Volume : 4 | Issue : 6 | June 2015 ISSN - 2250-1991

Research Paper Engineering

Development of an Automation Framework for Power Measurement at Rail Level for Intel's Android Based Mobile Platforms.

Srinidhi S	M.Tech Student, Dept. of Telecommunication, R V College of Engineering, Bangalore, India
Teerth Reddy	Engineering Manager, Client Computing Group, Intel Technologies, Bangalore, India
Sujith Thomas	Engineering Manager, Client Computing Group, Intel Technologies, Bangalore, India
Bhagya R	Asst.Professor, Dept. Of Telecommunication, R V College of Engineering, Bangalore, India

ABSTRACT

Power measurement, analysis and optimization of smartphones and tablets is an active area of discussion.[1]Lack of tools to measure power at rail levels makes this problem even bigger and complex. This automation frame work gets the power consumption breakdown at rail levels for different test scenarios (use case). Tools like this are critical to ensure building of competitive products. It drastically reduces the time consumed to manually run a test and the human errors induced in manual testing. This framework outputs a highly detailed breakup of power numbers for the entire platform. Most importantly, this framework can be easily scalable to adapt to any new gen phone/tablet platform and benefit the future generation of Intel SoC.

KEYWORDS Platform, Rail level breakdown, Breakup of power numbers, Platform, use case.

I. INTRODUCTION

Code updates and bug fixes of user space components gets updated on a daily basis hence there is a need for power measurementand testing of daily builds to catch the regression and fix issues as soon as possible. This tool can do this in automated way for all Intel tablet and smartphone platforms. It can run the entire test suite without any manual intervention thereby saving the valuable time.

Fig 1. Shows the breakup diagram of each block of a generic smart phone/tablet platform. Each of the blocks have a set of voltage rails connected to them. PMIC is the Power Monitor Integrated Circuit which is used to power up different blocks. PMIC has the control over the entire system. Each rail will have a sense resistor attached to it. The voltage measurement is shown in Fig.2 and current measurementis shown in Fig. 3.

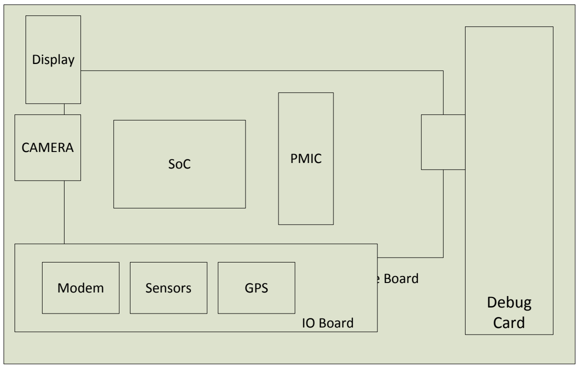


Fig.1 – Block Diagram of a Smart phone/ Tablet

Debug card is used to access the system BIOS and to get the serial logs and critical event notifications.

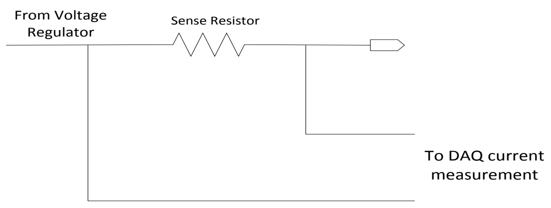


Fig.2 – Current measurement using a sense resistor

Sense resistors will generally be in the order of Milli/Micro ohms.

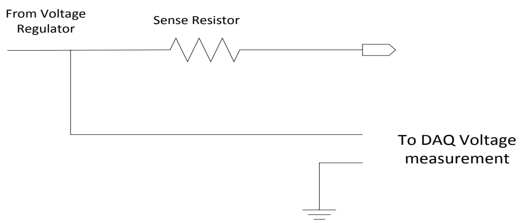


Fig.3 – Voltage measurement using a sense resistor

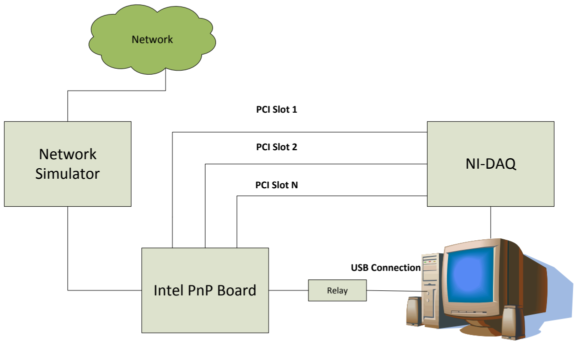


Fig.4 – Automated Hardware setup illustrating all physical connections

Network Simulator: In some test environment, the platform needs to be connected to a 2G or 3G or LTE environment. Network simulator simulates/duplicates the actual network behavior. In other words, the network simulator is a configurable base station which can mimic any network scenario. The simulator used is an Agilent 8960 network simulator. [2]

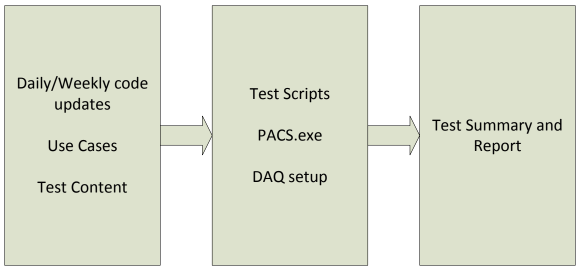
Relay: It is a switching device which is used to enable USB pass through. It automatically connects/disconnects the platform with the Host PC. This is mainly required because, all platforms has USB charging capabilities. Therefore, when a test case is run, the USB power from the HOST PC adds up to the actual power number resulting in wrong numbers. Therefore, the platform needs to be disconnected when the power measurement starts. The USB relay is controlled using a COM port. [3]

*NI-DAQ:*It is a Data acquisition tool from National Instruments with the sampling rate of approximately 5M samples per second. It is controlled by host PC running Windows. The platform is connected to the DAQ using PXI slots [4]

Apart from the above shown hardware setup, a host of soft wares are used to automate the test suite.

Test scripts are coded using Python 2.7. Variety of internal and external libraries are used. For data acquisition, a software called PACS is used. PACS uses NI-LabView libraries to capture the data. The software part is discussed in detail in the upcoming sections.

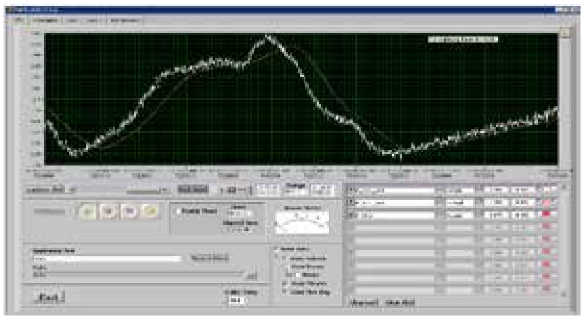
III. SOFTWARE SETUP



*Use Case:*It is a predefined test methodology or a set of rules to measure power in different environment scenarios. It mimics the user behavior at different instances. Ex: Idle Display (also called as standby) – In this particular use case, the platform is set to a particular display time out and it is uninterrupted. The display brightness is set to X nits and measured using nits meter. From the user prospective, it is the behavior of the user when his device's display is ON but he is not indulged in any activity on the platform. The major power consumption blocks in this Use case will generally be Display, Modem (if connected to the network). Other Blocks should ideally be very minimal (in terms of micro watts). These blocks would be consuming very less power due to paging and other activities like interrupts[6] happening on a frequent basis.[5]

*Test scripts:*These are basically python codes which defines the behavior of the platform depending on the use case.

*PACS:*Power Automation and Control System is a software which captures the power data fetched from DAQ and outputs it in a user readable .csv file format. Fig.6 shows the PACS user interface



Summary: This is the end result generated at the end of each run. This file shows the voltage, current and power breakup of each rail at each instance of time. DAQ can be configured for different sampling rates. Higher the sampling rate, more accurate will be the data we get.

IV. WORKFLOW

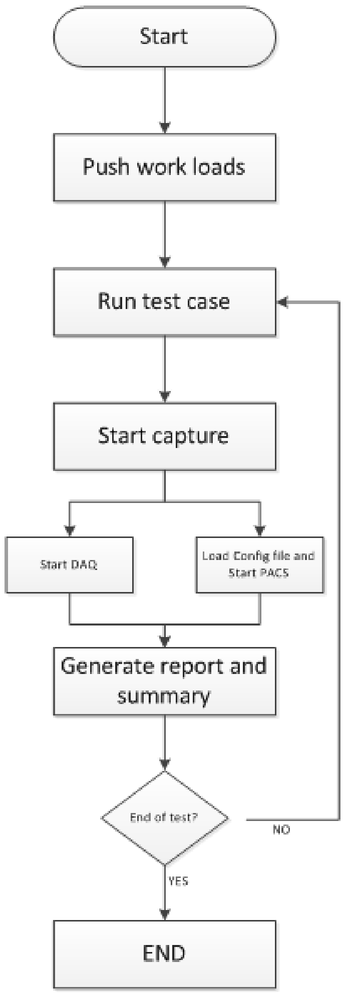
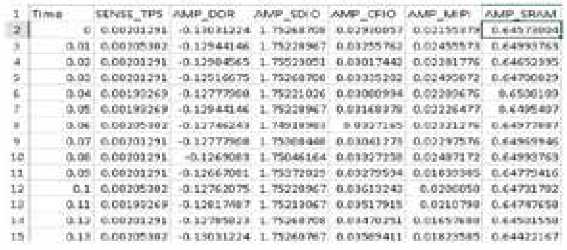


Fig.7 – Program flow diagram

- ➢ Platform is flashed (flashing is the process of updating the platform OS to a newer/older version) with a new build, powered up, booted and is left to settle for ~5 minutes. The reason to initially wait for 5 minutes is, after every re-boot, few services from Google will start by default. Therefore, if the recording starts during this period, it invariably results in wrong numbers.
- ➢ Workloads are those files (.apk, video, audio etc.) which needs to be there in the device during the test run. Depending on the use case that starts to run, the device uses the corresponding files. For ex: If, Video playback test is being run, then the corresponding video file should be playedand the power capture need to begin after the video starts playing.
- ➢ Test scripts are the python scripts which executes depending on the use case data we are interested in capturing. These scripts automates the whole manual procedure.
- ➢ After the successful beginning of the test case, Rail level power capturing will happen.
- ➢ This happens in 2 phases: DAQ is started in the first phase. This is the hardware that captures the voltage and currentconsumed by each rail. In the second phase, PACS is launched. PACS reads the data from DAQ and outputs the same in graphical as well as user readable form (.csv).
- ➢ The run may happen for few seconds or can stretch up to several minutes depending on the usecase requirement.
- ➢ After successfully running the test case, a report is generated showing the average power consumed by each rail. There will be hundreds of rails running to different modules. The report shows the average voltage, current and power consumed by each rail.
- ➢ After each use case run, the framework looks if there is any more use cases to be run. If yes, the process restarts by running the next test case without any manual intervention. If No, the process ends and the setup goes Idle.
- ➢ This process helps to run test cases overnight without intervention thereby saving the valuable time and also avoids the human errors caused by manual testing.
- ➢ With this automation, the reliability of the test has been tremendously improved.

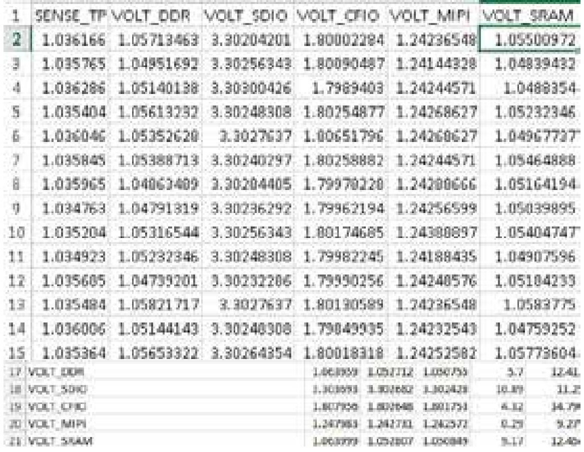
V. RESULTS

The below screen shots shows the results generated after each test run.



1	Time	SENSE_TPS	AMP_DDR	AMP_SDIO	AMP_CFIO	AMP_MIPI	AMP_SRAM
2	0	0.00201291	-0.13031224	1.75268708	0.02930057	0.02195879	0.64573004
3	0.01	0.00205302	-0.12944146	1.75228967	0.03255762	0.02455573	0.64993763
4	0.02	0.00201291	-0.12904565	1.75523051	0.03017442	0.02281776	0.64652595
5	0.03	0.00201291	-0.12516675	1.75268708	0.03335202	0.02495072	0.64700929
6	0.04	0.00193269	-0.12777968	1.75221026	0.03000994	0.02289676	0.6530193
7	0.05	0.00193269	-0.12944146	1.75228967	0.03168978	0.02226477	0.6495407
8	0.06	0.00205302	-0.12746243	1.74919983	0.0327165	0.02221276	0.64977887
9	0.07	0.00201291	-0.12777968	1.75308468	0.03041775	0.02297576	0.64969946
10	0.08	0.00201291	-0.1269083	1.75046164	0.03327758	0.02487172	0.64993763
11	0.09	0.00201291	-0.12667081	1.75372025	0.03279534	0.01630385	0.64775416
12	0.1	0.00205302	-0.12762075	1.75220967	0.03613243	0.0200050	0.64731792
13	0.11	0.00193269	-0.12617487	1.75213067	0.03517915	0.0210798	0.64747658
14	0.12	0.00201291	-0.12785023	1.75268708	0.03470291	0.01657688	0.64501558
15	0.13	0.00205302	-0.13031224	1.75260767	0.01589411	0.01873585	0.64422167

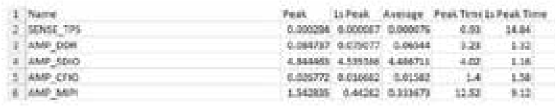
Fig.8– Current sampling trace.



1	SENSE_TP	VOLT_DDR	VOLT_SDIO	VOLT_CFIO	VOLT_MIPI	VOLT_SRAM
2	1.036166	1.05713463	3.30204201	1.80002284	1.24236548	1.05500972
3	1.035765	1.04951692	3.30256343	1.80090487	1.24144328	1.04839432
4	1.036286	1.05140138	3.30300426	1.7989403	1.24244571	1.0488354
5	1.035404	1.05613232	3.30248308	1.80254877	1.24268627	1.05232346
6	1.036046	1.05352620	3.3027637	1.80651796	1.24260627	1.04967727
7	1.035845	1.05388713	3.30240297	1.80258882	1.24244571	1.05464888
8	1.035965	1.04063409	3.30204405	1.79970220	1.24200666	1.05164194
9	1.034763	1.04791319	3.30236292	1.79962194	1.24256599	1.05039895
10	1.035204	1.05316544	3.30256343	1.80174685	1.24380897	1.05404747
11	1.034923	1.05232346	3.30248308	1.79982245	1.24188435	1.04907596
12	1.035605	1.04739201	3.30232206	1.79990256	1.24240576	1.05104233
13	1.035484	1.05821717	3.3027637	1.80130589	1.24236548	1.0583775
14	1.036006	1.05144143	3.30240308	1.79849935	1.24232543	1.04759252
15	1.035364	1.05653322	3.30264354	1.80018318	1.24252582	1.05773604

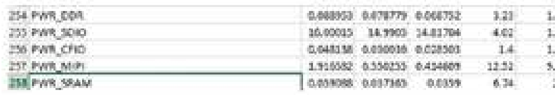
17	VOLT_DDR	1.063959	1.052712	1.050755	5.7	12.41
18	VOLT_SDIO	3.303093	3.302682	3.302428	10.89	11.2
19	VOLT_CFIO	1.807956	1.802648	1.801751	4.12	14.79
20	VOLT_MIPI	1.247983	1.242731	1.242572	0.29	9.27
21	VOLT_SRAM	1.063999	1.052807	1.050849	5.17	12.46

Fig.10– Current readings



1	Name	Peak	1s Peak	Average	Peak Time	1s Peak Time
2	SENSE_TPS	0.000294	0.000087	0.000076	0.93	14.84
3	AMP_DDR	0.084737	0.075077	0.06044	3.23	1.32
4	AMP_SDIO	4.844465	4.535306	4.486711	4.02	1.16
5	AMP_CFIO	0.025772	0.016662	0.01582	1.4	1.58
6	AMP_MIPI	1.542835	0.44262	0.333673	12.52	9.12

Fig.11– Voltage readings



254	PWR_DDR	0.088953	0.078779	0.068752	3.21	1.32
255	PWR_SDIO	16.00015	14.9903	14.81784	4.02	1.30
256	PWR_CFIO	0.048158	0.030016	0.028303	1.4	1.58
257	PWR_MIPI	1.916582	0.550233	0.424609	12.52	9.12
258	PWR_SRAM	0.039088	0.037365	0.0359	6.34	2.5

Analysis:

- Fig.8 shows the current captured in few rails at the rate of 100 samples per second. It shows the instantaneous value of current at that point of time. Similar to that of current, voltage samples are read at the same sampling rate. Fig. 9 interprets that. This is a .csv file generated after every run.
- Fig.10 and Fig.11 summarizes the average of current and voltage readings of each rail from Fig.8 and Fig.9 respectively.
- Fig. 12 shows the average power consumed by each rail.
- All these data are generated with one click of a mouse and absolutely without any manual intervention.

REFERENCES

[1]Ge Bai, HansiMou, YinhongHou, YongqiangLyu, Weikang yang, "Android Power Management and Analyses of Power Consumption in an Android Smartphone," IEEE International Conference on High Performance Computing and Communications,2013. | [2] http://cp.literature.agilent.com/litweb/pdf/1000-1903.pdf | [3] http://www.robot-electronics.co.uk/acatalog/USB-Relay-Modules.html | [4] http://www.ni.com/data-acquisition/pci/ | [5] Rabiul Islam, Anil Sabbavarapu and Rajesh Patel, "Power Reduction Schemes in Next Generation Intel® ATOM™ Processor Based SoC for Handheld Applications", 2010 Symposium on VLSI Circuits/Technical Digest of Technical Papers. | [6] Rafael J. Wysocki, "Fixing PCI Suspend and Resume", University of Warsaw, Faculty of Physics, Hoża 69, 00-681 Warsaw, 2009Linux Symposium.