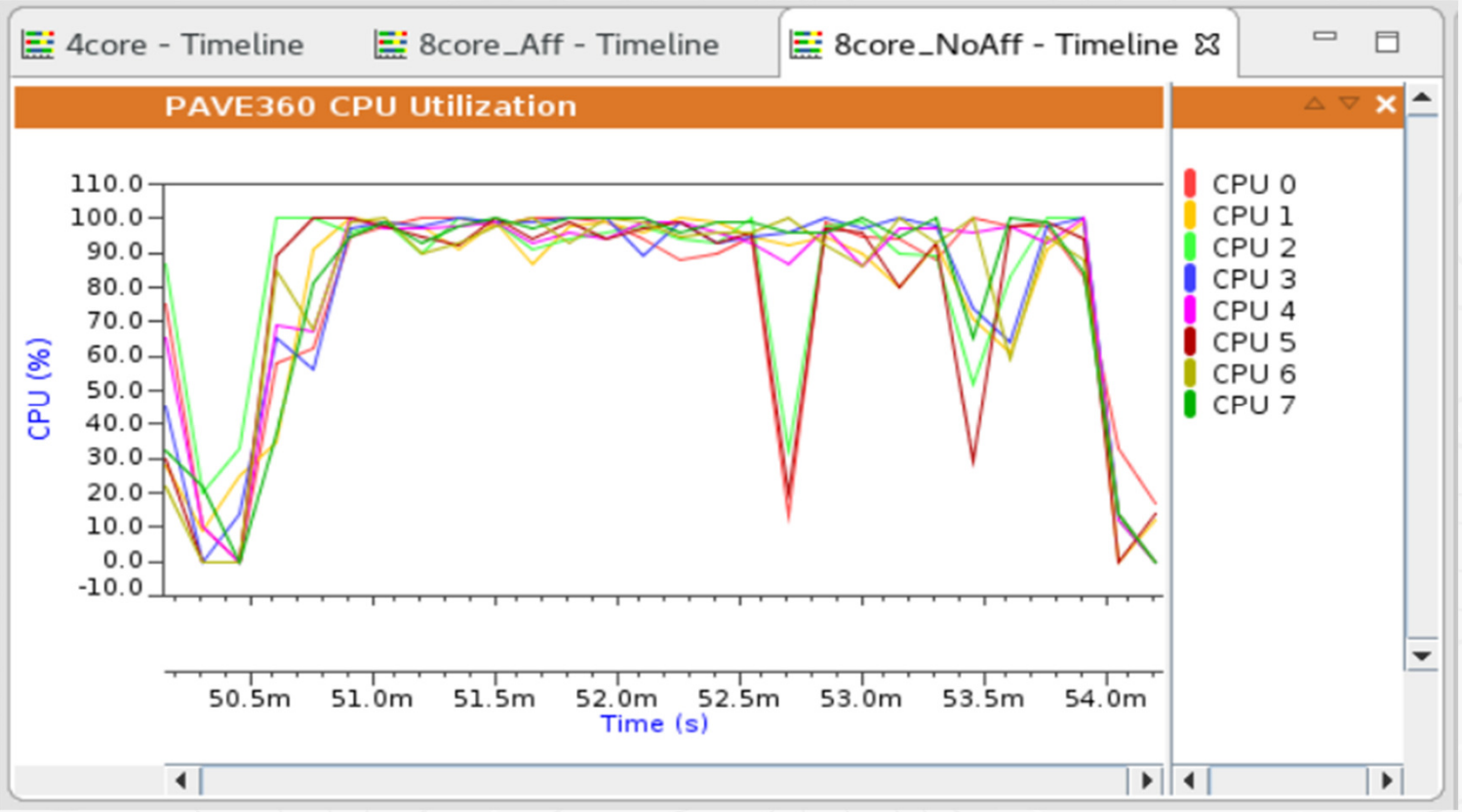


CPU Utilization with 4-Core Hardware Design (App1)



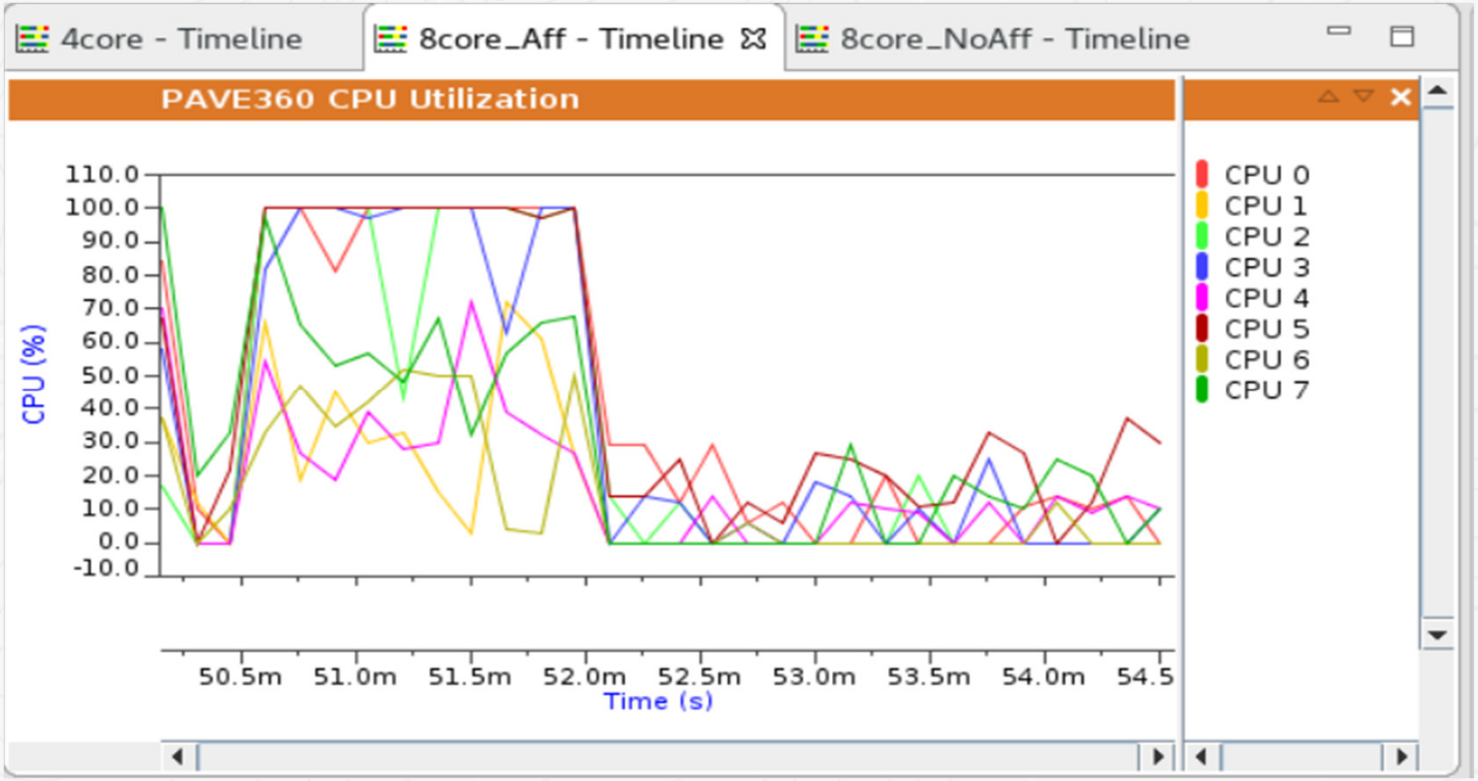
HW 4-core

CPU Utilization with 8-Core Hardware Design (App1)



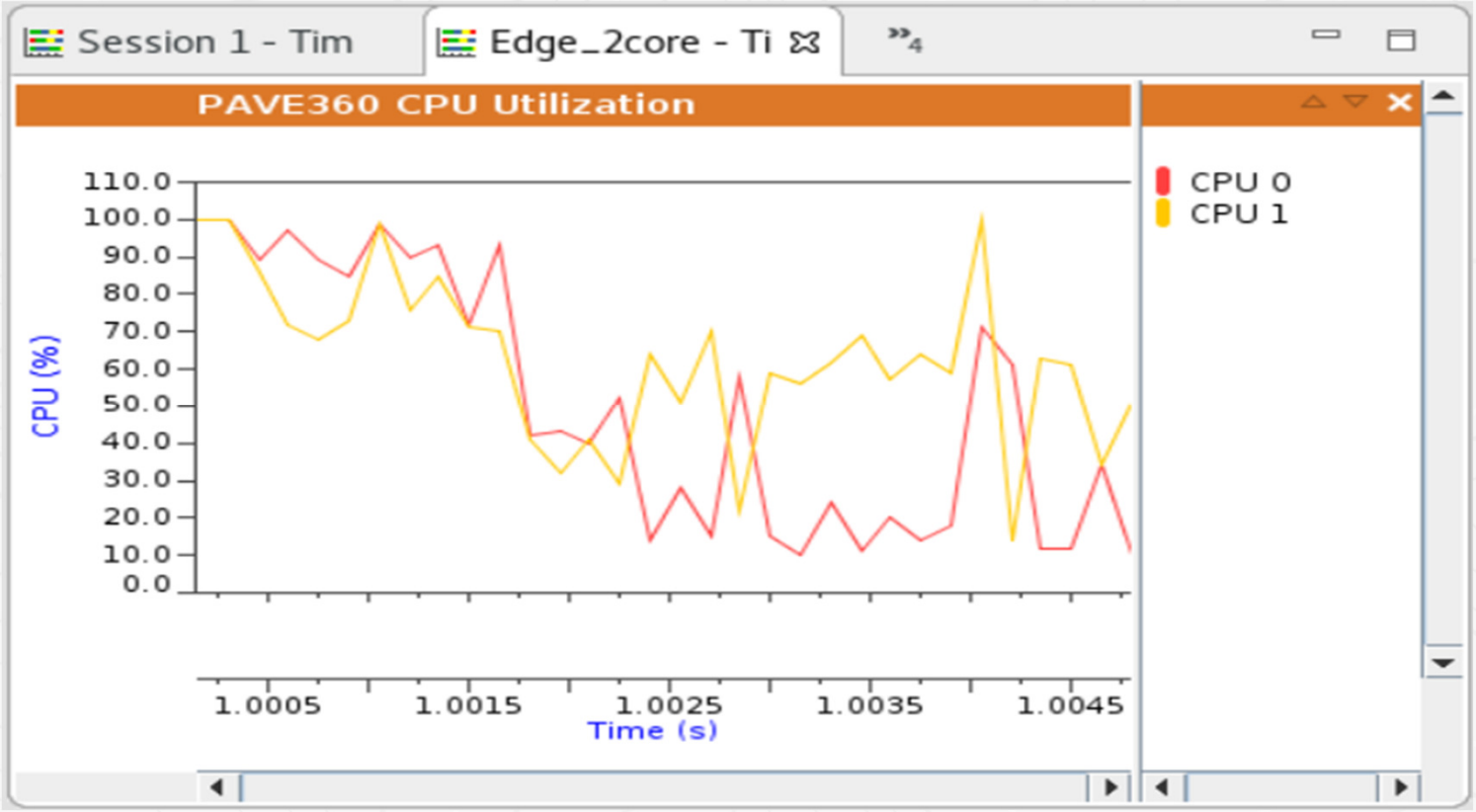
HW 8-core with Affinity Disabled (open SW to 8-core)

CPU Utilization with 8-Core Hardware Design (utilizing only 4 cores in SW, App1)



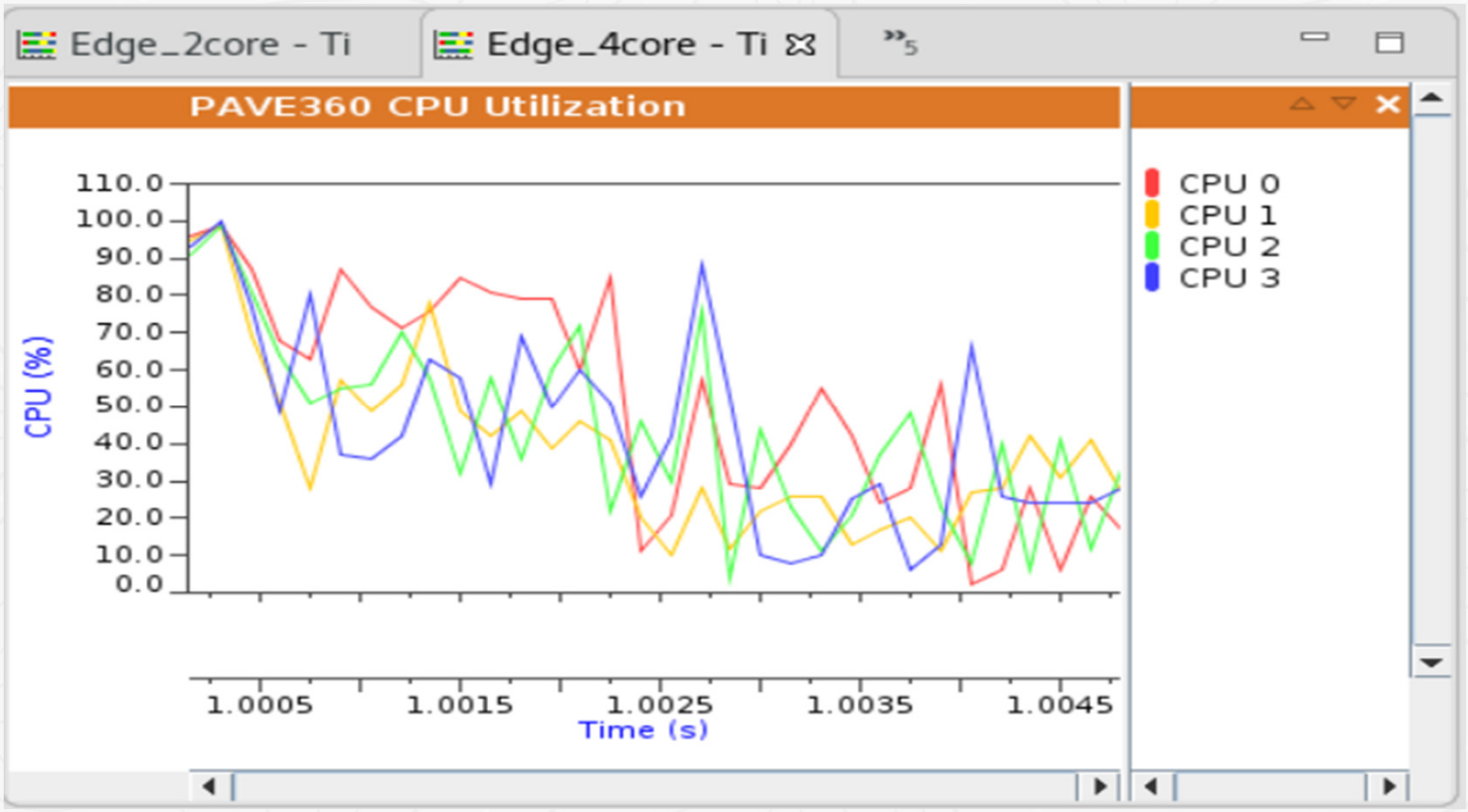
HW 8-core with Affinity Enabled (limit SW to 4-core):

CPU Utilization 2-Core Edge SW (App2)



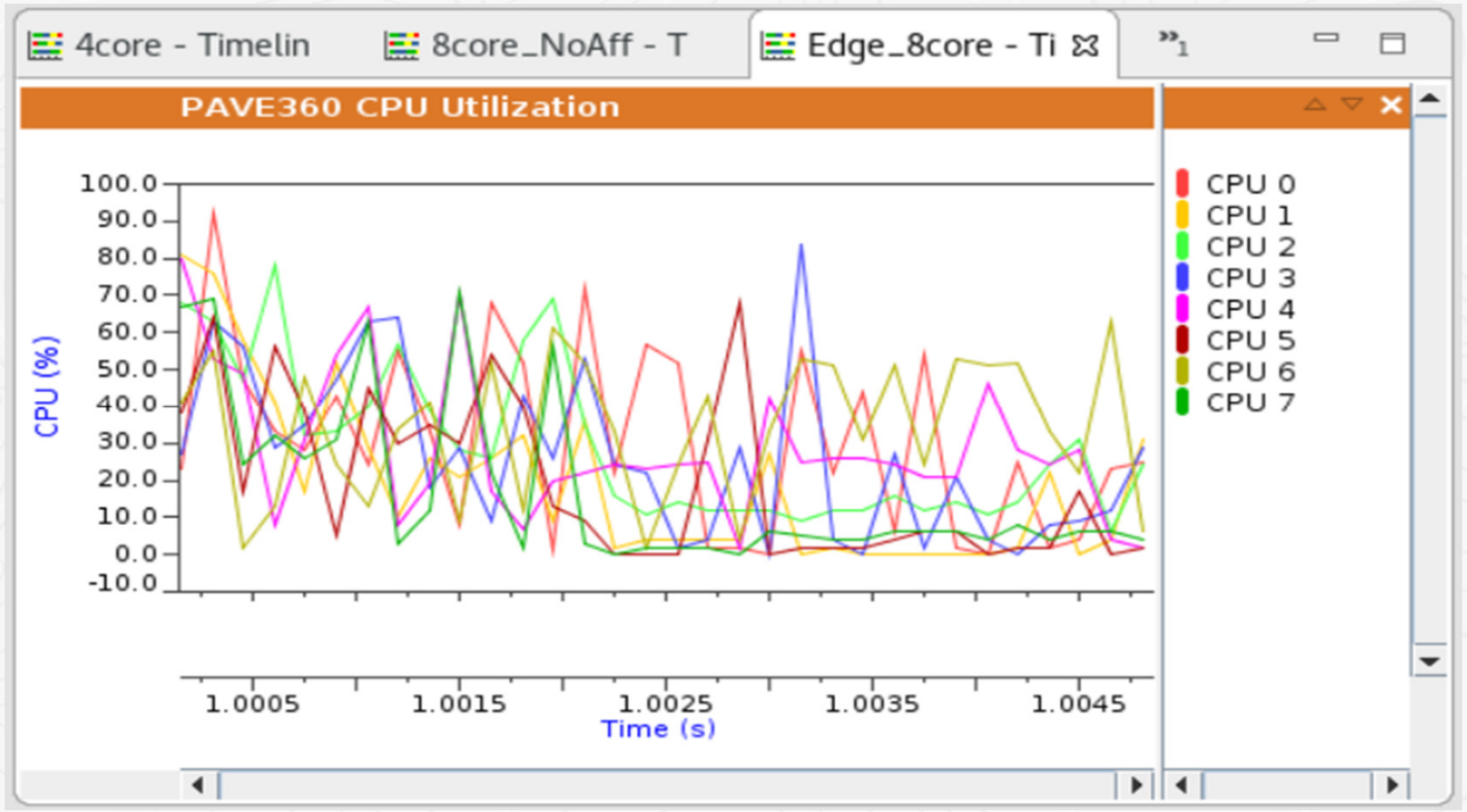
HW 2-core

CPU Utilization 4-Core Edge SW (App2)



HW 4-core

CPU Utilization 8-Core Edge SW (App2)



HW 8-core

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

Changing the platform

Summary

Reconfiguring the Platform

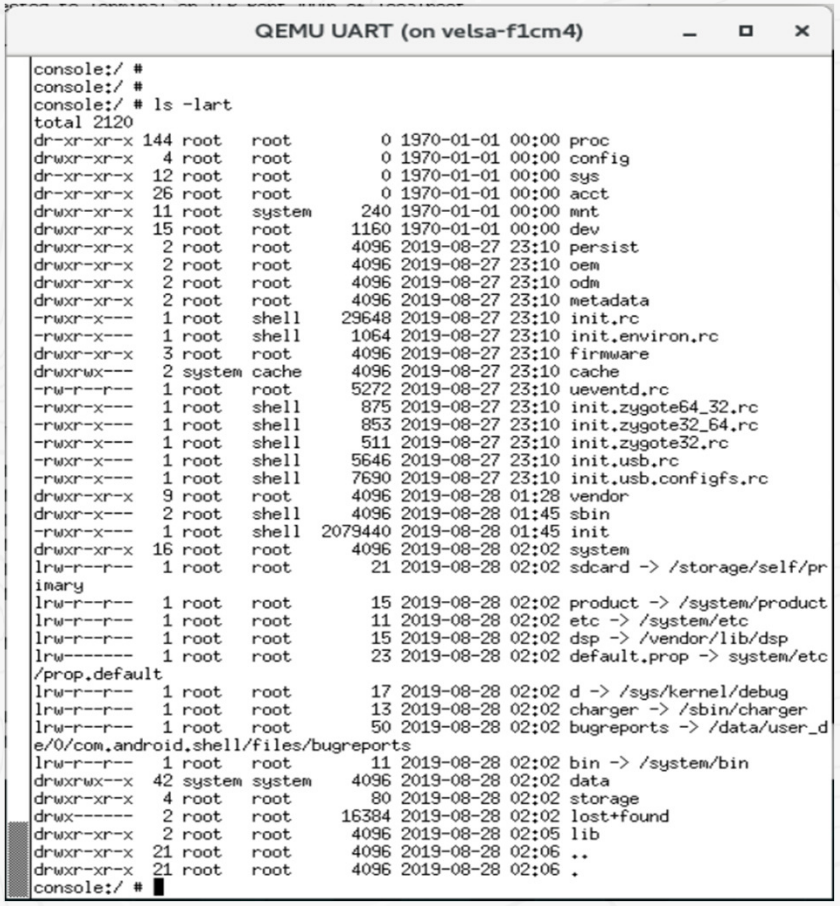
- **The standard way to re-configure the platform is to use *hycon-configure***
  - It provides couple of options to configure the platform HW and SW
  - An option “--list” gives the current configurations
    - hycon-configure - -list
  - An option “--interactive” gives interactive way to re-configure
    - hycon-configure - -interactive

```
bash-4.2$ hycon-configure --list
CONFIG_DESIGN_CPU_MODELS='CORTEXA76AEx2'
CONFIG_DESIGN_GIC_MODEL='GIC400'
CONFIG_DESIGN_INTERCONNECT_MODEL='CCI550'
CONFIG_MEMORY_SIZE='6GB'
CONFIG_FLEXMEM='n'
CONFIG_HEADLESS='n'
CONFIG_DISPLAY='hdlcd'
CONFIG_DISP_RES='1280x720'
CONFIG_DISP_DEPTH='32'
CONFIG_FILESYSTEM_BASE='$HYCON_PLATFORM_DIR/sw/output/images'
CONFIG_FILESYSTEM_LIST='rootfs.ext4'
CONFIG_KERNEL='$HYCON_PLATFORM_DIR/sw/output/images/Image'
CONFIG_DTB='$HYCON_PLATFORM_DIR/sw/output/images/board.dtb'
CONFIG_RAMDISK_PROVIDED='n'
CONFIG_ROOT_OPTION='root=/dev/vda'
CONFIG_INTERCONNECT='library'
CONFIG_INTERCONNECT_QUESTA_LIB=''
CONFIG_INTERCONNECT_VELOCE_LIB_MAP='CCI550_lib:/home/srreddy/voyager/SPP/veloce/
CCI-550/r1p0/cci550_si5dvm2mem3sys3/veloce/v21_0_1/CCI550_lib'
CONFIG_SLA='n'
CONFIG_USER_IP='none'
CONFIG_RFRA='y'
CONFIG_RECORD_CODELINK='n'
CONFIG_CPU='library'
CONFIG_CPU_QUESTA_LIB=''
CONFIG_CPU_VELOCE_LIB_MAP='MP095_Cortex_A76AE_lib:/home/srreddy/voyager/SPP/velo
ce/MP095_Cortex_A76AE/r1p0/rtl_2core/veloce/v21_0_1/MP095_Cortex_A76AE_lib'
CONFIG_GIC='library'
CONFIG_GIC_QUESTA_LIB=''
CONFIG_GIC_VELOCE_LIB_MAP='GIC400_lib:/home/srreddy/voyager/SPP/veloce/GIC/veloc
e/v21_0_1/GIC400_lib'
CONFIG_RTL_BUILD_SCRIPT=' '
CONFIG_TLM_DESIGN='y'
```

```
bash-4.2$ hycon-configure --interactive
CPU cluster models in design (format: <cpu name>x<num cores>x<num clusters>):
  (current: CORTEXA76AEx2)
GIC model name:
  0: GIC400 (current)
  1: GIC500
  2: GIC600
Please select the number of the desired option:
Interconnect model name:
  0: No
  1: CCI550 (current)
  2: CMN600
Please select the number of the desired option:
How much memory is present?
  0: 256MB
  1: 512MB
  2: 1GB
  3: 2GB
  4: 3GB
  5: 4GB
  6: 6GB (current)
Please select the number of the desired option:
CONFIG_FLEXMEM:  (y/N):
Run without host visualization windows. (y/N):
Display controller model:
  0: None
  1: ARM HDLCD Controller (current)
Please select the number of the desired option:
Display resolution:
  (current: 1280x720)
Display color depth:
  0: 8
  1: 16
```

Installing Different SW

- PAVE360 applications are easily installed on the platform
- You have Linux based UART terminal to communicate with the platform machine to install any application and run it
- It is good practice to be able to run your application in batch mode to allow automation
- Also, the target file system is mapped to the host to easily move data between them



Changing the Emulation Execution

- **Emulation execution is done through script to maintain determinism and automation.**
- **To change the Emulation time, you need to change only one value in the script.**

```
hwtrace off
#codelink veloce record on
source $env(WARPCORE_HOME)/include/warpcore_command.tcl

if {[file exists $env(HYCON_SIMULATION_DIR)/user-run.do] == 1} {
    source $env(HYCON_SIMULATION_DIR)/user-run.do
}

### *** Booting OS ***
### ******
### run for Simulation Time (ST) = 250msec for Android to boot (to have a deterministic run)
### This could change with each OS
### ******
#run 250ms

### increase the resolution to check the Utilization
### 200msec
#run 200ms
### 1sec
run 1s
run
catch {exec hycon-adb shell " mount -t 9p -o trans-virtio dropbox /data/robok " }
catch {exec hycon-adb shell " echo
'****cpu_Utilization_After_1sec_(RoboK_is_running_in_RF)****' >  /data/robok/
cpu_utilization/cat_proc_stat_new.txt" }
catch {exec hycon-adb shell " cat /proc/stat >>  /data/robok/cpu_utilization/
cat_proc_stat_new.txt" }
stop
run 150us
run
catch {exec hycon-adb shell " echo
```

```
# for { set i 0 } { $i<2 } { incr i } {
### Will do only two seconds for testing
 for { set i 0 } { $i<30 } { incr i } {
run 1s
run
catch {exec hycon-adb shell " echo
'****cpu_Utilization_After_more_1sec_(RoboK_is_running_in_RF)****' >>  /data/robok/
cpu_utilization/cat_proc_stat_new.txt" }
catch {exec hycon-adb shell " cat /proc/stat >>  /data/robok/cpu_utilization/
cat_proc_stat_new.txt" }
stop
run 150us
run
catch {exec hycon-adb shell " echo
'****cpu_Utilization_After_more_150usec_(RoboK_is_running_in_RF)****' >>  /data/robok/
cpu_utilization/cat_proc_stat_new.txt" }
catch {exec hycon-adb shell " cat /proc/stat >>  /data/robok/cpu_utilization/
cat_proc_stat_new.txt" }
stop
}
```

Agenda

Introduction to software/hardware co-development methodology

Understanding power evaluation

Run-Fast Run-Accurate with hybrid emulation

Sampling mechanism

Automating the execution flow

Power profile and power analysis

Comparing emulated power with real HW

Execute the platform for performance

Viewing the results

Changing the platform

Summary

Summary

- A methodology to run end-user application on top of an OS in an emulation environment was presented
- Methodology enables running the full end-user application in minutes not years
- Methodology allows switching to the RTL accurate hardware model when needed
- Switching to RTL allows collecting valuable data about the power and performance of the SW/HW
- Whole flow takes days not years to get the power and performance profiles
- Analyzing the effect of changing the SW and/or the HW on the power and performance profiles is easier with the presented methodology

Questions