VideoWall Bench: A Benchmark for Evaluating Hardware Accelerated Video Decoding on Linux

By

Muhammad Amirul Syahmi Bin Hamzah

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information & Communication Technology)

SEPTEMBER 2012

Universiti Teknologi PETRONAS
Bandar Seri Iskandar,
31750 Tronoh
Perak DarulRidzuan

CERTIFICATION OF APPROVAL

VideoWall Bench: A Benchmark for Evaluating Hardware Accelerated Video Decoding on Linux

By

Muhammad Amirul Syahmi Bin Hamzah

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION & COMMUNICATION TECHNOLOGY)

Approved by,

____

(Dr. Suziah Binti Sulaiman)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

September 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the reference and acknowledgements, and that the original work contained herein has not been undertaken or done by unspecified sources or persons.

____

(MUHAMMAD AMIRUL SYAHMI BIN HAMZAH)

ABSTRACT

VideoWall Bench is a benchmark script for benchmarking video decoding capabilities using hardware acceleration on Linux. Intel has introduced Video Acceleration API (VA-API) which enabled and provides access for graphics hardware to do hardware acceleration. VA API provides a set of video decoders (Codecs) for the H.264 video standards. Multiple video decoding using video wall methodology is a method of benchmarking that be implemented in this script. Using this method, users can really stress the multiple video decoding capabilities of one platform and at the same time measure processor usage for video decoding process. VideoWall Bench benchmark video decoding performance by measuring processor utilization, memory utilization, total frame rate per second (FPS) and time fluctuation in video decoding process. Additionally, VideoWall Bench also includes set of 1080p and 720p files for input sequences in video decoding workload process.

ACKNOWLEDGEMENT

First and foremost, the author would like to take this opportunity to express his greatest gratitude and appreciation to my supervisor, Dr. Suziah Binti Sulaiman, who had continuously monitored his progress throughout the duration of the project. Her constructive comments and advices, continuous positive support and guidance, and good suggestions towards project enhancement have guided the project towards its successful final outcome.

This gratitude also dedicated towards Mr. Muhammad Syazwan Mazlan, Mr. Andreas Grois and Mr. Dennis E. Mungai for their feedback and kind cooperation which have helped a lot in developing, improving and implementation of the system prototype. Not to forget to the committee of Final Year Project of Computer Information Sciences (CIS) department for excellent organization and management of this course.

Last but not least, I would like to thank my parents for their unconditional support, both financially and emotionally throughout my degree. In particular, the patience and understanding shown by my mum and dad during the honours year are greatly appreciated. Thank you so much.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOMENCLATURES

VAAPI	Video Acceleration API
FPS	Frame Rate per Second
HD	High Definition
EEMBC	Embedded Microprocessor Benchmark Consortium
SIMD	Single Instruction, Multiple Data
SPEC	The Standard Performance Evaluation Cooperative Consortium
CPU	Central Processing Unit
GPU	Graphic Processing Unit
XvMC	X-Video Motion Compensation
GMA	Graphics Media Accelerator
VDPAU	Video Decode and Presentation API for Unix
BIOS	Basic Input-Output System
XvMC	X-Video Motion Compensation

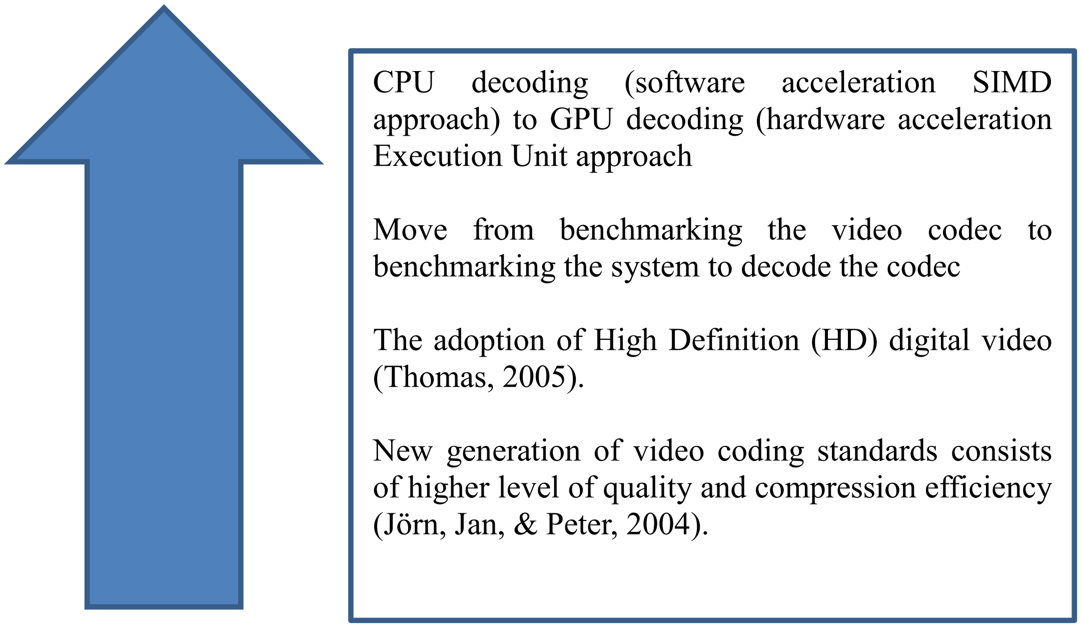
CHAPTER 1

INTRODUCTION

1.1 Background of Study

Nowadays video applications are becoming a very important workload in multiple computing environments, ranging from mobile media players to Internet servers. In order to deliver the increasing levels of quality and compression efficiency that new multimedia applications are demanding, in the recent years a new generation of video coding standards have been defined (Jörn, Jan, & Peter, 2004). Furthermore, the trend towards high quality video systems has pushed the adoption of High Definition (HD) digital video (Thomas, 2005). The combination of the complexity of new video Codecs and the higher quality of HD systems has resulted in an important increase in the computational requirements of the emerging video applications (Jörn et al., 2004). The most important part is current processors in the market also become much more powerful compared to previous generation processors. Introduction of internal and external graphic card make the processor become more efficient to process multimedia workloads especially in video decoding process (Guobin Shen et al., 2005).

While the video quality and resolution across these systems currently varies considerably based on the computing capabilities of the systems, we can expect two trends to continue indefinitely into the future: (1) research in information theory will continue to develop increasingly sophisticated methods for maximizing video compression, and (2) users will demand to watch higher quality video at lower CPU usage. Furthermore, new architectures are being recommended with the objective of providing the required performance of HD video applications (Michael, Yukio, & Takeshi, 2006). All of the improvement that have been to video decoding process such better the codec compression, and processor technology either using hardware or software acceleration requires a representative benchmark with a well-defined operation environment.



CPU decoding (software acceleration SIMD approach) to GPU decoding (hardware acceleration Execution Unit approach

Move from benchmarking the video codec to benchmarking the system to decode the codec

The adoption of High Definition (HD) digital video (Thomas, 2005).

New generation of video coding standards consists of higher level of quality and compression efficiency (Jörn, Jan, & Peter, 2004).

Figure 1: Multimedia Trends

VideoWall Bench is an application that has been made for benchmarking video decoding capabilities using VA API technology. Intel has introduced Video Acceleration API (VA-API) which enabled and provides access for graphics hardware to do hardware acceleration. Decoding using hardware acceleration will take minimum processor usage thus enable the system to decode more video in the same time. This program enable user to measure processor usage for multiple video decoding and total frame rate per second of video decoding.

1.2 Project Significance

Video decoding process requires high amount of CPU utilization. One of the examples that require powerful video decoding capabilities is a Digital Security Surveillance (DSS). It needs a computer system that capable of handling multiple HD video streams and driving a large digital display. Therefore a VideoWall Bench was been developed to benchmark and collect decoding performance data. It will assist developers to fully utilize their system resources by having an optimum and cost saving decoding system

1.3 Problem Statements

There are 3 main problem statements of this project which are:

- Current video decoding benchmarking system does not optimize hardware acceleration for external and internal graphics
- Can only benchmark one video decode at one time, thus not maximizing decoding capability of the system
- Current video decoding benchmark does not include new video codec MKV

There are several multimedia benchmarks, such as Media Bench (Jason et al., 2009), Berkeley Multimedia Workload (Nathan et al., 2002), EEMBC (Markus, 2005) and HD-VideoBench (Mauricio et al., 2007), but none of them fulfills all the requirements for a complete HD video benchmark. Some of them use the reference versions of the applications that were written with the purpose of validating the standards but not for high performance. Furthermore, these reference codes usually do not include machine specific optimizations like SIMD instructions. Additionally, most of the existing benchmarks focus on the MPEG-2 (or MPEG-4 at the most), but only a few of them include recent video Codecs like H.264 that incorporates the most recent techniques in video compression technology. Plus, none of the benchmarking system include new video codec Matroska video format (MKV). Even in the case of including H.264, none of them addresses HD applications, which requires a particular and careful selection. Furthermore, most of the benchmarking system only measured codec performance not the display performance of the system.

1.4 Objectives

This project was been developed to achieve below objectives:

- To develop video decoding benchmarking that optimize hardware acceleration on Linux
- To write a set of scripts for video decoding benchmarking that could measure:
    - Processor and memory usage for one and multiple streams of video decoding
    - Total frame rate per seconds of video decoding
    - Time latency of video decoding
- To test the scripts in terms of processor and memory utilization of one system with different video codecs.

CHAPTER 2

LITERATURE REVIEW

2.1 Benchmarking

According to David (2000) benchmarking for computer science can be defined as a combination of measurement, interpretation and communication of a computer system speed or size. It is also mention that benchmarking is not necessary dealing with complete systems. Some may deal with only small portion of the system independent of other components. Unfortunately components of a computer system interaction is incredibly complex and have unpredictable frequently ways. Computers performance evaluation should be representative of applications that run on actual systems.

Four decades ago, computer performance was measured using speed of ADD instruction or a MULTIPLY instruction. After that synthetic programs and micro benchmarks were used, in 1980's computer performance was usually evaluated using small benchmarks, such as kernels extracted from applications (e.g., Lawrence Livermore Loops, Linpack, Sorting, Sieve of Eratosthenes, 8-queens problem, Tower of Hanoi) or synthetic programs such as Whetstone or Dhrystone (Weicker, 1990). Both programs were simple programs and did not compute anything useful. Many results computed during the program's run were not ever printed or used (Wichmann, 1976).

Nowadays there are many varieties of applications in computer workloads and it is not easy to create representative benchmark and have becoming a controversial issue to the benchmarking industry (John, 2005). She also mentioned that different benchmarks are appropriate for systems targeted for different purposes. Plus, it is also a fact that simple numbers such as processor speed 2.4 GHz are easy to understand. Even today, many of the people buy their computers based on their clock frequency or memory capacity as opposed to any results based on any benchmark applications.

Benchmark can be many type of different application. Most of the benchmarks used fixed amount of computation to measure the performance of the computer. The computer that performs the task in the shortest time is considered as winner. There are also throughput benchmarks, in which there is no concept of finishing the fixed amount of work. Throughput benchmarks are used to measure the rate at which work gets done, that is, a task accomplished in a fixed time is used to compare processors or systems. The SPEC CPU benchmarks are examples of fixed-computation benchmarks, whereas the TPC benchmarks are examples of throughput benchmarks. One may also design benchmarks where neither computation nor time is kept fixed. Misuse/abuse has happened in the use of these programs and in interpretation of results from these programs. Synthetic benchmarks have been in disrepute since then. The Standard Performance Evaluation Cooperative (SPEC) consortium and the Transactions Processing Council (TPC) formed in 1988 have made available several benchmark suites and benchmarking guidelines to improve the quality of benchmarking

2.2 Common Goals of Benchmarking

David (2000) mentioned that the goals of any analysis of the benchmarking of a computers system, or one of its components will depend on the specific situation and the interests, skills and abilities of the analyst. Below are the several different typical goals of benchmarking a computer that are useful both to computer system designers and to users (David, 2000)

- **Compare alternatives.** Purchasing new computer system may be a hassle for users because they may have several different systems from which to choose. Different option have different impact both cost and performance. The goal of the benchmarking in this case is to provide quantitative information about which computer set up are best under specific conditions.
- **Determine the impact of a feature.** Adding or removing specific feature may give an impact to the new systems or existing systems. Therefore this type of analysis is often referred to as before-and-after comparison since only one well-defined component of the system is changed.
- **System tuning.** The goal of benchmarking in system tuning is to discover the set of parameter that produces the best optimize performance. The overall performance perceive by the users may consist of different parameter and closely interconnected. Therefore it is very difficult task to find the best set of parameters values in maximizing the computer performance.
- **Identify relative performance.** Computer performance typically has meaning only in the context of its performance relative to another systems or same configuration of another system. The goal of this case is to quantify the change in performance relative to history
- **Performance debugging.** The goal of benchmarking for this case is to apply appropriate tools and analysis technique why the program is not meeting performance expectations.
- **Set expectations.** Computer users may have some idea what are the new capabilities that may offer in the next new line of computer generations. In this case, the task is to set the appropriate expectations for what a system is actually capable of doing.

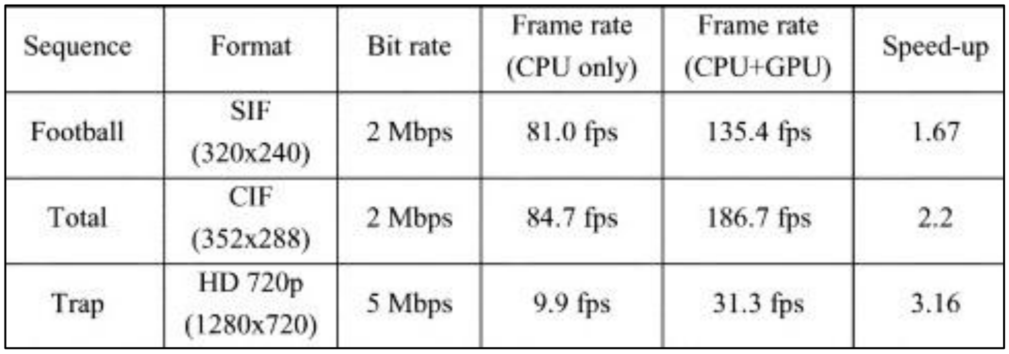
2.3 CPU Video Decoding and GPU Assisted Video Decoding

In the last decade, some multimedia-oriented SIMD processor extensions such as Intel’s Matrix Math eXtension (MMX) instructions were introduced to CPU designs. These instructions improve the performance significantly and is been heavily used throughout the multimedia applications. However CPU still heavily loaded proving that CPUs processing power cannot meet the requirement to decode high-definition (HD) video in real-time even with highly optimized code (Guobin Shen et al., 2005).

Guobin Shen et al. (2005) present a study on accelerating the digital video decoding using the programmable graphics pipeline of commodity GPU. The GPU is use to off-load some of CPUs tasks such as video decoding when the CPU is heavily loaded while GPU is idle.In fact, most today’s GPUs have a special hardware unit that can perform the video decoding process provided that the video is encoded with an established international video coding standards such as MPEG-1/2/4 and the wide-accepted DirectX video accelerator (DXVA) specification. However, the application of such hardware video decoding unit is very limited. It cannot decode video contents that are coded with a very popular video coding format such as Windows Media Video (WMV) and RealVideo.

Guobin Shen et al. (2005) paper also study on how to speed up video decoding using common DirectX-8 compatible graphics engines. They choose DirectX-8 because of its powerfulness, programmability, predominance and rich application program interfaces (APIs). Their study proves that GPUs power can be utilized for applications other than graphics such as video decoding. Furthermore, since the major task of video de-coding is already handled by the GPUs graphics engine, it provides a more efficient way to incorporate video into computer graphics. This is of high interest in today’s gaming industry.

Guobin Shen et al. (2005) also mentioned that the CPU and GPU have to be considered together because GPU alone cannot meet the requirement of video decoding. Moreover a CPU plus GPU configuration is far more popular than a dual-processor configuration in consumer commodity PCs. Guobin Shen et al. (2005) have performed extensive tests on a PC with an Intel Pentium III 667 MHz CPU, 256 MB memory, and an nVidia GeForce3 Ti200 GPU. Some of the initial experimental results are reported in table 1. The test sequences are Football, Total, and Trap.



Sequence	Format	Bit rate	Frame rate (CPU only)	Frame rate (CPU+GPU)	Speed-up
Football	SIF (320x240)	2 Mbps	81.0 fps	135.4 fps	1.67
Total	CIF (352x288)	2 Mbps	84.7 fps	186.7 fps	2.2
Trap	HD 720p (1280x720)	5 Mbps	9.9 fps	31.3 fps	3.16

Table 1: Experimental Results of CPU Video Decoding and GPU Assisted Video Decoding

The Football sequence is a standard MPEG test sequence in SIF format (320 240) with very high motion. The Total sequence is a concatenation of several Standard MPEG test sequences (such as Car phone, Stephan, Silence, Akiyo, Mobile-Calendar etc.) in CIF format (352 288). The Trap sequence is a high definition version (1280 720) of the movie trailer of “The Parent Trap” (Disney, 1998). The original frame rate of Trap is 23.98 f/s. In this experiment, they compare the video decoding speed achieved using CPU only (with MMX technology) against that achieved with GPU acceleration. It is obvious that the speed is significantly improved by leveraging the power of GPUs graphics engine. It is interesting to observe that the speed-up of Total sequence is much higher than that of Football sequence, while the speed-up of Trap is by large the most significant. This experiment has greatly proved that using GPU acceleration can make a significance difference compared to CPU only which is used software acceleration.

2.4 Theory of Video Acceleration API (VA API)

The Video Acceleration API (VA API) is a public software API specification. It provides access to graphics hardware acceleration for video processing. The API is meant to enable hardware accelerated video decode at various entry points for the current coding standards today such as MPEG-2, MPEG-4/ASP/H.263, MPEG-4 AVC/H.264, and VC-1/WMV9(Video Acceleration API, 2012). The VA API provides much more functionality than the existing X-Video Motion Compensation (XvMC) API. XvMC was designed to support MPEG-2 motion compensation only (X-Video Motion Compensation, 2012).

VA API is use at the X Window System on Unix-based operating systems (including Linux, FreeBSD, and Solaris). Originally it is designed by Intel for its Graphics Media Accelerator (GMA) series of GPU hardware. However, the API is not limited to GPUs or Intel specific hardware, as other hardware and manufacturers can also freely use this API for hardware accelerated video decoding (Willis, 2009).

2.5 Comparing VA API with other API

Larabel (2011) in his work try to compare the VA-API video playback performance with Intel Sandy to its performance of using X-Video with the same hardware. This Sandy Bridge testing was done with an Intel Core i5 2500K CPU, the Intel Bearup Lake motherboard, 2GB of DDR3 system memory, and an OCZ 60GB Vertex 2 SSD. The software stack was Ubuntu 10.10 with the Linux 2.6.38 kernel, GNOME 2.32.0, X.Org Server 1.9.0, xf86-video-intel 2.14.901 driver, GCC 4.4.5, an EXT4 file-system, Mesa 7.11-devel from the beginning of March, and the LibVaGit library from early March. The performance was also compared to X-Video playback from an ATI Radeon HD 4550 using both the open-source driver stack and the proprietary Catalyst 11.2 driver, and a NVIDIA GeForce 9500GT with the 270.30 beta driver under the X-Video API and then with their flagship Video Decode and Presentation API for Unix(VDPAU) implementation.

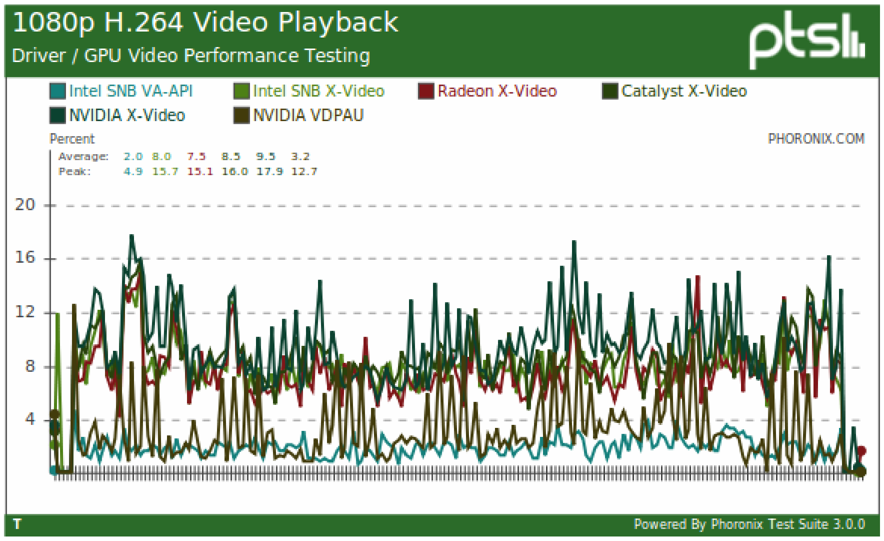


Figure 2: Driver/GPU Video Performance Testing

Figure 2 show all of the video CPU usage data for the different VA-API / X-Video / VDPAU video playback tests with the different graphics adapters. The CPU usage with VA-API and VDPAU performance is far lower than with X-Video. Moreover, common X extension does not offload much work to the GPU so the CPU is left with a much greater burden. However, with the Core i5 2500K processor, the CPU usage using X-Video is still 7~9% for this quad-core part. Between the different drivers / GPUs, the X-Video performance does not different much.

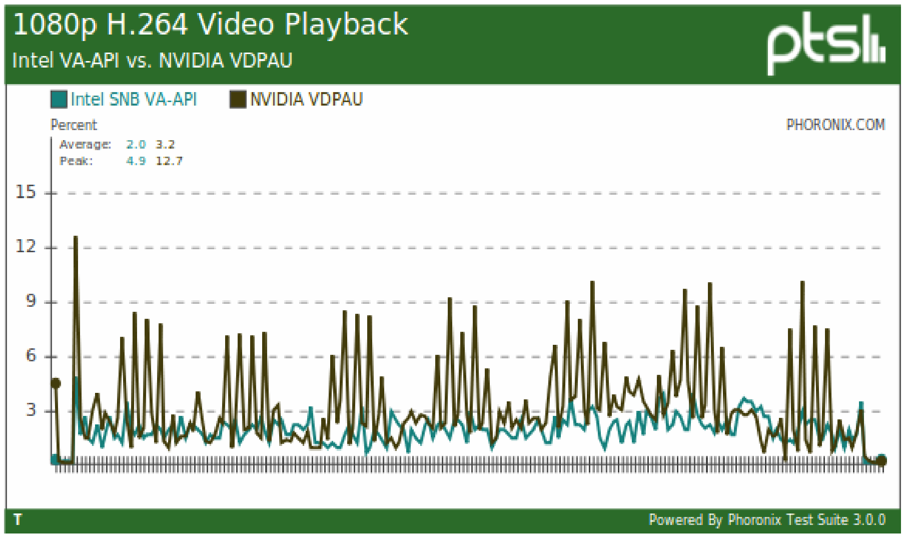


Figure 3: Intel VA API vs NVIDIA VDPAU

Figure 3 shows Phoronix Test Suite data for Sandy Bridge VA-API and NVIDIA VDPAU for the GeForce 9500GT, it can be clearly see the CPU usage when using Sandy Bridge with onboard graphics is actually lower than using the Video Decode and Presentation API for UNIX on the discrete NVIDIA card. The average CPU difference is just 2% vs. 3.2%. Besides, the CPU usages dramatically increase over the course of playing "Big Buck Bunny" 1080p H.264 with VDPAU. With VA-API, the CPU topped out at 4.9% (no other CPU work was going on in the background during any of this video testing) while the NVIDIA driver spiked to nearly 13%.

Larabel (2011) also try to down-clock the processor below 1600MHz to see how low it can possibly go with the CPU's performance while still handling Intel VA-API fine and making X-Video choke. Unfortunately the Intel H67 motherboard doesn't allow down clocking fewer than 1600MHz. However Larabel (2011) try to limit the CPU's performance by disabling three of the four CPU cores. The Core i5 2500K is limited to just one CPU core from the basic input-output system (BIOS).

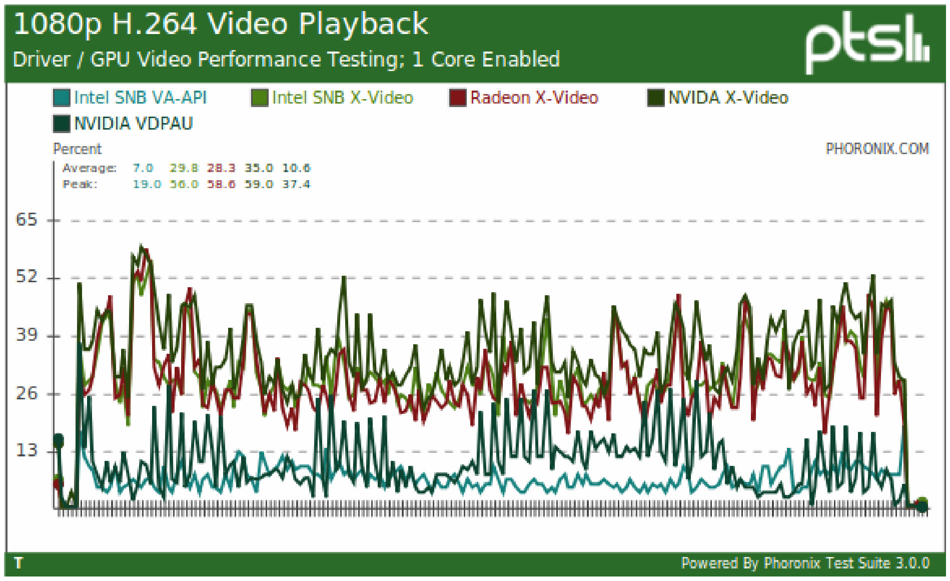


Figure 4: Driver/GPU Video Performance Testing: 1 Core Enabled

Figure 4 shows driver/GPU video performance testing when only 1 core is enabled. When just a single physical CPU core is exposed, the VA API utilization average is around 7%, NVIDIA VDPAU is around 10%, and the various X-Video implementations are 30~35%.

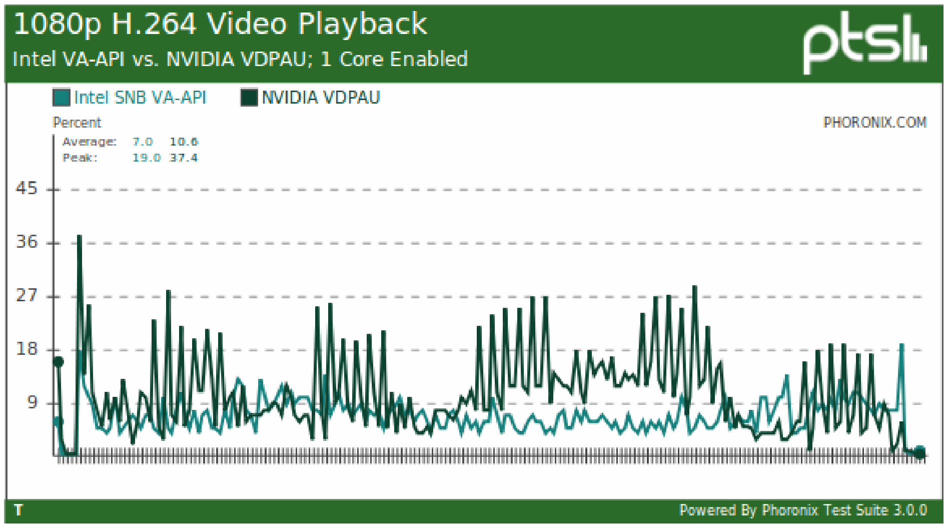


Figure 5: Intel VA API vs NVDIA VDPAU

The NVIDIA VDPAU driver continues poking the CPU more often than VA-API. Overall Larabel (2011) work already shows the relevancy why VideoWall Bench uses VA API as the API in the benchmarking system. Using VA API, CPU utilization is much lower thus it optimizes video decoding process in a system.

2.6 Previous Video Decoding Benchmarking System

Media Bench is one of the most famous multimedia benchmark (Lee, Potkanjak, & H. Mangione-Smith, 1997). This benchmarking system includes a MPEG-2 encoder and decoder based on the implementation of the MPEG Software Simulation Group (MSSG) with short input videos in low resolution (352x240 pixels). Besides, the MSSG codec does not implement SIMD optimizations thus it has low performance of video decoding. However, to solve the limitation of Media Bench, Media Bench+ was been develop to solve the limitations of Media Bench by including MPEG-4 and H.263 video codec, unfortunately it select the reference implementations and they do not address high definition video (Jason, Wayne, & Bede , 1999).

A new version of the Media Bench which is called Media Bench II has been released in which includes codecs for MPEG- 2, MPEG-4, H.263 and H.264 (Jason, Frederick, Joseph, & Wayne, 2009). The MPEG-2 Codec is using the same MSSG implementation, the MPEG-4 is taken from the FFmpeg Codec library, the H.263 Codec is the Telenor implementation, and the H.264 is taken from the reference software (called JM). The main problem with this selection is the combination of reference implementations for some of the Codecs (MSSG for MPEG-2 and JM for H.264) with highly optimized version for others (FFmpeg for MPEG-4). This shows inconsistency of quality in terms of optimization thus can jeopardize the result of the benchmark. Media Bench II also has increased the resolution compared to the original Media Bench, but unfortunately they do not address HD applications and remains on Standard Resolution (SD). Additionally, Media Bench II provides only one short input sequence (10 frames) and the coding options are not tuned for HD applications.

Intel's addition of MMX to their x86 architecture motivates them to develop the Intel Media Benchmark. It was been developed because during that time an adequate industry standard multimedia benchmark did not exist to measure multimedia performance. However the distribution of benchmark for only x86 architecture have make this benchmark greatest weaknesses. This made the benchmark system only applicable to x86 instruction set compatible processors. Moreover the Intel Media Benchmark source code is not publicly available to the users. On the other side The Berkeley Multimedia Work load solved the problem of the low resolution of the input sequences by including inputs with higher resolutions and The source code is publicly available and users are free to take the workload and modify it to suit their needs. Unfortunately they have selected only the MPEG- 2 Codec into the benchmark program (Nathan & Alan, 2002).

The EEMBC Digital Entertainment benchmark includes codec for MPEG-2 and MPEG-4 video standards which address low and standard resolutions and provide a different set of input sequences (Markus, 2005). Most benchmarks perform a fixed workload. Throughput benchmarks, on the other hand, have no concept of finishing a fixed amount of work. EEMBC's approach has always been based on a fixed workload. It uses the MPEG-x benchmark as an example. A fixed workload approach would process a video with a specific number of frames, measuring how long it took to process the entire video. Alternatively, running the benchmark for a fixed amount of time would measure the number of frames processed. Nevertheless, they do not have recent codec like H.264 and the coding options and input sequences are not publicly available.

The BDTI Video Encoder and Decoder Benchmark is a set of applications representative of modern video codecs, but they are not complete video codec applications. The codec seems to be similar to H.264 but the details of the codec. For each set of standard parameters, BDTI provides a set of input and output test data. To obtain certification, a solution must process the input test data and generate output data that matches the test output data provided by BDTI within the specified tolerance. Performance may be measured in terms of processor loading, total program and data memory use, cost, and energy consumption (BDTI H.264 Decoder Benchmark, 2006).

HD-Video Bench try to solve all the before mentioned limitations by providing different a set of different video codec applications optimized for high performance, and providing a complete set of input sequences and coding options tuned for HD applications(Mauricio, Esther, Alex, & Mateo, 2007). In HD-Video Bench application is more likely as a VideoWall Bench. It consists of MPEG-2 Application, MPEG-4 benchmarks and H264 Benchmarks. In H264 Benchmarks there are 3 types of benchmarks that have been included and based on my project which is focus on H264 video decoding FFmpeg H.264 decoder is the closest software that can decode H264 files as VideoWall Bench did. The code is much optimized with SIMD instructions and widely used in free multimedia players.

Codec	Application	Execution Command
MPEG-2 decoder	`libmpeg2`	mplayer mpeg2/576p25_blue_sky.avi -vc mpeg12 -nosound -vo null -benchmark
MPEG-2 encoder	`FFmpeg-mpeg2`	mencoder yuv/576p25_blue_sky.yuv -demuxer rawvideo -rawvideo \ fps=25:w=720:h=576 -o out/576p25_blue_sky_mpeg2.avi -ofps 25 \ -ovc lavc -lavcopts vcodec=mpeg2video:vqscale=5:vmax_b_frames=2:subq=8:psnr
MPEG-4 decoder	`Xvid`	mplayer mpeg4/576p25_blue_sky.avi -vc xvid -nosound -vo null -benchmark
MPEG-4 encoder	`Xvid`	mencoder yuv/576p25_blue_sky.yuv -demuxer rawvideo -rawvideo \ fps=25:format=i420:w=176:h=144 -o out/576p25_blue_sky_mpeg4.avi \ -ofps 25 -ovc xvid -xvidencopts fixed_quant=5:max_bframes=2:qpel:psnr
H.264 decoder	`FFmpeg-h264`	mplayer h264/576p25_blue_sky.h264 -vc ffh264 -nosound -vo null -benchmark
H.264 encoder	`x264`	x264 –bframes 2 –no-b-adapt –b-bias=0 –ref 16 –qp=26 –analyse all \ –weightb –me hex –merange 24 –subme 7 –8x8dct -fps 25 –frames 101 \ –progress -o out/576p25_blue_sky.h264 yuv/576p25_blue_sky.yuv 720x576

Table 2: Sample of Command in HD Bench

.

Test Sequence	Resolution	Frames / second	No. frames	Comments
Blue_sky	720x576 1280x720 1920x1088	25	100	Top of two trees against blue sky. High contrast, small color differences in the sky. Many details. Camera rotation.
Pedestrian_area	720x576 1280x720 1920x1088	25	100	Shot of a pedestrian area. Low camera position, people pass by very close to the camera. High depth of field. Static camera.
Riverbed	720x576 1280x720 1920x1088	25	100	Riverbed seen through the water. Very hard to code.
Rush_hour	720x576 1280x720 1920x1088	25	100	Rush-hour in Munich city. Many cars moving slowly, high depth of focus. Fixed camera.

Table 3: Input Sequences of HD-Video Bench

HD-Video Bench use Mplayer that includes support for multiple video Codecs by using FFmpeg, libmpeg2, Xvid and other multimedia libraries. Mplayer simplifies the process of installing and running multiple video libraries because Mplayer selects the appropriate Codec and uses it to encode or decode the input video. However Mauricio et al. (2007) mentioned that HD-Video Bench is interested on benchmarking the video Codecs not the displaying process. Therefore they disabled the output of the video to the screen. Below is the summary for the previous video decoding benchmarking systems.

Benchmarking System	**Video Supported**	**High Definition**	**Availability of Source Code**	**SIMD Optimization**
Media Bench	MPEG-2	No	No	No
Media Bench+	MPEG-2, MPEG-4, H.263	No	No	No
Media Bench 2	MPEG-2, MPEG-4, H.264, H.263	No	Yes	Only MPEG-2
Intel Media Benchmark	MPEG-2, MPEG-4, H.263	No	No	Yes
The Berkeley Multimedia Workload	MPEG-2	Yes	Yes	Unknown
The EEMBC Digital Entertainment	MPEG-2, MPEG-4	No	No	Yes
The BDTI Video Encoder and Decoder	H.264	Unknown	No	Unknown
HD Video Bench	MPEG-2, MPEG-4,H264	Yes	Yes	Yes

Table 4: Comparison between other Benchmarking Systems

Mauricio et al. (2007) paper also list the desired characteristics for a video benchmark

- The benchmarks should be complete applications and implement all the features defined in the standards.
- The Codecs should be optimized for high performance.
- A complete set of input sequences must be provided.
- A detailed description of the coding parameters must be provided.
- Programs and input sequences need to be free.
- The code must be portable.
- Programs must be representative of the multimedia application domain.

2.7 Proposed Solution

Based on all research given, the big differences between VideoWall Bench and the others are the usage of the VAAPI technology in video decoding application. It enabled hardware acceleration thus can lower the CPU utilization in the system. Compare with the FFmpeg H264 which are closest software to VideoWall Bench, it used software acceleration and cause higher CPU utilization for decoding process. Hardware acceleration enables the platform to decode more than one video, thus my software has been developed to decode multiple video files at the same time and the output decode will be in video wall output. In terms of performance measurement, VideoWall Bench technology will measure processor utilization, memory utilization, total frame rate per second and time latency.

CHAPTER 3

METHODOLOGY

3.1 Agile Waterfall Model

This chapter will cover the details explanation of methodology that is being used to make this project complete and working well. Many methodology or findings from this field mainly generated into journal for others to take advantages and improve as upcoming studies. The method is use to achieve the objective of the project that will accomplish a perfect result. In order to evaluate this project, the methodology of VideoWall Bench is based on Agile and Waterfall model which is more flexible provide excellent plan for software development. In addition, the author do some self-reference towards existing network books, websites, research papers and journals as well. There are few becomes main reference along the project completion. Besides, a good guidance during internship at Intel Performance Measurement Analysis do helps the project progress. Plus, the author already started to play around with the Perl script, EEMBC Test and Mplayer which is to get familiar with all the functions and capabilities.

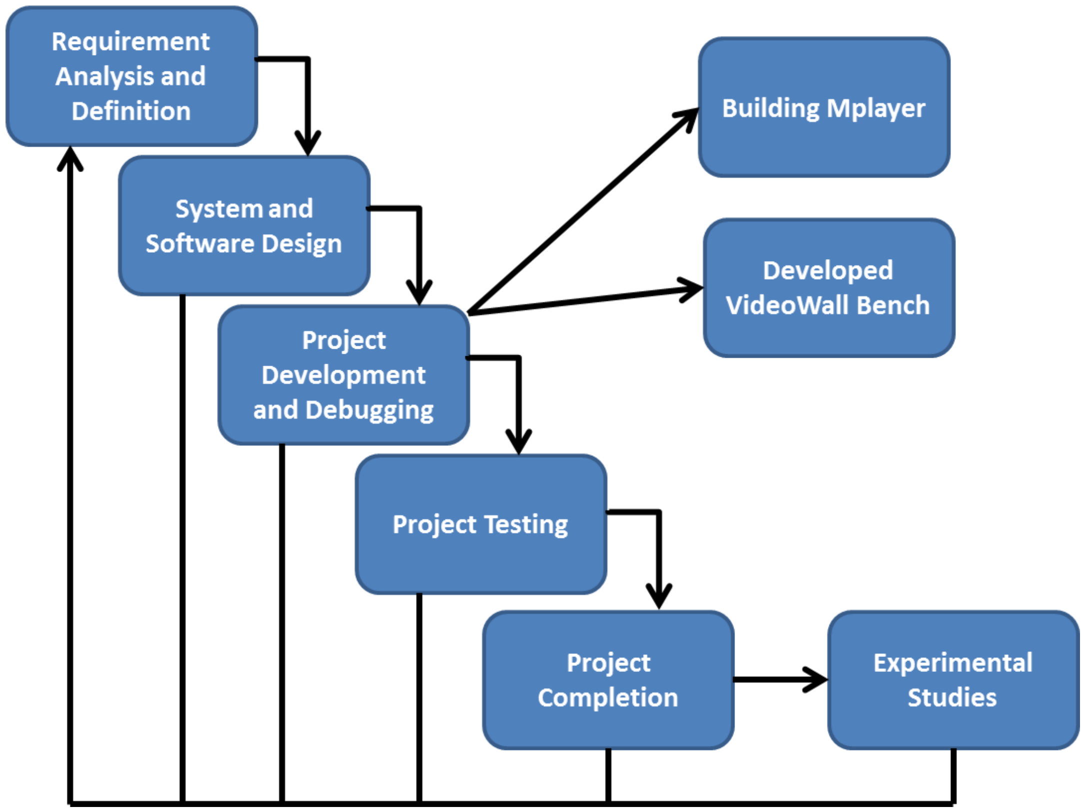

Requirement Analysis and Definition
System and Software Design
Project Development and Debugging
Project Testing
Project Completion
Experimental Studies
Building Mplayer
Developed VideoWall Bench

Figure 6: Agile Waterfall Model

<u>Phase 1: Requirement Analysis and Definition</u>

All possible requirements of the system to be developed are captured in this phase. Requirements are a set of functions and constraints that the end user (who will be using the system) expects from the system.

VideoWall Bench requirement is to decode multiple video streams and benchmark the decoding capabilities of the platform. This application also was specially designed to benchmark H.264 files in 720p and 1080p. Result of benchmark process must consist of processor utilization, frame rate per second and time latency for video playback. It also must be portable to all type of Linux operation systems. Not all processors support VideoWall Bench. In order to use this application, user must make sure that their processors support Video Acceleration API.

As been stated in Wikipedia (Video Acceleration API, 2012), below is the hardware that supports Video Acceleration API.

- The free and open source drivers of Broadcom Crystal
- The free and open source drivers of the integrated graphics of known as "Intel® HD Graphics" (Intel HD Graphics 2000/2500/3000/4000)
- The free and open source drivers of the Intel G45 chipset (with Intel GMA X4500HD integrated graphics), and later
- The closed source proprietary drivers for Intel's Poulsbo Chipset based GMA 500integrated graphics
- The closed source proprietary drivers for Atom E6xx and Penwell supported via its Media Infrastructure Accelerator (MI-X).
- The closed source proprietary drivers Intel Medfield SoCs with Imagination Technologies's PowerVR (VXD375/385 and VXE250/285) based integrated graphics.[9]
- The closed source proprietary drivers of S3 Graphics's Chrome 400 and later series are also supported.
- In November 2009, VA-API also gained a new proprietary backend named "xvba-video" which allows VA-API powered applications to take advantage of AMD Radeon's proprietary fglrx drivers for its chipsets with UVD2 support via the XvBAlibrary (X-Video Bitstream Acceleration API designed by AMD), for closed source proprietary driver only.
- Additionally, VDPAU (Video Decode and Presentation API for Unix), a competing API designed by NVIDIA, can potentially also be used as a backend for the VA API. If this is supported, any software that supports VA API then also indirectly supports a subset of VDPAU.

These requirements are analyzed for their validity, and the possibility of incorporating the requirements in the system to be developed is also studied. Finally, a requirement specification document is created which serves the purpose of guideline for the next phase of the model.

Phase 2: System and Software Design

Before starting the actual coding phase, it is highly important to understand the requirements of the end user and also have an idea of how should the end product looks like. The requirement specifications from the first phase are studied in this phase and a system design is prepared. Below is the high level algorithm for video decoding project:

```
1.0   Start
2.0   Input Sources
      2.1   Enter H264 File
      2.2   Enter Width and Height for Video resolution
      2.3   Enter Number of Video Stream
      2.4   Specify Hardware Acceleration or Software Acceleration
3.0   Program Executions
4.0   Display output
5.0   Log File Compilation
6.0   End
```

Figure 7: Basic Algorithm for VideoWall Bench

VideoWall Bench is made from Perl language with the integration of Mplayer command line code. Overall the software design should be in command line so the operating system will be in optimum condition for testing.

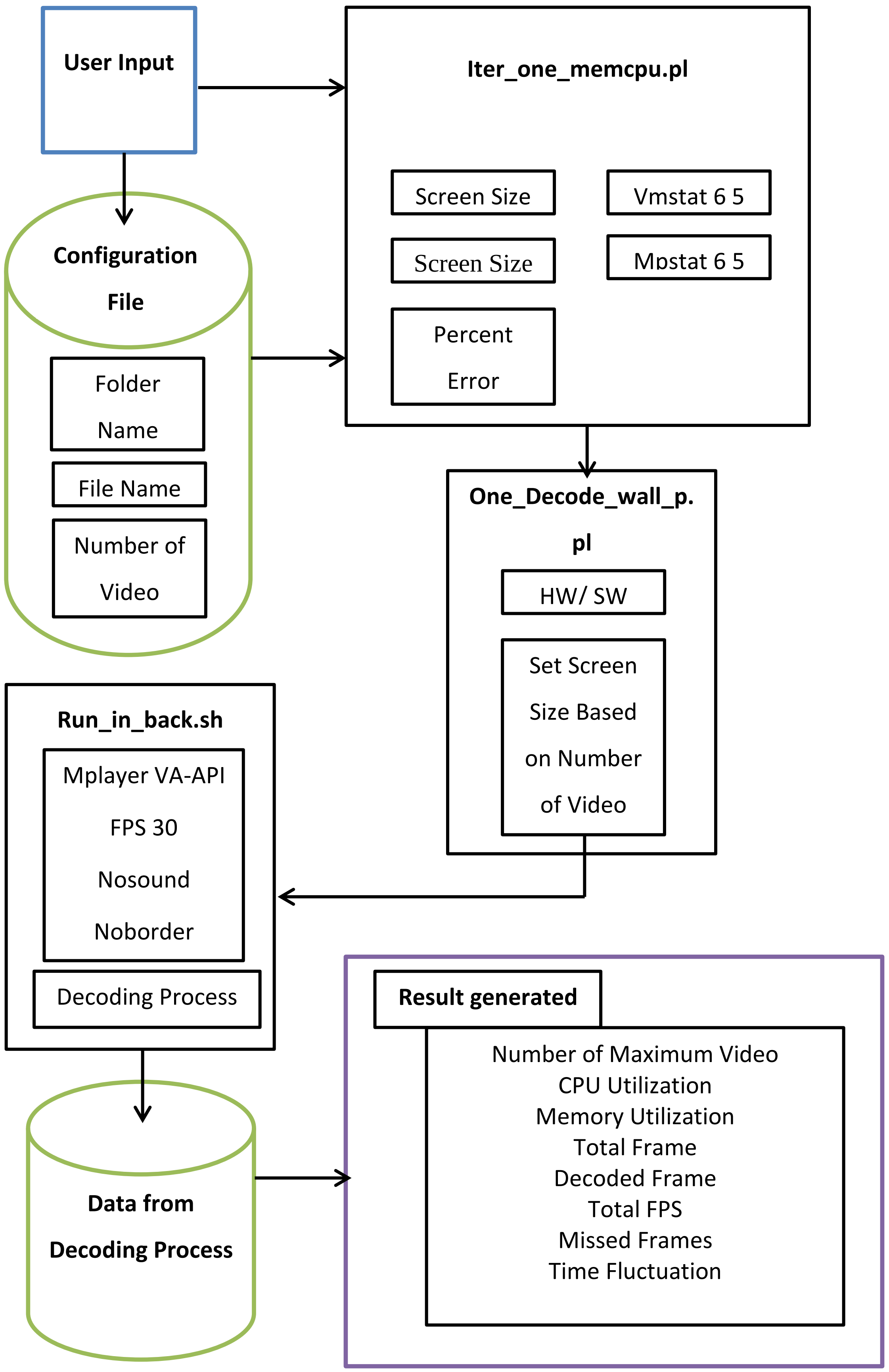


Figure 8: System Architecture

Figure 8 shows the system architecture which includes the element of user's input, database relationship details and the output to be display to user. An in depth review has been conducted on the tools that are available in order to select the most appropriate tools for the development of the VideoWall Bench. As a result, Perl language, Shell Script and Mplayer VA-API have been chosen for the development of the VideoWall Bench

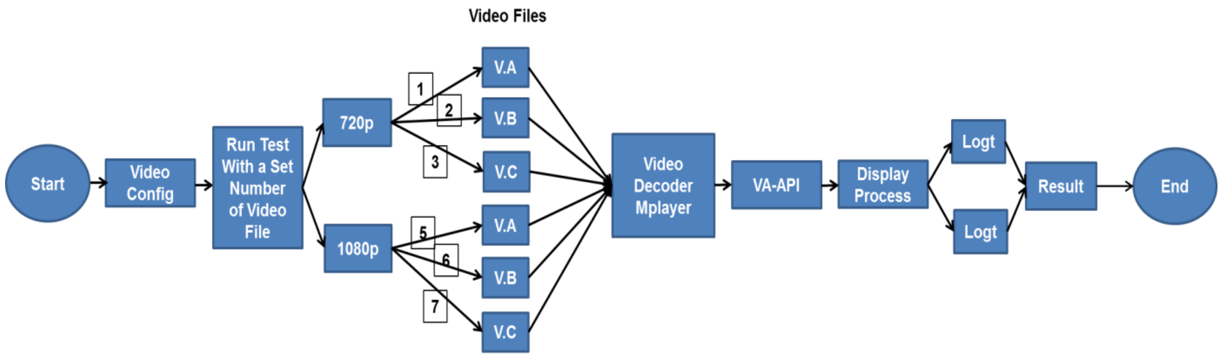


Figure 9: Process Flow Chart

In Figure 9 it shows the process flow chart for the VideoWall Bench. It will decode the 720p type first then it will move to 1080p video decode. Each of video resolution contains three files. The VideWall Bench will decode each file one after another. The decoding process will be done by Mplayer VA-API and during this time it will capture CPU utilization, memory utilization, frame per second and time latency of video decoding. After the decoding process finish it will generate two log files which are .log and .logt. Both of the log files will be compile into Result text file.

<u>Phase 3: Project Development and Debugging</u>

VideoWall Bench

In this part I will elaborate about the project development of VideoWall Bench which contains two parts:

1. Build the Mplayer VA-API (Collaboration with Andreas Grois)
2. Development of VideoWall Bench Script.

The build of Mplayer VA-API in this VideoWall Bench was collaboration between me and Andreas Grois. He already successful installing Mplayer VA-API on AMD processor while for this Mplayer is specifically for Intel processor. He is a member of the Magic Spin magnetic materials group at the Institut Für Halbleiter- und Festkörperphysik of the Johannes Kepler Universität Linz, Austria.

Users must build Mplayer VA-API before using VideoWall Bench. There are many things that need to install before we can build Mplayer VA-API. Users also must make sure that their hardware supports the usage of hardware acceleration. Below was my hardware configuration for successful building of Mplayer VA-API.

Item	Type
Processor	Intel i5-2500k
Operating System	Ubuntu 12.10
Motherboard	Asus P8Z68-V LX
Memory	Kingston 1333 Mhz 2GB x 2

Table 5: Computer Requirements

This testing used Ubuntu 12.10, but the methodology will apply for any Linux variation that supports a kernel version higher than 2.6.4. First one has to create an Ubuntu CD from a downloaded .iso file. The file can be downloaded from: http://releases.ubuntu.com/ and below are simple instructions on how to produce the CD:

1. Download the appropriate .iso file from Ubuntu's website - if you are not sure of what copy to get, get the generic “PC (Intel x86) desktop CD” .iso image.
2. After you select the appropriate .iso file you may need to verify that you indeed do have a valid image. This is done by checking the MD5 checksum. If the checksum does not pass you have a corrupt image and will need to download again.
3. The next step is to get CD-burning software. There are many free options on the Internet.

The next steps are to install Ubuntu (Linux) on onto a hard drive from the CD. In order to do this, your Linux system must have a CD/DVD reader and a hard drive connected to the 2nd Generation platform. Both of these must be recognized by the BIOS. When a system is powered up with this CD present (assuming the BIOS settings are correct) the system will boot from this CD instead of the operating system present on any connected hard drives. If the user installing the Ubuntu operating system so chooses, the entire hard drive will be erased, making room for Ubuntu.

The next step is to use the Ubuntu CD to install the OS with desired computer name, user-id (ivi) and password on to a new (blank) hard drive. In my case I used 250 GB drive but smaller drives will work also. For a lesser number of tests and apps 32GB will be sufficient. For speedy installation one can use a SSD (flash) hard drive. Initially, we do not need to be connected to the network however in some cases where you are behind a “firewall” and require proxy settings connection issues can be resolved.

For people with proxy/firewall settings, one needs to use either system tools (System → Preferences → Network proxy) or the browser internet setting to set the proxy after the installation. One can check with their IT administrator or from a working system to figure out the settings. Once the settings are correct, one should be able to access

Internet content with Firefox, git, apt-get.   One may need also to put the following lines in their *.bashrc* file:

```
http_proxy=http://proxy…..: [port]
ftp_proxy=http://proxy…..: [port]
https_proxy=http://proxy…..: [port]
```

After the initial installation, from (System → Administration → Update Manager) make sure you have an up-to-date system. And update your desktop:

```
>>>sudo –i
>>>http_proxy=http://proxy…..: [port]
>>>  apt-get clean
>>>  apt-get check
>>>  apt-get update
```

Then

```
>>>sudo –i
>>>  apt-get  install  x11proto-xf86dri-dev  libxmu-devlibxi-devlibxmu-
headers     libxt-devlibsm-devlibice-dev     libexpat1-dev     libxext-
devlibxdamage-dev  x11proto-damage-dev  libxfixes-dev  x11proto-fixes-dev
x11proto-gl-dev  xserver-xorg-devlibpciaccess-dev  x11proto-xinerama-dev
x11proto-xext-dev   x11proto-video-dev   x11proto-render-dev   x11proto-
randr-dev  x11proto-fonts-dev  libxkbfile-dev  libpixman-1-dev  x11proto-
dri2-dev  libx11-dev  libxcb1-dev  xtrans-dev  x11proto-kb-dev  x11proto-
input-dev libxdmcp-devlibxau-dev x11proto-core-dev libtalloc-devxutils-
dev    libpthread-stubs0-dev    g++    libpthread-stubs0    libglew1.5-dev
gitlibxt-devlibxmu-devlibxi-devllvmautoconflibtool  flex  bison  openssh-
server aptitude x11proto* mesa-utils
>>>  apt-get  update  ;   apt-get  install  openssh-server  openssh-client
yasm ; apt-get update ;
>>> /etc/init.d/ssh stop;  /etc/init.d/ssh start
```

Make sure that all updates are completed. One may have to rerun the “Update Manager” and “apt-get update” multiple time. For obtaining the correct version of Mplayer one needs subversion, to compile it yasm and build-essential are required. Since the defaultgcc-4.7 fails to build it, install gcc-4.6. To conveniently build a debian package out of it I recommend checkinstall:

```
>>>sudo apt-get install yasm build-essential subversion checkinstall
>>>sudo apt-get install gcc-4.6
```

Then the libraries used by Mplayer can be installed by simply grabbing the build-deps of it.

```
>>>sudo apt-get build-depMplayer
```

Next step is to obtain the Mplayer source and the vaapi patches. The original site (splitted-desktop.com) does not longer host the patches for Mplayer. Download Andreas Grois copy shared at http://ubuntuone.com/36afxAZcJfNAyEQA273UgS

After that we need to go to to the directory that we downloaded this file to and untar it. Enter thedirectory extracted from the archive

```
>>tar -xf Mplayer-vaapi.tar.gz
>>>cd Mplayer-vaapi-20110127
```

Run the script to obtain the Mplayer source code and to apply the vaapi patches. You don't need to build Mplayer, because for now it would be built without vaapi, therefore the patch parameter is given to the script, so it stops after patching:

```
>>> ./checkout-patch-build.sh patch
```

Now enter the Mplayer source folder:

```
>>>cd Mplayer-vaapi/
```

The configure file checks for vaapi by calling a function that does not work on Ubuntu 12.04 and 12.10. To correct this, either manually editthe file, or use Andreas Grois patch which is available fromhttp://ubuntuone.com/2z4wBZ6gCwcvaUxJzN1hCv. Then we need to save the patch in the Mplayer-vaapi folder and name it to test.patch. After that simply apply it to configure by running:

```
>>>patch configure test.patch
```

Next, run configure with any options that need to be configure. User also must have to make that they uses gcc-4.6, for the build fails with gcc-4.7 and don't forget to enable vaapi and disable vdpau:

```
>>>CC="gcc-4.6" ./configure --prefix=/usr --confdir=/etc/Mplayer \--
enable-vaapi --disable-vdpau.
```

Check the output of configures and if it’s complete, simply run make and wait for the compile to finish:

```
>>>make
```

Sometimes it will give an "illegal instruction" error which might happen to user that use virtual machine. User can try "makeclean" to delete the build and add the --enable-runtime-cpu detection option to configure before building again. And to see if it is working with vaapi, play some file with vaapi output.

```
>>>./Mplayer -vovaapi -vavaapi [some video file]
```

If it works, user can proceed with command:

```
>>>sudo make install
```

The files for Mplayer with vaapi should already be at the appropriate folders. The whereis command will give the location where files named Mplayer reside. Normally, the program will be located at /usr/bin/Mplayer. There is the folder where to place the system wide configuration file /etc/Mplayer and the folder /usr/bin/X11 is actually just a link to /usr/bin, so every file in /usr/bin is also displayed as being in /usr/bin/X11.

However, if we now install packages that depend on Mplayer, they might overwrite our installation by the stock Ubuntu Mplayer, because officially the package Mplayer is not installed (since checkinstall failed). So we need to edit the version number to be actually a number and not vaapi.

So when it asks:

```
>>>Enter a number to change any of them or press ENTER to continue:
```

Choose 3 and give it a number, example like 20110127 to remind us that it's the svn-version from then. We can easily see if vaapi is used when playing some h.264 video file. Close to the end of Mplayers output it should give information about the video output driver used. For a H.264 file that can be played using vaapi, the output should be something like (of course with the resolution of the video file):

```
>>>VO:  [vaapi]  1920x816  =>  1920x816  H.264  VA-API  Acceleration
[VD_FFMPEG]XVMC-accelerated MPEG-2.
```

Last but not least to confirm that we are decoding the video file using hardware acceleration, we can type this command:

```
>>>vainfo
LibVa: VA-API version 0.32.0
LibVa: va_getDriverName() returns 0
LibVa: Trying to open /usr/lib/i386-linux-gnu/dri/i965_drv_video.so
LibVa: va_openDriver() returns 0
vainfo: VA-API version: 0.32 (LibVa 1.0.15)
vainfo: Driver version: Intel i965 driver - 1.0.17
vainfo: Supported profile and entrypoints
      VAProfileMPEG2Simple            :    VAEntrypointVLD
      VAProfileMPEG2Main              :    VAEntrypointVLD
      VAProfileH264Baseline           :    VAEntrypointVLD
      VAProfileH264Baseline           :    VAEntrypointEncSlice
      VAProfileH264Main               :    VAEntrypointVLD
      VAProfileH264Main               :    VAEntrypointEncSlice
      VAProfileH264High               :    VAEntrypointVLD
      VAProfileH264High               :    VAEntrypointEncSlice
      VAProfileVC1Simple              :    VAEntrypointVLD
      VAProfileVC1Main                :    VAEntrypointVLD
      VAProfileVC1Advanced            :    VAEntrypointVLD
```

Development of VideoWall Bench Script

For this project, the development of the coding was been developed into two phase. Phase one was full screen methodology and second phase was video wall methodology. Below is the diagram:

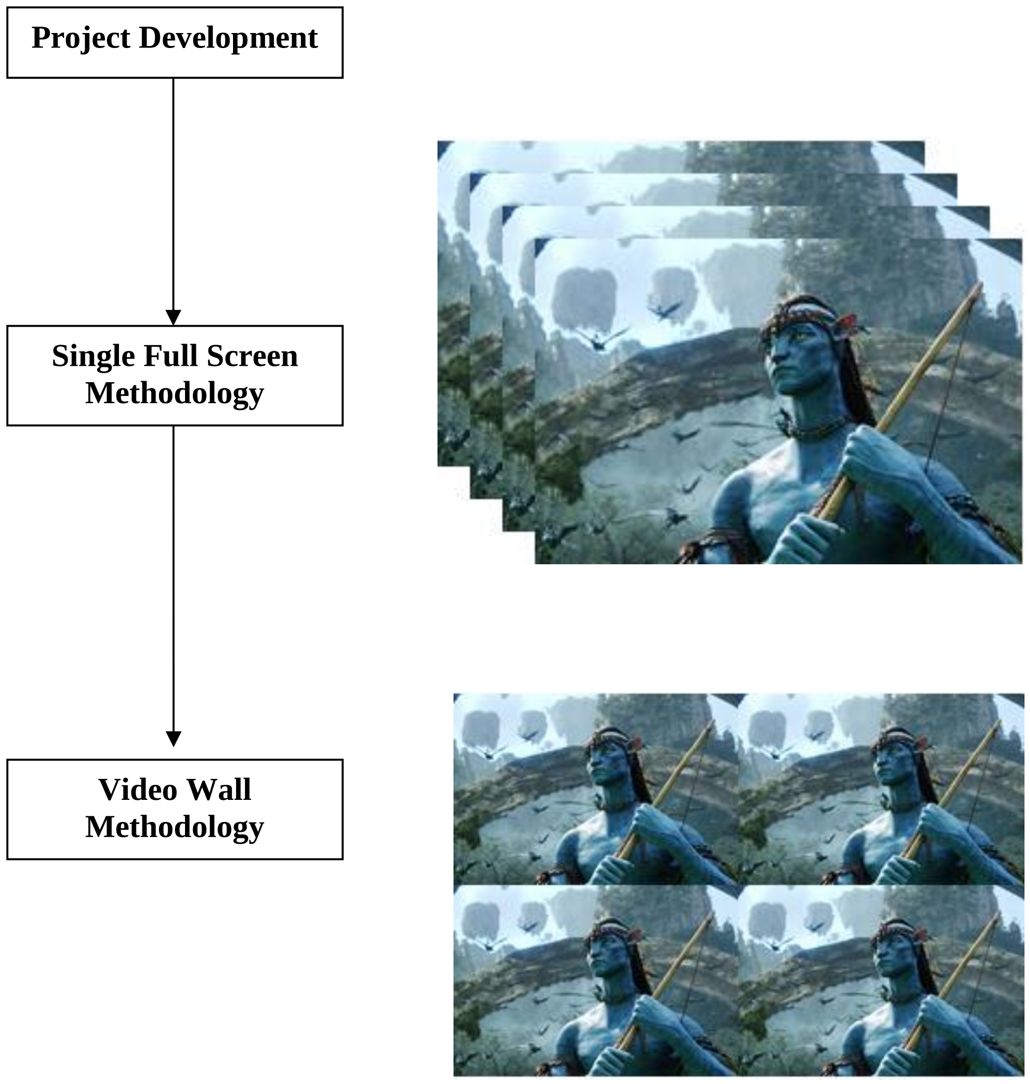


Figure 10: Video Decoding Methodology

Single full screen methodology is a first phase of VideoWall Bench. Using this methodology, the applications will execute Mplayer VA-API command in the Perl script to decode video file into multiple stream. All the videos are being play concurrently but this technique overlaps other videos make other streams not visible to tester.

Phase 4: Project Testing

I also have to make sure that the code execute correctly and all the result score being compiled properly by the program. This testing phase also been executed in many type of operating system to make sure the software can be used in main Linux operating system. In this phase, the script was tested for their functions. VideoWall Bench will be tested with all parameter to make sure it works fine and no issue on the code while the script was been executed. All of the code in VideoWall Bench will be integrated into a complete system during integration phase and test to check if all modules/units coordinate with each other and the system as a whole behaves as per the specifications. This script also has to be tested on other Linux operating system such as Meego and Ubuntu.

In this stage, all scripts are tested for several kind of error testing as descried in **Table 6**. The testing process is also taking place in configuration file to ensure that script of the VideoWall Bench is properly connected. As for "Iter_one_memcpu.pl" and "One_Decode_wall_p.pl", all the data input by the user in this page must be able to be stored inside its respective database. Meanwhile, for "Result.txt", it should be able to retrieve all the data from its connected database without returning any wrong information or empty data. The testing phase is also important to ensure that the system can successfully display all the result that has been generated. Each script and its respective code must be interrelated with each other and hence, provide the correct calculation in order to obtain the maximum number of video decoding. **Table 7** meanwhile shows the type of testing being performed to check the system as a whole.

Test No	Type of Error Handling Test	Purpose	Status
1	Calculation and Computation Error	To make sure that the system can correctly perform all mathematical operations and return the correct values.	Passed
2	Invalid Data Type Error Handling	To make sure that the system do not proceed with execution and notify user when the input data type is wrong. For example, if the supposed data to be entered is numbers, and user entered characters, the system will notify user that the input data is wrong and the operation will break.	Passed
3	Empty input error handling	The system will not proceed with execution whenever it required the input data, if there is no input being entered.	Passed
4	Error notification message	For all error that occurs, the system should prompted user to notify them.	Passed

Table 6: Error Handling Testing

Test No	Type of System Testing	Purpose	Status
1	Black Box testing	To test for system requirement and functionality without considering the internal architecture of the system.	Passed
2	White Box Testing	To test for internal functionality of the system. This testing included the coding of the system.	Passed
3	Unit Testing	To test for the functionality of each separated system page.	Passed
4	Acceptance Testing	To ensure the system is completed and performed as requested by user.	Passed
5	Functional Testing	To ensure the system able to performed all its intended functionality.	Passed

Table 7: Type of System Testing performed

Phase 5: Project Completion

This phase of the waterfall with agile model is virtually a never-ending phase. Generally, problems with the system developed (which are not found during the development life cycle) come up after its practical use starts, so the issues related to the system are solved after deployment of the system. In this phase, completion means the system can be used for experimental studies that will be discuss further in result and discussion part. During this phase also I need to make documentation and records all the steps how to install the programs. This step is very important for other people to use VideoWall Bench script.

Phase 6: Experimental Studies

Experimental studies are last part for VideoWall Bench. Upon the completion of the system, the author would like to make several testing that will answer the following questions.

Is there any performance difference in terms of processor, fps and memory utilization between H.264 and MP4

Number of maximum videos that can be display at one time using Intel i5 2500k Intel HD 3000

Is there any performance difference in terms of processor, fps and memory utilization for changing processor speed in video decoding process

Is there any performance difference in terms of processor, fps and memory utilization for changing screen size of video display

All the results and findings from the testing will be put on result and discussion part. Generally, user can change several parameters in the VideoWall Bench

- Frames per second (FPS) (30 fps, 60 fps)
- Screen Size (800x600, 1280x720, 1920x1080)
- Video File

3.2 Software Involved

1. **MplayerVaapi**

   It is a free and open source media player. The program is available for all major operating systems, including Linux and other Unix-like systems**.** MplayerVaapi version is specially made to integrate the LibVa library with Mplayer media player.

2. **LibVa**

   LibVa is an open source software for video acceleration API. It enables and provides access to graphics hardware (GPU) acceleration for video processing. VA API is targeted at the X Window System on Unix-based operating systems. Accelerated processing includes video decoding, video encoding, sub picture blending and rendering. The specification was originally designed by Intel for its GMA (Graphics Media Accelerator) series of GPU hardware. However, the API is not limited to GPUs or Intel specific hardware, as other hardware and manufacturers can also freely use this API for hardware accelerated video decoding.

3. **Latest Intel Graphic Driver**

   It is very important in this project to use latest Intel Graphic Driver which is supported the usage of LibVa. Without supported graphic driver, video decoding in VideoWall Bench will not able to use hardware acceleration thus maximizing the processor utilization.

3.3 Hardware Involved

1. **Intel Sandy Bridge i5-2500k 3.30 GHz**

2. **250GB Hitachi Hard Disk Model HTS54**

3. **2GB Kingston 2GB PC1333 D3 x 2 unit = 4GB RAM**

4. **BenQ G220-HD 22 Inch Monitor**

3.4 Gantt Chart

Final Year Project Part I

Detail Week	1	2	3	4	5	6	7	8	9	10	11	12
Selection of Project Topic & Supervisor	🟦	🟦										
Submission of Proposal to research cluster			🟦									
Submission of Extended Proposal			🟦	🟦	🟦	🟦						
Research Class		⬛	⬛	⬛	⬛	⬛	⬛	⬛				
Conduct the survey									🟦	🟦	🟦	
Submission of Viva: Proposal defense and Progress Evaluation									🟦	🟦		
Submission of Interim Report									🟦	🟦	🟦	🟦

Final Year Project Part II

Detail Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Programming Research	🟦													
Prototype Development		🟦	🟦	🟦	🟦	🟦								
Submission of Progress Report I		🟦												
Submission of Progress Report II							🟦							
Pre-SEDEX										🟦				
Submission of Final Report Draft											🟦			
Oral Presentation													🟦	
Submission of Final Dissertation														🟦

CHAPTER 4

RESULT AND DISCUSSION

4.1 Results from consulting an expert on multimedia benchmarking

A number of experts were needed to give guideline that could help to design the proposed VideoWall Bench script. Therefore during the development of VideoWall Bench, They are two experts that have been approach through email and Facebook to give some comments and critics about the VideoWall Bench. One of the experts was Mr. Muhammad Syazwan Mazlan, System Performance Engineer from Performance Measurement Analysis at Intel Microelectronics, Penang. He graduated from Universiti Malaysia Perlis with a Bachelor's Degree Program in Computer Engineering. Regarding the design of the software, he gave comments on the following issues to be considered:

- **Linux Based System**

  Mr Syazwan likes the idea of having a video decoding benchmarking system on Linux. Using Linux it is more flexible and Linux has been widely used in performance benchmarking such as EEMBC performance test.

- **Application been developed using Perl Language**

  Perl language is very popular in the Unix community because it has a rich and powerful feature set, but is still easy to use. Perl borrows heavily from other languages such as C and awk. Perl has been ported to many non-Unix environments, including DOS, OS/2, Macintosh, VMS, and Windows NT. Perl program can run with little or no modification on many different platforms is another reason for its popularity.

- **Multiple Video Decoding**

  Nowadays processors have more powerful capabilities especially in terms of multimedia compared to previous years. Therefore it is quite invalids for current processor to used previous video decoding benchmarking system such as Media Bench and HD Benchmark. Furthermore most of the video benchmarking systems are using SIMD optimization which decode video using software acceleration, thus unable the processor to decode more than 2 videos at one time.

- **Optimize for Hardware Accelerated Video Decoding**

  Hardware accelerated video decoding enable the systems to decode multiple video at one time. However Mr. Syazwan also suggests that I should make an option for users to benchmark using software acceleration or hardware acceleration. By having this option, users can compare performance difference between decoding using software acceleration and hardware acceleration.

- **Performance Measurement.**

  VideoWall Bench will measure video decoding performance in terms of processor utilization, memory utilization, frame rate per second and time latency of the video been play. According to Mr. Syazwan all those variable are valid to be measured. Besides using top command in Linux, he suggests me to measure processor utilization using time command. Using time command it will capture real, user and system time for each video that been decode. Below are the formulas that have been suggested by Mr. Syazwan.

CPU Utilization= (User Time + System Time)/ Real Time

Second expert that have been interviewed through Facebook was Mr. Dennis E. Mungai. He is a Founder of Brainiarc Eight, a Facebook community page about computer hardware designs and software reviews. He graduated from Kenyatta University at Kenya in BSc Software Engineering. He highlights some important matters about video decoding concept that need to be considered for the development of the VideoWall Bench.

- **Video decoding depends on a number of factors, notably:**

  First is the codec type used to compress the video streams. For Intel HD 3000 GPU, it supports H.264, VC-1 and MPEG-2 Hardware Decoding pipelines enabled. However, depending on the codec used and its' implementation level, not all portions of it can be offloaded to the GPU for processing. Therefore, uneven GPU shader usage is to be expected. Again, not all H.264 Video Decode parts are offloaded to the GPU. This driver VA-API enables hardware accelerated video decode/encode at various entry-points (VLD, IDCT, Motion Compensation etc.) for the prevailing coding standards today (MPEG-2, MPEG-4 ASP/H.263, MPEG-4 AVC/H.264, and VC-1/VMW3). It provides an interface to fully expose the video decode capabilities in today's GPUs.

- **The Video Container encapsulating the Video Stream:**

  Video container is what we inaccurately call the Video Format, e.g. MP4, AVI, Matroska, etc. Depending on the Media Container Splitters used to split the container into video and audio streams needed by a media player for decoding and playback, CPU and GPU usage spikes may be different.

- **The Media Player in use:**

  Some media players, notably VLC, do not implement the full H.264 decoding standard, and may fail in Full Hardware Decoding especially with H.264 content encoded outside the official Blu-ray standards. The noticeable implication is that the media decoder will perform software fallback, incurring massive load on the Video Driver especially where Post-processing is enabled. Secondly, transferring bit stream data between GPU and CPU cores and re-shuffling it between pipelines is also memory-expensive, an effect you'll notice with Integrated Graphics since IGPs share main system memory with their buses.

- **The Video Resolution, Aspect Ratio and Bitrates:**

  CPU and GPU usage is directly proportional to all these three factors. Concerning frame rates, these are just statistics. With Vertical Sync Enabled (VSYNC, usually enforced by the Intel Linux Driver by Default), we will not get any more than 60fps. Video rendering APIs use the same stack rules as 2D image rendering. A video is just an array of continuous picture frames encoded with a gap distance (GOP) by a codec, and in most cases, frame rates are explicitly encoded into the video bit stream's Meta data.

- **GPU Usage in video decoding process**

  Concerning GPU usage, GPUs are highly paralleled computing units, and it would take massive data sets to fully utilize a GPU (100% Usage), such as intensive 3D gaming or 1080p Video Post-processing.

4.2 Discussion on the Work Progress

- **First File – run_all_test.pl**

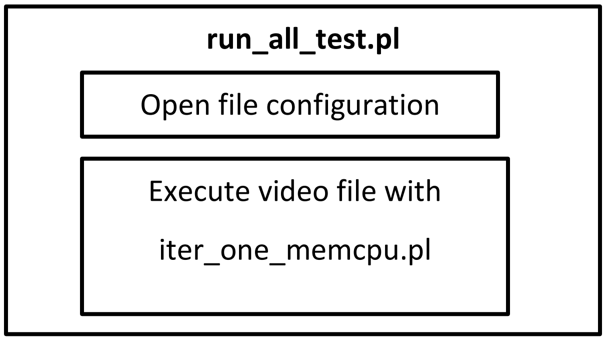


Figure 11: Components of run_all_test.pl

This file script is to use to run the VideoWall Bench with configuration file. In this script it will extract the video file name from configuration file into array test_v[0], test_v[1], test_v[+1] to be executed with iter_one_memcpu.pl script. After the script finishes its execution, it will print the result into Result.Txt.

Code Snippets

```
system(" echo   > ./Results.TXT");

while($line=<FILE_IN>){
#print "$line \n " ;
$line=~s/\n$//g;

#print "$line \n " ;
$test2=$line;

#print "$test2 \n " ;
@test_v=split(" ",$test2);

#print "@$test_v \n " ;'
$file_name=@test_v[0]."/".@test_v[1];
if(-e$file_name){

$cmd_l="        perl                         iter_one_memcpu.pl
".@test_v[0]."/".@test_v[1]."  ".@test_v[2];
$cmd_l=~s/\n/ /g;

#print "     @test_v    \n";
```

- **Second File – iter_one_memcpu.pl**

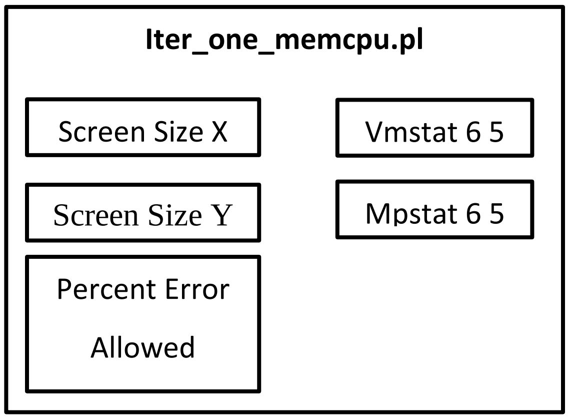


Figure 12: Components of Iter_one_memcpu.pl

In this part, user can adjust the screen size x and screen size y. Example such as if the user set up the screen size x = 1366 and screen size y = 768, it will be divided according to the number of file that have been set in the configuration file. This section also allow user to change the percent error allowed during video decoding. Default value for it is 2.5. The percent error allowed field also will determine the number of maximum video that the system able to decode. Let say if the percent error allowed that has been set is 2.5 and during decoding of 40 videos the percent error accumulated is 3. Therefore the VideoWall Bench will decode 39 videos or less until the percent error allowed is less than 2.5. Depending on what system that run VideoWall Bench, each of it will respond according to their decoding capabilities. In order to capture CPU utilization and memory utilization, Vmstat and Mpstat are been executed in this script during the video decoding process. All the information that been capture will be copy into out_Vmstat.txt and out_Mpstat.txt.

Code Snippets

```
$size_sx=1800;
$size_sy=1000;
$PERCENT_ERROR_ALLOWED=2.5;
# set to 2.5 percent deviation

$HELP_LINE=" perl perl_file.pl  video file upper_bound    \n";
$num_arg=$#ARGV;
if($num_arg!= 1 ){print"usage $num_arg  :  $HELP_LINE ";exit;}

$file_n=shift(@ARGV);
$n_max=shift(@ARGV);
$n_max=$n_max+1;
$n_min=1;
system(" killallmplayer ");
$not_done= 0;
while($not_done< 1){

$n_test=int(($n_min+$n_max)*0.5);
system("Vmstat 6 5 > out_Vmstats.txt & ");
system("Mpstat 6 5 > out_Mpstats.txt & ");
system("\/usr\/bin\/perl    .\/one_Decode_wall_p.pl    $file_n
$size_sx  $size_sy  ".$n_test."  HW  \|  grep  \"Results  :\"  2\>
\/dev\/null \> dummy11.txt");
my$Results=`cat dummy11.txt`;
my$temp=$Results;
$Results=~s/\n//g;
$temp=~s/\n//g;
$temp=~s/.*missed frames%: *//g;
$temp=~s/time fluctuation%/ /g;
$temp=~s/:/ /g;
$temp=~s/  */ /g;
($frames_error,$time_error)=split(/ /,$temp);
$max_error=$frames_error;
if($max_error<$time_error){$max_error=$time_error;}

if($PERCENT_ERROR_ALLOWED>$max_error){
$n_min=$n_test;
$last_r=$Results;
$vm_out=`cat out_Vmstats.txt | tail -1 | cut  -c 13-21 `;
$vm_out=~s/\n//g;
$mp_out=`cat out_Mpstats.txt |tail -1 `;
$mp_out=~s/.* //g;
$mp_out=~s/\n//g;

}
else{
$n_max=$n_test;
}



}
```

- **Third File – One_Decode_wall_p.pl**

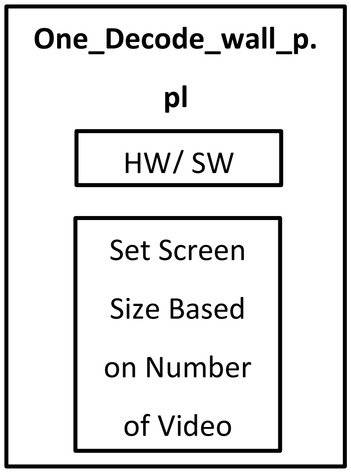


Figure 13: Components of One_Decode_wall_p.pl

This file script is use to set the Mplayer VA-API to decode the video using hardware acceleration. In this script user can edit the Mplayer whether to play it using hardware acceleration or software acceleration. The implementation of hardware acceleration must be at the video output (-vo) and video input (-va) at the code. The Perl code which defines the Mplayer configuration looks like this:

```
$HW_=shift(@ARGV);
$_=$HW_;
if(/SW/){
#  $HW_=" -vo direct3d "
$HW_="  "
}
else{
$HW_="-vovaapi -vavaapi  ";
}
```

Take an example of screen size x= 1800 and screen size y = 1000. It is impossible for the 22 inch screen monitor to display all 40 videos that have screen size 1800x1000 for each video. Therefore in order to make sure all the video decode by Mplayer is been display at the screen, an algorithm of screen display is been implemented in this code. This algorithm will determine the screen size of the video based on the number of video file that needed to be playin the configuration file. The Perl code which defines the screen size algorithm looks like this:

```
if($total_n< 65){$nx= 8;$ny= 8;}
if($total_n< 50){$nx= 7;$ny= 7;}
if($total_n< 37){$nx= 6;$ny= 6;}
if($total_n< 26){$nx= 5;$ny= 5;}
if($total_n< 17){$nx= 4;$ny= 4;}
if($total_n< 10){$nx= 3;$ny= 3;}
if($total_n< 5){$nx= 2;$ny= 2;}
if($total_n< 2){$nx= 1;$ny= 1;}
```

Using formula below it will calculate screen size of x and y based on number of video files that need to be decode

```
$size_x=int($screen_size_x/$nx);
$size_y=int($screen_size_y/$ny);
```

After the size of x and y have been determine by the formula, this script will execute Shell Script (run_in_back.sh) which contain Mplayer command line. This command will generate two logs file which are .log and .logt. Below is the perl code which execute run_in_back.sh

```
if($total_n> 1){

print" .\/run_in_back.sh $file_n    $size_x $size_y $x:$y   1>
$file_n"."_"."$n$N.log 2> $file_n"."_"."$n$N.logt \n";
system(" .\/run_in_back.sh $file_n   $size_x $size_y $x:$y   1>
$file_n"."_"."$n$N.log 2> $file_n"."_"."$n$N.logt& ");

}
elsif($total_n== 1){
print" .\/run_in_back.sh $file_n    $size_x $size_y $x:$y   1>
$file_n"."_"."$n$N.log 2> $file_n"."_"."$n$N.logt  \n";
system(" .\/run_in_back.sh $file_n   $size_x $size_y $x:$y   1>
$file_n"."_"."$n$N.log 2> $file_n"."_"."$n$N.logt  ");

}
```

During the execution of the run_in_back.sh, two dummy file which are dummy1.txt and dummy2.txt will be created. The purposes of the creation of these dummy files are to include information regarding the calculation of the total frame of video decoding, decoded frame and second calculation. All of this data will be used to calculate the time latency of video decoding. Below is the Perl code:

```
$x=0;
$y=0;
$N=$total_n;
@frames_decoded=(0);
@frames_total=(0);
@elapse_secs=(0);
$n=0;
for($j= 0;$j<$ny;$j++){
for($i= 0;$i<$nx;$i++){
$file_log="$file_n"."_"."$n$N.log";
if($n<$N){
$logfile="$file_n"."_"."$n$N.log";
$logfilet="$file_n"."_"."$n$N.logt";
system(" tail $logfile | grep V: | sed \'\/V\:\/p\' | sed
\'s\/.\*V\:\/\/g\' | tail \-1 | sed \'s\/\^ *\/\/g\' | sed
\'s\/ *\/ \/g\' | sed \'s\/ \/T\/\' | sed \'s\/.*T\/\/g\' |
sed \'s\/ .*\$\/\/g' | sed \'s\/\\\/\/ \/g\' > dummy2.txt ");
system(" cat $logfilet | grep elapsed | sed \'s\/.*system
*\/\/g\' | sed \'s\/elapsed.*\/\/g\' | sed \'s\/:\/ \/g\' >
dummy1.txt");
my$elap_time=`cat dummy1.txt`;
my$frames_dec_total=`cat dummy2.txt`;
($frames_decoded[$n],$frames_total[$n])=split(/
/,$frames_dec_total);
($min,$sec)=split(/ /,$elap_time);
$sec_cal=$min* 60.0 +$sec;
$elapse_secs[$n]=$sec_cal;
print" $sec_cal $frames_decoded[$n] $frames_total[$n] \n";
$n=$n+1 ;
}

}}
```

Calculation of total frame, decoded frame, total FPS, missed frames and time fluctuation need a special formula which are:

1. Total FPS = Number of Video x (Minimum Decode F / Maximum Time)
2. Time Fluctuation = Decode Time Error x100.00
3. Missed Framed = Missed Frame x 100.00

All of the above formulas are been derived from the below if else statements:

```
if($elapse_secs[$i]<$min_time){$min_time=$elapse_secs[$i];}
if($elapse_secs[$i]>$max_time){$max_time=$elapse_secs[$i];}


if($frames_total[$i]<$min_total_f){$min_total_f=$frames_total[$
i];}
if($frames_total[$i]>$max_total_f){$max_total_f=$frames_total[$
i];}

if($frames_decoded[$i]<$min_decodef){$min_decodef=$frames_decod
ed[$i];}
if($frames_decoded[$i]>$max_decodef){$max_decodef=$frames_decod
ed[$i];}
}

$decode_total_error= 1.0;
if($min_total_f>
0){$decode_total_error=($max_total_f/$min_total_f)-1.0;}
$decode_decode_error= 1.0;
if($min_decodef>
0){$decode_decode_error=($max_decodef/$min_decodef)-1.0;}
$decode_time_error= 1.0;
if($min_time> 0){$decode_time_error=($max_time/$min_time)-1.0;}
$missed_frms=1.0;
if($max_total_f> 0){
$missed_frms=($max_total_f-$min_decodef)/$max_total_f;}


$max_total_f=~s/\n//g;
$min_decodef=~s/\n//g;
$decode_time_error=~s/\n//g;

$t_fps=$N*$min_decodef/$max_time;
$decode_time_error=$decode_time_error*100.00;
$missed_frms=$missed_frms*100.00;

print"  Results  :  Total  Frms  $max_total_f,  Decoded  Frms
$min_decodef,  Total  FPS  $t_fps,  missed  frames%:  $missed_frms
time fluctuation% : $decode_time_error \n";
```

- **Fourth File – run_in_back.sh**

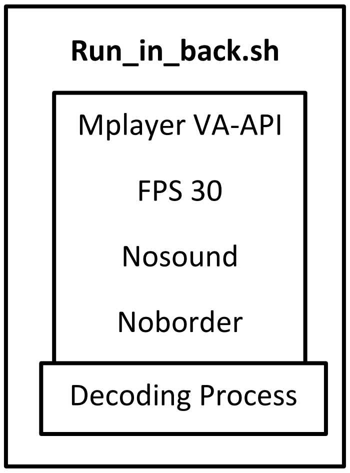


Figure 14: Components of run_in_back.sh

This Shell Script was been created to execute Mplayer VA-API from its destination folder. It is very important for user to put the correct path of Mplayer so it will be able to execute successfully. User also may try to copy the code written in this shell script and paste it at the terminal to see whether Mplayer is working and decode video successfully. The Mplayer command line in this Shell Script looks like this:

```
# /bin/bash
#echo $DISPLAY
#DISPLAY=:0.0
#export DISPLAY

time/usr/bin/mplayer\
-vavaapi-vovaapi-nosound\
-fps30-noborder -x $2-y $3-geometry $4$1-loop 1\
</dev/null
```

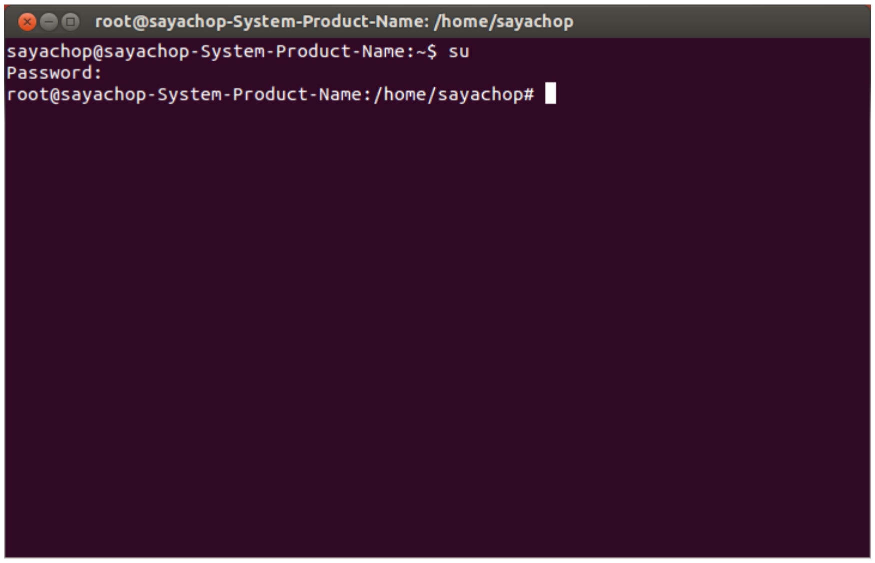
Parameter	Function
time	Measure time in Linux to calculate **CPU utilization**
/usr/bin/Mplayer	Call Mplayer application in Install directory
-vavaapi –vovaapi	Use Vaapi technology to decode using hardware acceleration
-no sound	Disable sound in video file to minimize the CPU utilization during video decoding
-fps 66	Force selected H264 video to be decoded to 66 fps

Table 8: Parameter in Mplayer

4.3 System Operation and VideoWall Bench User Guide

1) In order for the users to run VideoWall Bench successfully, users must install ALL the dependencies, installation package, and Mplayer VA-API completely. Any skip of steps will lead to the failure of VideoWall Bench execution.

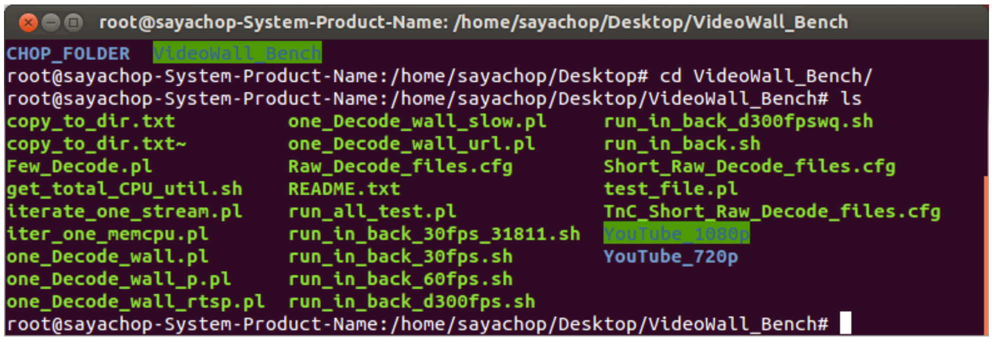
2) After completely installed all the things needed, user can copy VideoWall_Bench folder and place it at the Desktop. Then open your terminal and type su to become Superuser. It is a special user account used for system administration. It allows users to become the root of the system which permits any execution of files in the operating system. Failing to do so will make Mpstat and Vmstat command in the Perl script fail to execute.



```
root@sayachop-System-Product-Name: /home/sayachop
sayachop@sayachop-System-Product-Name:~$ su
Password:
root@sayachop-System-Product-Name:/home/sayachop#
```

Figure 15: Command to become a Superuser

3) Then enter to the directory of the VideoWall_Bench whichmine is situated at Desktop Folder (/home/sayachop/Dekstop/VideoWall_Bench).User also must make sure thatthe folder of VideoWall_Bench has been root permission approved. To allow root permission type the command “chmod 777 VideoWall_Bench” at the terminal. If the command successful,VideoWall_Bench folder will become into green colour like a picture below



```
root@sayachop-System-Product-Name: /home/sayachop/Desktop/VideoWall_Bench
CHOP_FOLDER  VideoWall_Bench
root@sayachop-System-Product-Name:/home/sayachop/Desktop# cd VideoWall_Bench/
root@sayachop-System-Product-Name:/home/sayachop/Desktop/VideoWall_Bench# ls
copy_to_dir.txt          one_Decode_wall_slow.pl     run_in_back_d300fpswq.sh
copy_to_dir.txt~         one_Decode_wall_url.pl      run_in_back.sh
Few_Decode.pl            Raw_Decode_files.cfg        Short_Raw_Decode_files.cfg
get_total_CPU_util.sh    README.txt                  test_file.pl
iterate_one_stream.pl    run_all_test.pl             TnC_Short_Raw_Decode_files.cfg
iter_one_memcpu.pl       run_in_back_30fps_31811.sh  YouTube_1080p
one_Decode_wall.pl       run_in_back_30fps.sh        YouTube_720p
one_Decode_wall_p.pl     run_in_back_60fps.sh
one_Decode_wall_rtsp.pl  run_in_back_d300fps.sh
root@sayachop-System-Product-Name:/home/sayachop/Desktop/VideoWall_Bench#
```

Figure 16: VideoWall_Bench Folder

4) There are two types of video files that been provided in VideoWall Bench which are YouTube_720p and YouTube1080p. Both folders contain six videos that have same content but different resolution.

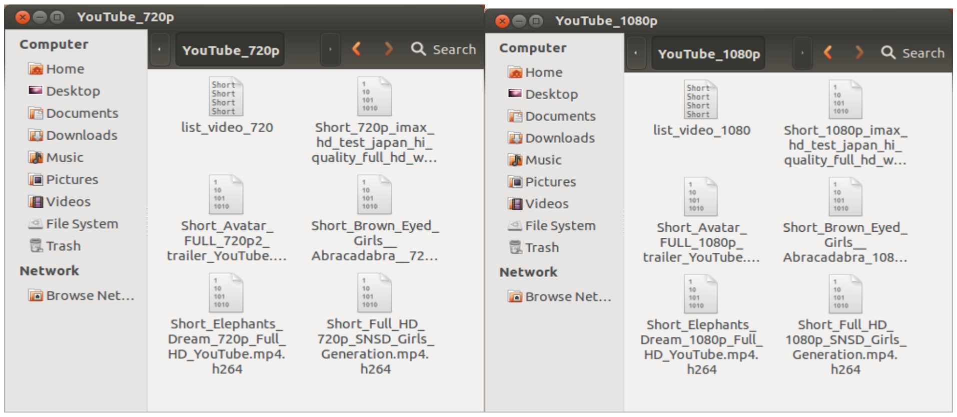
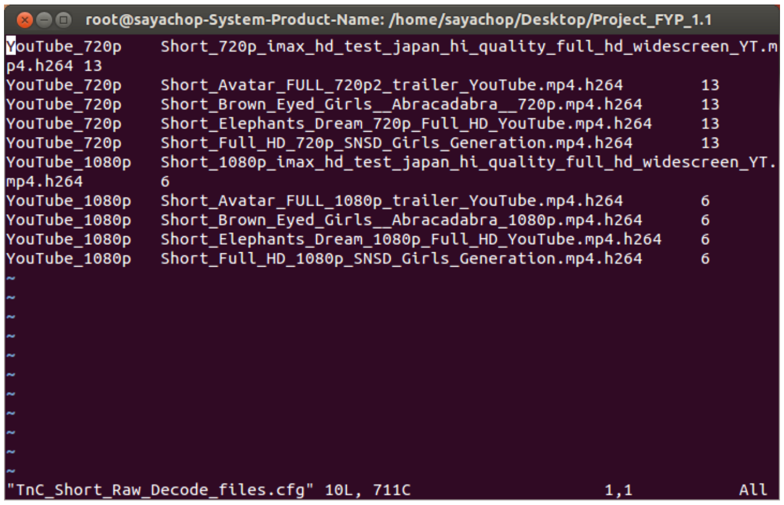

YouTube_720p
Computer
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
File System
Trash
Network
Browse Net...
YouTube_720p
Search
list_video_720
Short_720p_imax_hd_test_japan_hi_quality_full_hd_w...
Short_Avatar_FULL_720p2_trailer_YouTube....
Short_Brown_Eyed_Girls__Abracadabra__72...
Short_Elephants_Dream_720p_Full_HD_YouTube.mp4.h264
Short_Full_HD_720p_SNSD_Girls_Generation.mp4.h264
YouTube_1080p
Computer
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
File System
Trash
Network
Browse Net...
YouTube_1080p
Search
list_video_1080
Short_1080p_imax_hd_test_japan_hi_quality_full_hd_w...
Short_Avatar_FULL_1080p_trailer_YouTube....
Short_Brown_Eyed_Girls__Abracadabra_108...
Short_Elephants_Dream_1080p_Full_HD_YouTube.mp4.h264
Short_Full_HD_1080p_SNSD_Girls_Generation.mp4.h264

Figure 17: Video Files in YouTube720p and YouTube1080p

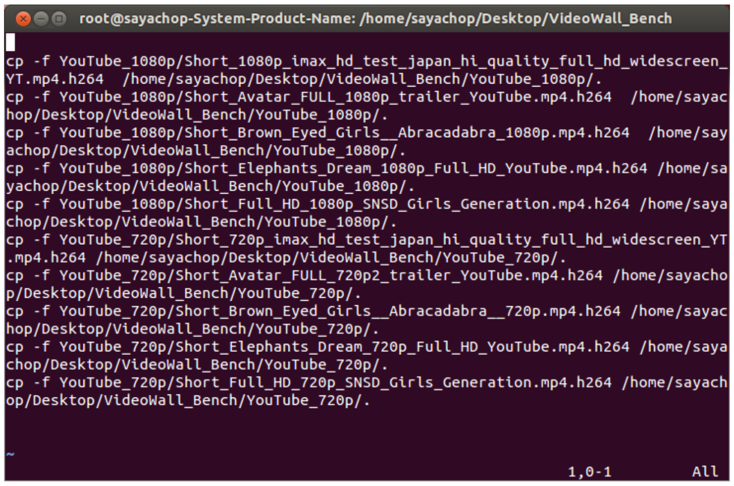
5) Next step is to check the configuration file (TnC_Short_Raw_Decode_files.cfg). In this file user can view the name of folder, name of video files and number of video files that want to decode simultaneously. User also can change to any type of video, provided that the video folder and the video file are stored in VideoWall_Bench directory.



```
root@sayachop-System-Product-Name: /home/sayachop/Desktop/Project_FYP_1.1
YouTube_720p    Short_720p_imax_hd_test_japan_hi_quality_full_hd_widescreen_YT.m
p4.h264 13
YouTube_720p    Short_Avatar_FULL_720p2_trailer_YouTube.mp4.h264         13
YouTube_720p    Short_Brown_Eyed_Girls__Abracadabra__720p.mp4.h264       13
YouTube_720p    Short_Elephants_Dream_720p_Full_HD_YouTube.mp4.h264      13
YouTube_720p    Short_Full_HD_720p_SNSD_Girls_Generation.mp4.h264        13
YouTube_1080p   Short_1080p_imax_hd_test_japan_hi_quality_full_hd_widescreen_YT.
mp4.h264        6
YouTube_1080p   Short_Avatar_FULL_1080p_trailer_YouTube.mp4.h264        6
YouTube_1080p   Short_Brown_Eyed_Girls__Abracadabra_1080p.mp4.h264      6
YouTube_1080p   Short_Elephants_Dream_1080p_Full_HD_YouTube.mp4.h264    6
YouTube_1080p   Short_Full_HD_1080p_SNSD_Girls_Generation.mp4.h264      6
~
~
~
~
~
~
~
~
~
~
~
"TnC_Short_Raw_Decode_files.cfg" 10L, 711C                    1,1           All
```

Figure 18: Folder and Video Names in Configuration File

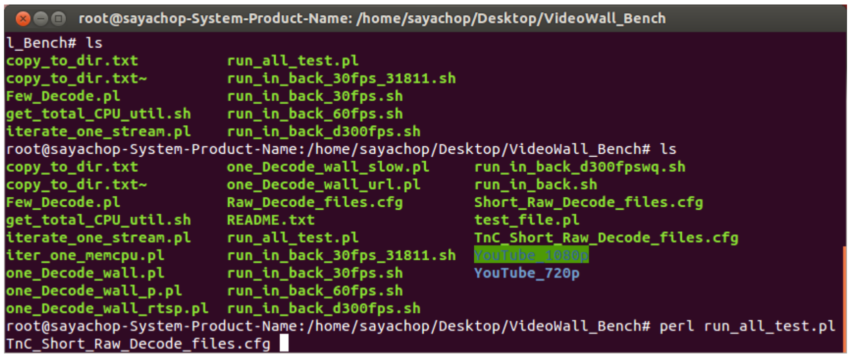
6) Other than that, user also must have to make sure that the directory of the video folders and files in copy_to_dir.txt are correct and precise. This is to make sure that all the result of .log and .logt of the video files will be copied properly into respective folders.



```
root@sayachop-System-Product-Name: /home/sayachop/Desktop/VideoWall_Bench

cp -f YouTube_1080p/Short_1080p_imax_hd_test_japan_hi_quality_full_hd_widescreen_
YT.mp4.h264  /home/sayachop/Desktop/VideoWall_Bench/YouTube_1080p/.
cp -f YouTube_1080p/Short_Avatar_FULL_1080p_trailer_YouTube.mp4.h264  /home/sayac
hop/Desktop/VideoWall_Bench/YouTube_1080p/.
cp -f YouTube_1080p/Short_Brown_Eyed_Girls__Abracadabra_1080p.mp4.h264  /home/say
achop/Desktop/VideoWall_Bench/YouTube_1080p/.
cp -f YouTube_1080p/Short_Elephants_Dream_1080p_Full_HD_YouTube.mp4.h264 /home/sa
yachop/Desktop/VideoWall_Bench/YouTube_1080p/.
cp -f YouTube_1080p/Short_Full_HD_1080p_SNSD_Girls_Generation.mp4.h264 /home/saya
chop/Desktop/VideoWall_Bench/YouTube_1080p/.
cp -f YouTube_720p/Short_720p_imax_hd_test_japan_hi_quality_full_hd_widescreen_YT
.mp4.h264 /home/sayachop/Desktop/VideoWall_Bench/YouTube_720p/.
cp -f YouTube_720p/Short_Avatar_FULL_720p2_trailer_YouTube.mp4.h264 /home/sayacho
p/Desktop/VideoWall_Bench/YouTube_720p/.
cp -f YouTube_720p/Short_Brown_Eyed_Girls__Abracadabra__720p.mp4.h264 /home/sayac
hop/Desktop/VideoWall_Bench/YouTube_720p/.
cp -f YouTube_720p/Short_Elephants_Dream_720p_Full_HD_YouTube.mp4.h264 /home/saya
chop/Desktop/VideoWall_Bench/YouTube_720p/.
cp -f YouTube_720p/Short_Full_HD_720p_SNSD_Girls_Generation.mp4.h264 /home/sayach
op/Desktop/VideoWall_Bench/YouTube_720p/.


~
                                                          1,0-1          All
```

Figure 19: Directory File

7) After finish with checking all the files, user can now proceed to run VideoWall Bench. To run VideoWall Bench, user needs to be in VideoWall Bench directory (home/sayachop/Dekstop/VideoWall_Bench). At this directory, user needs to execute command "perl run_all_test.pl TnC_Short_Raw_Decode_files.cfg".



```
root@sayachop-System-Product-Name: /home/sayachop/Desktop/VideoWall_Bench
l_Bench# ls
copy_to_dir.txt         run_all_test.pl
copy_to_dir.txt~        run_in_back_30fps_31811.sh
Few_Decode.pl           run_in_back_30fps.sh
get_total_CPU_util.sh   run_in_back_60fps.sh
iterate_one_stream.pl   run_in_back_d300fps.sh
root@sayachop-System-Product-Name:/home/sayachop/Desktop/VideoWall_Bench# ls
copy_to_dir.txt         one_Decode_wall_slow.pl     run_in_back_d300fpswq.sh
copy_to_dir.txt~        one_Decode_wall_url.pl      run_in_back.sh
Few_Decode.pl           Raw_Decode_files.cfg        Short_Raw_Decode_files.cfg
get_total_CPU_util.sh   README.txt                  test_file.pl
iterate_one_stream.pl   run_all_test.pl             TnC_Short_Raw_Decode_files.cfg
iter_one_memcpu.pl      run_in_back_30fps_31811.sh  YouTube_1080p
one_Decode_wall.pl      run_in_back_30fps.sh        YouTube_720p
one_Decode_wall_p.pl    run_in_back_60fps.sh
one_Decode_wall_rtsp.pl run_in_back_d300fps.sh
root@sayachop-System-Product-Name:/home/sayachop/Desktop/VideoWall_Bench# perl run_all_test.pl
TnC_Short_Raw_Decode_files.cfg
```

Figure 20: Command to run VideoWall Bench

8) The decoding process will be done by following the configuration setting that has been set by the user.

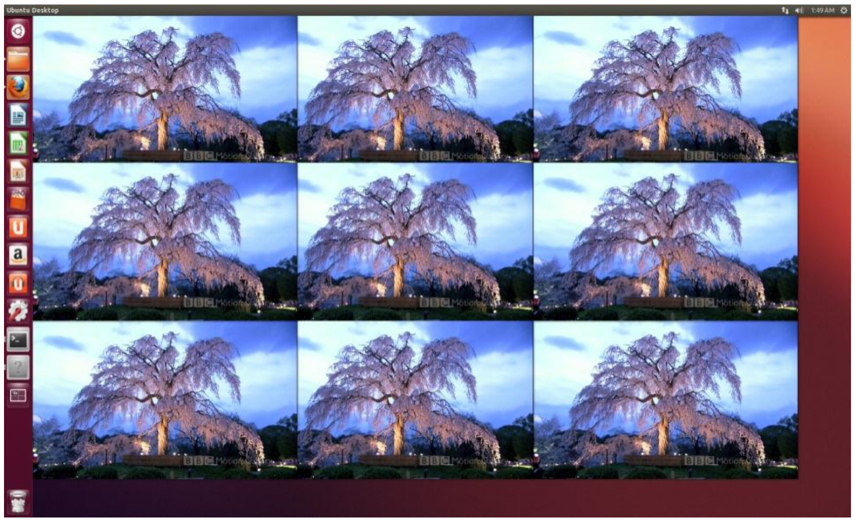
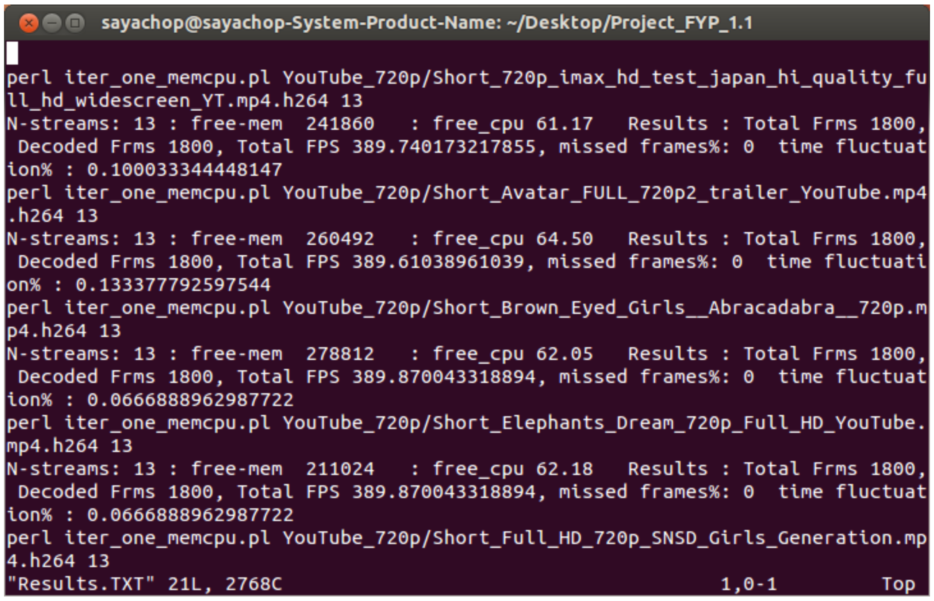


Figure 21: Running VideoWall Bench

9) After all the files have been successfully been decode by the Mplayer, a Result.txt file will be generated. This result file will display number of maximum video that can be decode, processor utilization, memory utilization, total frames, decoded frames, total FPS, missed frames and time fluctuation.



```
sayachop@sayachop-System-Product-Name: ~/Desktop/Project_FYP_1.1

perl iter_one_memcpu.pl YouTube_720p/Short_720p_imax_hd_test_japan_hi_quality_fu
ll_hd_widescreen_YT.mp4.h264 13
N-streams: 13 : free-mem  241860   : free_cpu 61.17   Results : Total Frms 1800,
 Decoded Frms 1800, Total FPS 389.740173217855, missed frames%: 0  time fluctuat
ion% : 0.100033344448147
perl iter_one_memcpu.pl YouTube_720p/Short_Avatar_FULL_720p2_trailer_YouTube.mp4
.h264 13
N-streams: 13 : free-mem  260492   : free_cpu 64.50   Results : Total Frms 1800,
 Decoded Frms 1800, Total FPS 389.61038961039, missed frames%: 0  time fluctuati
on% : 0.133377792597544
perl iter_one_memcpu.pl YouTube_720p/Short_Brown_Eyed_Girls__Abracadabra__720p.m
p4.h264 13
N-streams: 13 : free-mem  278812   : free_cpu 62.05   Results : Total Frms 1800,
 Decoded Frms 1800, Total FPS 389.870043318894, missed frames%: 0  time fluctuat
ion% : 0.0666888962987722
perl iter_one_memcpu.pl YouTube_720p/Short_Elephants_Dream_720p_Full_HD_YouTube.
mp4.h264 13
N-streams: 13 : free-mem  211024   : free_cpu 62.18   Results : Total Frms 1800,
 Decoded Frms 1800, Total FPS 389.870043318894, missed frames%: 0  time fluctuat
ion% : 0.0666888962987722
perl iter_one_memcpu.pl YouTube_720p/Short_Full_HD_720p_SNSD_Girls_Generation.mp
4.h264 13
"Results.TXT" 21L, 2768C                                        1,0-1         Top
```

Figure 22: Result.txt

4.4 Experimental Studies on Intel Atom

Video decode playback performance is judged in many ways. There are certain HD video benchmarking tests that focus on silicon performance in terms of video noise and spatial quality of video. For this experimental studies, performance is measured by focusing on CPU utilization, average frames per second being displayed, and time latency for acceptable user experience. This testing have used same H264 video files with two different resolutions which are 720p and 1080p. Besides that, I also try to compare the result between video decoding VA-API and video decoding non VA-API. Frame display for this video is 1441.

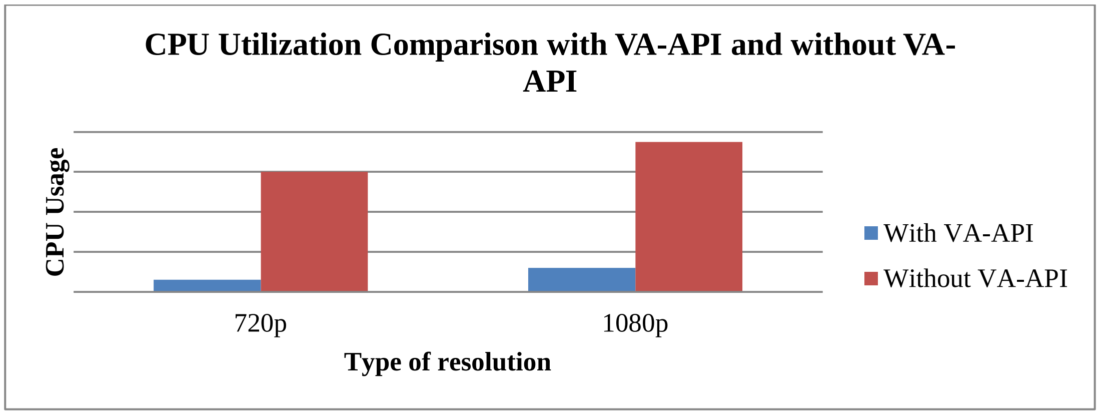


Figure23: CPU Utilization with VA-API and Without VA-API

Figure 23 shows comparison of CPU utilization with VA-API and without VA-API. From this diagram we can see that CPU utilization with VA-API is very low compare to CPU utilization without VA-API. This is because using VA-API, the video decoding process is using hardware acceleration. VA-API is used by calling the Library for VA-API (LibVa) from operation system and after that the CPU offloads the decoding process to the graphic driver. Integrated graphics then will play the video on the screen with minimal CPU utilization.

Video decoding process without VA-API state higher CPU utilization because the decoding process is been done by CPU itself. The CPU decodes the files and tells the graphic processor what to show. After that the video decoder paints the picture on the screen with high CPU utilization. We can see that when using VA-API, 720p files decoding only take half of CPU usage while 1080p is double of CPU usage that been consume for 720p. 1080p files consume higher CPU utilization because more rendering process needed to be done by the processor due to the quality of 1080p which is more superior then 720p files.

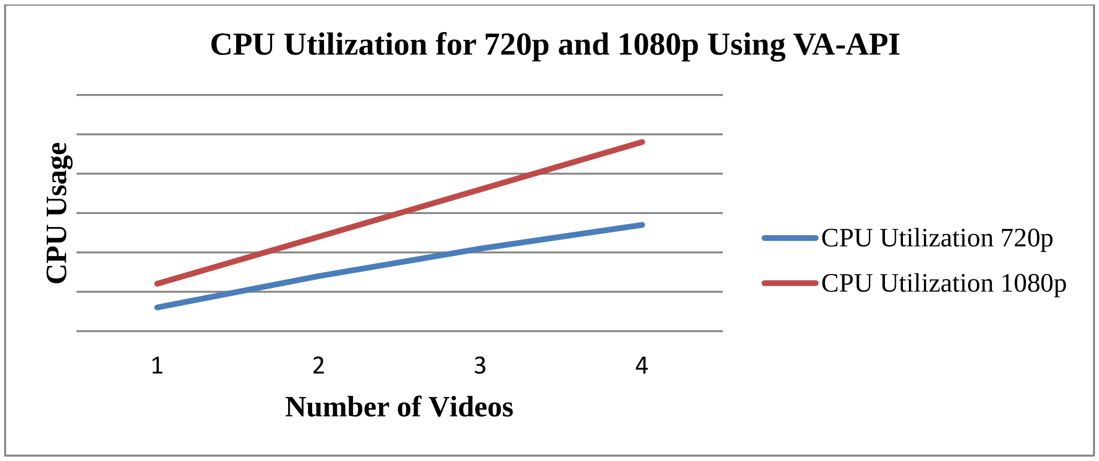


Figure 24: CPU Utilization for 720p and 1080p Using VA-API

Figure 24 shows the result pattern of CPU utilization for 720p and 1080p using VA-API. The higher the numbers of files being play, the higher the CPU usage that will be consume by processor. As you can see in the diagram, decoding process for one video of 1080p will consume double CPU usage of 720p decoding. We also observe that as the number of videos increase, total CPU utilization is equal to CPU usage of one video multiplied by number of videos. Using this diagram also we can know how much video decoding process that can be support by one atom platform. Although the CPU utilization for video decoding process VAAPI is smaller compare to non-VA-API, user must reserve some of processor usage for other process to run in the system. Besides that, it is also to ensure that the video decoding process will not cause the system hang or unresponsive due to high CPU usage on video decoding.

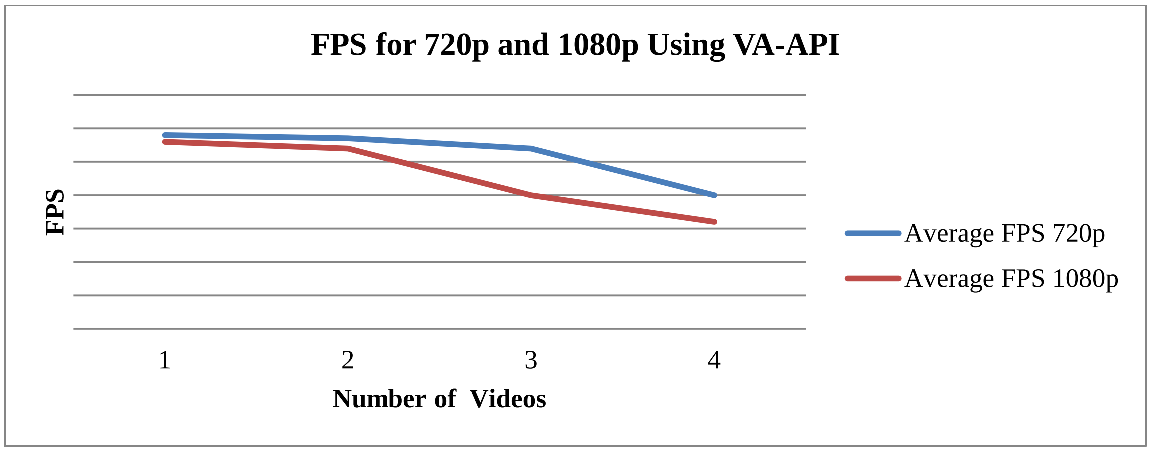


Figure 25: FPS for 720p and 1080p Using VA-API

Figure 25 shows FPS pattern for 720p and 1080p using VA-API. A video is typically encoded with a certain frame rate often but is not always aligned to the profile and level dictated by the specification. A video decoding system should ideally play back the video at the full frame rate at which the video stream was encoded. In this test, to measure the performance of the video decoding system we are using parameter –fps 66 to force the video to be decoded to 66 fps. If the average fps being measured is within 30 fps above which is ideal frame rate, then video playback is generally acceptable. In this research, the platform able to decode FPS for 720p is higher compared to 1080p because 720p file bitrates is less than 1080p causing the decoding process to achieve higher FPS using 720p file. The data also show when the numbers of video files increase, the FPS for each video files decrease. This is because the processor has to split up its decoding capabilities into several video files at the same time thus lower the FPS for each video files. FPS for four videos in 1080p is above 30 fps and it is slightly lower than 720p. Thus we can conclude that Intel Atom platform have amazing video decoding capabilities for 720p file and 1080p file. This conclusion is supported by proving the platform can successfully decode multiple video in 720p or 1080p with 0% of time latency.

4.5 Experimental Studies on Intel Sandy Bridge

At this stage, VideoWall Bench has successfully performed all of its intended functionality. Therefore, four experimental studies were conducted to benchmark the video decoding capabilities of Intel i5-2500k with Intel HD 3000 Integrated graphic card. Below were the experimental studies that have been conducted.

1. To find the number of maximum videos that can be display at one time using Intel i5 2500k Intel HD 3000
2. To test is there any performance difference in terms of processor, fps and memory utilization between H.264 and MP4 File
3. To test is there any performance difference in terms of processor, fps and memory utilization using different processor speed.
4. To test is there any performance difference in terms of processor, fps and memory utilization when decoding using different video size.

Result and Discussion for Experimental Studies 1

Experimental studies 1 was about to find the number of maximum videos that can be display at one time using Intel i5-2500k. In this test I was using 720p H.264 video files and 1080p H.264 video files. Figure 26 shows the maximum number of decoding for 720p video files on i5-2500k.

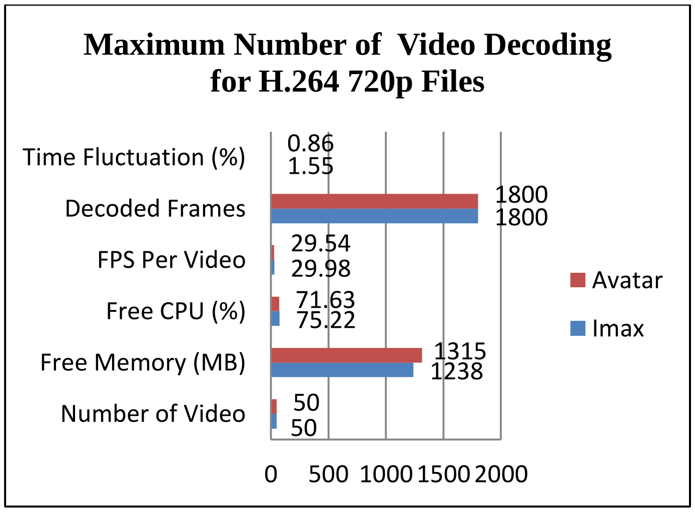


Figure 26: Maximum Number of Video Decoding for H.264 720p Files

In Figure 26 it shows that using i5-2500k, this computer was able to decode 50 videos at concurrently without having any problem. Decoded frames for videos Avatar and Imax show a complete decoding process which are 1800/1800. Besides, although there were 50 videos were being played concurrently, the FPS per video for Avatar and Imax shows a very good number which are 30 FPS per Video.

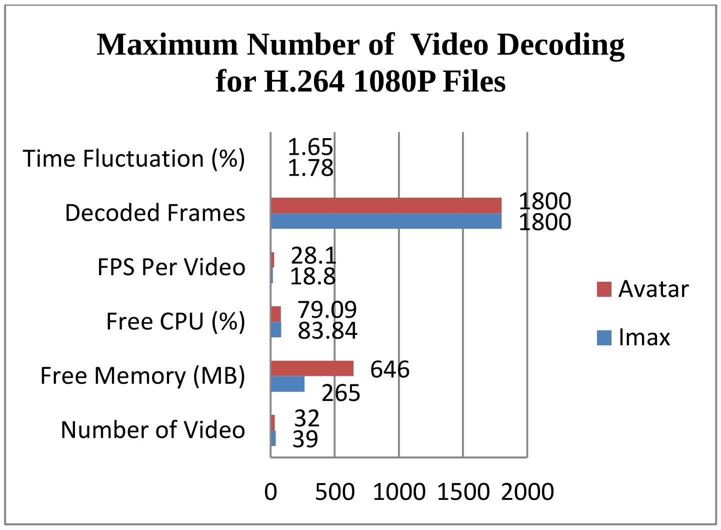


Figure 27: Maximum Number of Video Decoding for H.2641080p Files

As expected, the result for the maximum number of decoding for 1080p files is lower than the 720p files. Using 1080p files, i5-2500k can only able to decode 32 videos for Avatar and 39 videos for Imax. This is because the resolutions of 1080p files are higher compared to 720p. In terms of time fluctuation and decoded frames shows that these two files have been decode successfully with a small percentage of delay between them. However FPS per Video for Imax file which is 18.8 shown that there is higher possibility that during 39 video decoding, the video stuck and not being played in normal rate (25 FPS above). We also can observed that in both type of files, the CPU and memory still have a lot of power and free space but still the number of video decoding cannot exceed 50 and 39. This behavior has been discussed with Mr. Dennis E. Mungai.In that regard, memory bandwidth bottlenecks are inherited from system RAM, and for IGP's, that is a severely constrained environment that is often beyond the user's control. Integrated GPU like Intel HD 3000, not offer much performance improvement in regards to Memory Bandwidth. Integrated graphics do not have their own Video

RAM and they borrow it from main memory, depending on the Video Driver in use and the amount of RAM installed.

Result and Discussion for Experimental Studies 2

In second experiment, I want to show is there any performance difference in terms of processor, fps and memory utilization between H.264 and MP4 File video decoding. I used two video files which are Avatar and Imax. Both of the video files are same except one set in H.264 format and another in MP4 format. For this experiment, the technique that been used also was same like experiment one. Mplayer will try to decode maximum number of file so we can compare the result with H.264 files. Below is the result for MP4 File.

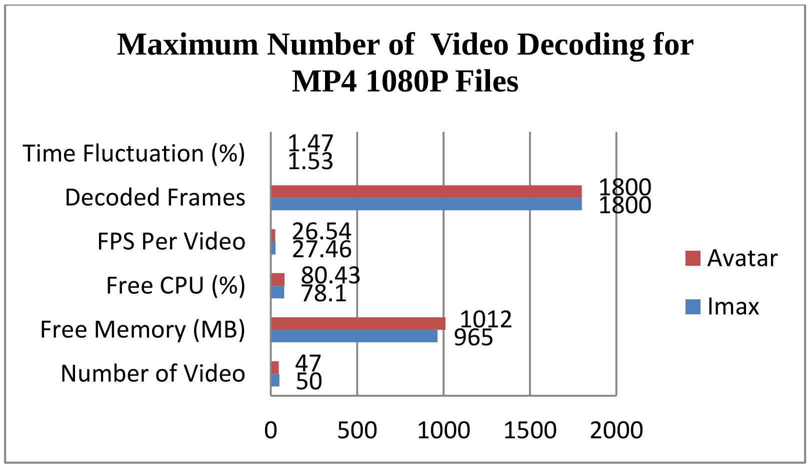


Figure 28: Maximum Number of Video Decoding forMP41080p Files

As we can see in Figure 28, the number of video files that can be decoded in MP4 format compared with H.264 is higher. Using MP4 format, i5-2500k can decode up until 50 and 47 videos at the same time compared to H.264 which is only 32 and 39 videos. Both of the files have same resolution but different video format. Although decoding using MP4 can achieve higher number of video file, it also excels in terms of FPS per video. Surprisingly MP4 decoding (50 videos) gets optimum FPS results which are 26.54 and 27. 46. Last but not least during decoding of 50 MP4 videos, it only used 20% of processing power compared to 83% for 39 videos of H.264. This is because MP4 containers impose the least bandwidth to Mplayer splitters whereas H.264 and AVI notably which with VC-1 Video, incurs massive bandwidth hits due to entropy encoding that must remain consistent and in sync at all times. Thus it enables MP4 files to be decoded in greater number compare to H.264 files.

Result and Discussion for Experimental Studies 3

Nowadays almost all processor can be overclock. Take an example such as Intel i5-2500k which have a unlock multiplier and it allow user to overclock their computer with a single click at the advance Bios menu. Therefore in experimental studies 3, I had made some testing regarding the effect of processor speed towards the performance of video decoding capabilities. In order to do so, this processor i5-2500k has been overclocked to 4.3 MHz. previously; the speed of the processor is 3.3 MHz

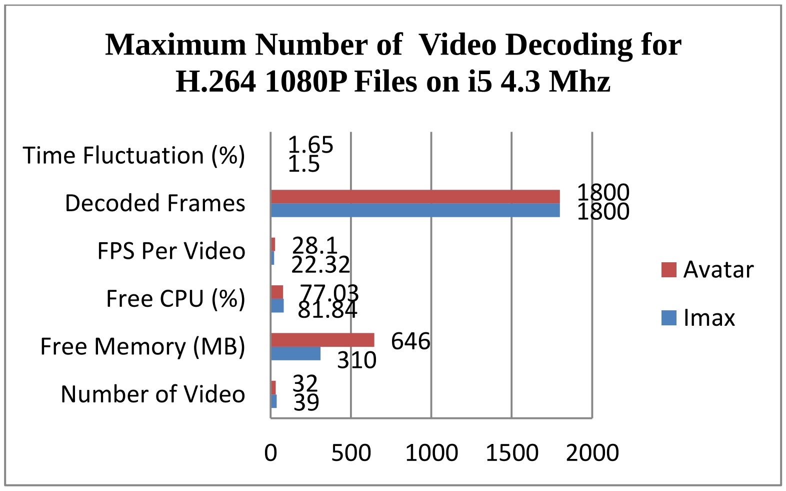


Figure 29: Maximum Number of Video Decoding for H.264 1080P Files on i5 4.3 MHz

As we can see in the Figure 29, overclocking the processor speed not brings any huge impact towards video decoding performance. It still can only decode 32 Avatar videos and 39 Imax video at the same time. Besides CPU utilization also almost the same as 3.3 MHz processing speed.

<u>Result and Discussion for Experimental Studies 4</u>

In this section it will show the effect of the screen size of video decoding towards the video decoding capabilities. User can change the screen size x and y at Iter_one_memcpu.pl.

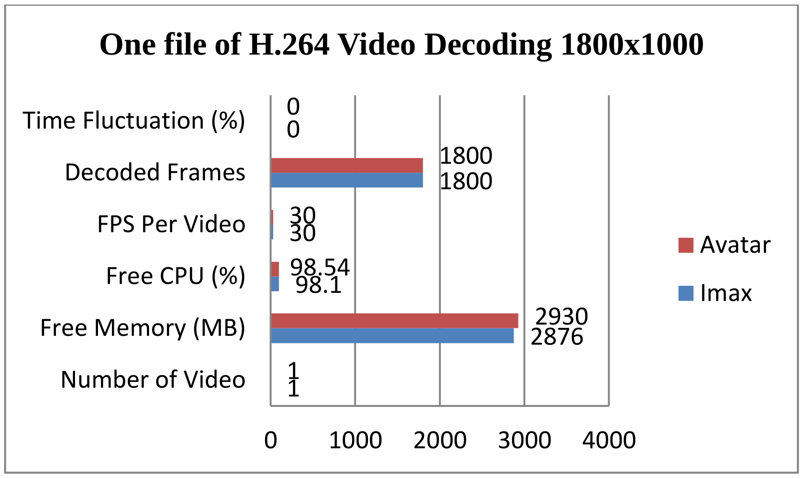


Figure 30: One file of H.264 Video Decoding 1800x1000

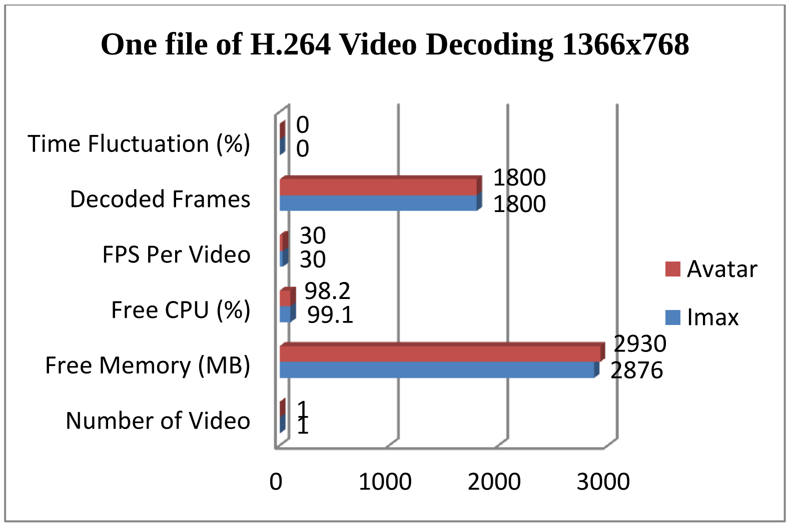
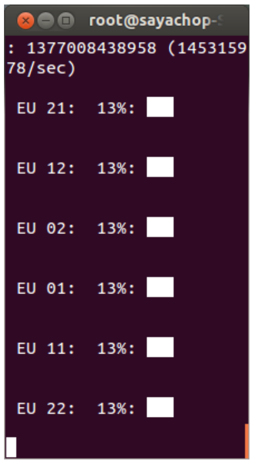


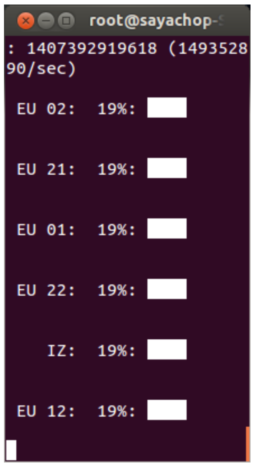
Figure 31: One file of H.264 Video Decoding 1366x768

Figure 30 and 31 shows that there were only small differences in terms of CPU utilization. On 1800x1000 screen size the CPU consumption was only 2% whereby on 1366x768 the CPU consumption was 1%. During video decoding process I also try a command to check GPU usage. Intel HD 3000 uses intel_gpu_top to monitor into the GPU usage during video decoding. GPU usage for video decoding on screen size 1800x1000 is 19% compared to screen size 1366x7678 which is 13%. Therefore we can conclude that the bigger the screen size the higher the GPU usage.



```
root@sayachop-
: 1377008438958 (1453159
78/sec)

 EU 21:   13%:

 EU 12:   13%:

 EU 02:   13%:

 EU 01:   13%:

 EU 11:   13%:

 EU 22:   13%:
```

Figure 32: GPU Usage for One file of H.264 Video Decoding 1366x768



```
root@sayachop-
: 1407392919618 (1493528
90/sec)

 EU 02:   19%:

 EU 21:   19%:

 EU 01:   19%:

 EU 22:   19%:

    IZ:   19%:

 EU 12:   19%:
```

Figure 33: GPU Usage for One file of H.264 Video Decoding 1800x1000

CHAPTER 5

CONCLUSION

5.1 Conclusion

This VideoWall Bench script successfully achieves its objective which enables the user to benchmark video decoding capabilities using VA-API. Using this program, user can know how much processor utilization and total frame rate per second for multiple video decoding using VA-API. Plus this script support most of the Linux operation system which is value added for this application compare to other benchmark software. This script also have been go through comprehensive testing by setting up an experimental studies to look for CPU usage, total FPS and time latency in video decoding using VA-API with various manipulated variable such as processor speed, different type of video format and screen size of the video display.

5.2 Recommendation

In future, this program can be improved by adding integrated graphic usage to check GPU utilization on video decoding using VA-API. This VideoWall Bench script also can be port to Android Platform to measure video decoding performance on mobile devices.

REFERENCES

BDTI H.264 Decoder Benchmark. (2006). Retrieved from bdti.com: http://www.bdti.com/Services/Benchmarks/H264

David, J. L. (2000). *Measuring Computer Performance: A Practitioner's Guide.* Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis.

Guobin Shen, G.-P. G.-Y.-Q. (2005). Accelerate Video Decoding With Generic GPU. *IEE Transactions on Circuits and Systems for Video Technology*, 695-693.

Jason, F. E., Frederick, S. W., Joseph, T. A., & Wayne, W. (2009). Media Bench II video: Expediting the next generation of video systems research. *Microprocessors and Microsystems 33* , 301-318.

Jason, F., Wayne, W., & Bede , L. (1999). Understanding Multimedia Application Chacteristics for Designing Programmable Media Processors. *SPIE Photonics West, Media Processors '99*, (pp. 2-13).

John, L. K. (2005). *Performance Evaluation and Benchmarking.* Taylor and Franchis Group .

Jörn, O., Jan, B., & Peter, L. (2004). Video coding with H.264/AVC: Tools, Performance, and Complexity. *IEEE Circuits and Systems*, 7-28.

Larabel, M. (2011, March 7). *Intel Sandy Bridge VA-API Video Acceleration Performance*. Retrieved from phoronix.com: http://www.phoronix.com/scan.php?page=article&item=intel_snb_video&num=1

Lee, C., Potkanjak, M., & H. Mangione-Smith, W. (1997). Media Bench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems. *30th International Symposium on Microarchitecture*, (pp. 330-335).

Markus, L. (2005). Evaluating Digital Entertainment System Performance. *IEEE Computer*, 68-72.

Mauricio, A., Esther, S., Alex, R., & Mateo, V. (2007). HD-VideoBench. A Benchmark for Evaluating High Definition Digital Video Applications. *IEEE Int. Symp on Workload Charaterization*, (pp. 120-125).

Mazlan, M. S. (2012, July). Guide on VideoWall Bench Requirements.

Michael, G., Yukio, W., & Takeshi, Y. (2006). Synergistic Processing in Cell's Multicore Architecture. *IEE Micro*, 10-24.

Nathan, S. T., & Alan, S. J. (2002). Design and Characterization of the Berkeley Multimedia Workload. *Multimedia Systems*, 315-327.

Thomas, S. (2005). Trends and Perspectives in Image and Video Coding. *Proceedings of the IEEE*, 6-17.

Video Acceleration API. (2012, June 12). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Video_Acceleration_API

Video Acceleration API. (2012, November). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Video_Acceleration_API

Weicker, R. P. (1990). An Overview of Common Benchmarks. *IEEE Computer*, 65-75.

Wichmann, H. J. (1976). A Synthetic Benchmark. *Computer Journal*, 43-49.

Willis, N. (2009, July 1). *VA API slowly -- but surely -- making progress*. Retrieved from lwn.net: http://lwn.net/Articles/339349/

X-Video Motion Compensation. (2012, January 16). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/X-Video_Motion_Compensation