
μSystems Research Group

School of Electrical and Electronic Engineering



**CPU Power Consumption Experiments and Results Analysis
of Intel i7-4820K**

Matthew Travers

Technical Report Series

NCL-EEE-MICRO-TR-2015-197

Contact: `m.travers@newcastle.ac.uk`

NCL-EEE-MICRO-TR-2015-197

Copyright © 2015 Newcastle University

μSystems Research Group

School of Electrical and Electronic Engineering

Merz Court

Newcastle University

Newcastle-upon-Tyne, NE1 7RU, UK

<http://async.org.uk>

CPU Power Consumption Experiments and Results Analysis of Intel i7-4820K

Matthew Travers

June 2015

Abstract

As we move into an ever more digital world, from communication techniques to data storage there has been an increase in the need of digital processors to process this information. We have come a long way since the early single core processors to a point where we can now have up to 8 cores in a single CPU (Central Processing Unit). Not only has the number of cores increased but the size has decreased and the speed increased. Whilst the world wants faster speeds this comes with a price, the price being power consumption. Power consumption is becoming a large factor to take into account since a lot of the processors being used today reside within mobile devices, for example mobile phones. These devices do not have an unlimited constant supply of electricity as they rely on batteries to operate and this is where a major problem is occurring. It is not just in mobile devices though where power consumption is a problem, as a society we are becoming more focused on green electricity and low power devices.

This paper is intended to outline some of the ways in which power usage can be reduced with different techniques such as splitting tasks between several cores. It will also show how frequency effects power consumption, another main subject within this paper will be the performance throughput compared to power consumption. Along with this Intel's RAPL counters will be considered to try and determine how accurate they are at measuring power. The first part of the paper will introduce some ideas and theories about power consumption and how it will be measured. The second part will focus more on the testing of the CPU.

1. Introduction

With the use of more advanced programs on digital devices there has arisen a need for faster CPU's to do the calculations in a reasonable amount of time. To get faster speeds we have had to increase the frequency that these CPU's work at, to do so we have to also increase the voltage and therefore more power is consumed by these processors. This has recently become a major concern due to the fact that the devices we are placing the CPU's in are mobile and can only operate from battery power. Not only is there this problem but also the problem that as whole we are using too much energy globally, so we are moving towards green energy and green devices that can still offer the performance requirements whilst being energy efficient [1]. Green energy products should not be discredited anymore and will make up a huge portion of the market in years to come.

There is only so much that can be done in terms of hardware to tackle this problem as frequency is proportional to power usage. So one solution is to actually to set parameters of how these devices are running to help reduce power consumption, especially with multicore devices [2]. This paper will consider the power consumption when running in different operating modes such as frequency and parallel computation. It will also investigate how accurate the Intel RAPL [3] counters are at measuring power usage.

This paper will first introduce some of the underlying properties of CPU's such as, how frequency is theoretically proportional to power consumption and how frequency affects performance. The paper will then outline the testing procedure along with the results acquired. Lastly there will be a discussion on what these results mean in real situations followed by a conclusion of the experiments as a whole.

2. Literature Review

2.1 Power Consumption of CPU Theory

The power consumption of a CPU is defined in many ways but the one proposed here will hopefully be comprehensive enough for the purposes of the paper. CPU's are based on complementary metal-oxide-semiconductor technology (CMOS) [4], which is used in many systems for example CPU's and DRAM. CMOS integrated circuits are built up from simple logic gates which are easy to analyse on their own but get complicated when there are as many as a billion of them on one silicon chip. CMOS technology theoretically only dissipates power when switching of states occurs; this is normally stated as the dynamic power of the circuit. There is however also leakage which is called static power, this is summed with the dynamic power for total power dissipation.

Static power dissipation can be split up into two factors; subthreshold conduction and tunnelling current through gate oxide layer [5]. The tunnelling power dissipation is becoming a large factor with the size of processors getting smaller. With the size decreasing so that many more transistors can be fit into the same area of chip a problem arises. The metal oxide layer becomes thinner and therefore easier for electrons to tunnel through the insulating layer. In this paper the Intel i7-4820k processor is used and it is based on 22nm transistor technology, the thickness of the insulating layer is roughly 0.5nm thick or around 2 atoms. With the insulating layers getting thinner whilst the supply voltage is staying the same tunnelling is the largest factor of leakage. This is the main source of static power dissipation. Since the power dissipation has no relation to the clock frequency it is simply a function of the supply voltage as shown below with m = constant and V = CPU core voltage [1].

$$P_{static} = m * V \quad (1)$$

Dynamic power dissipation can be split up into two factors causing power dissipation; Transition and short- circuit power dissipation. Transition power arises from the voltage source charging up the gates as if it is a capacitor and then the capacitor discharging to the ground. This process yields the following equation with f = operating frequency, V = core voltage and C = capacitance of the circuit [6].

$$P_{transition} = \frac{1}{2} CV^2 f \quad (2)$$

Most gates do not switch state every clock cycle so the variable “ α ” activity factor has to be added into the equation to make it a more robust approximation of the power consumption. The probability of the activity factor for data being below 0.1 is around 95% and for a clock would be 1 [7]. For these experiments we will take it as being 0.1 as we are processing data.

$$P_{transition} = \alpha \frac{1}{2} CV^2 f \quad (3)$$

The short-circuit power dissipation arises when both the transistors in a CMOS gate are on at the same time. This creates a short-circuit from the voltage supply to ground. This effect is a function of the frequency, therefore it becomes a problem at higher operating frequencies. It can be equated to the following equation with α = activity factor, $E_{short-circuit}$ = power of a single short-circuit and f = operating frequency.

$$P_{short-circuit} = \alpha * E_{short-circuit} * f \quad (4)$$

If the dynamic and static power is added together we arrive at the final equation representing the total power consumption of the CPU.

$$P_{CPU} = P_{static} + P_{short-circuit} + P_{transition} \quad (5)$$

$$P_{CPU} = (m * V) + (\alpha * E_{short-circuit} * f) + (\alpha \frac{1}{2} CV^2 f) \quad (6)$$

2.2 Multi-Core Power Efficiency

It is a common argument as to whether or not multi-core processing is more efficient than single core processing. It can be both more energy efficient and less, but it completely depends on the situation the multi-core processor is put in. In terms of a single thread it can be said that single core processors are more energy efficient than multi-core, this is because it has to power the other cores in the CPU even though they are not processing that thread. The voltage supplied to the CPU is the same across all cores, therefore it will waste energy powering the others. In an ideal situation this would be very minimal because there is an activity factor associated with the dynamic power. Due to the fact that the other cores are experiencing no activity the dynamic power of the idle cores will be very low. However there will still be static power losses which are directly proportional to the supply voltage.

Now consider we have two threads to be processed. If a single core is used for this it will have to work at 100% maximum workload to complete the task in a reasonable amount of time. This means that the core will be running at maximum frequency and voltage. Now if we split those threads across two cores it can complete the task in the same amount of time but with lower power consumption. This is achieved by the cores only having to run at around 50% workload and therefore can run at lower frequencies and voltages saving on power [8].

We can measure if it is more energy efficient to do multicore or single core processing by using the equation below [9].

$$\text{Energy Efficiency} = \frac{\text{Power}_{\text{serial}} * \text{Time}}{\text{Power}_{\text{parallel}} * \text{Time}} \quad (7)$$

In this equation we look at the time taken to process the same information in parallel and serial to see how power efficient multi-core processing is. From this equation the increase in performance for the same energy consumption can be calculated. If the value is less than one then it can be considered to be more energy efficient to run the process on multiple cores as opposed to a single core.

2.3 CPU Power Measurement

The power consumption of a CPU can be worked out through simple circuit theory. The CPU can be modelled as a variable resistor which changes its resistance as the workload increases. The power supply to the CPU provides a constant very stable 12V. The CPU will draw more current as its workload increases. Power dissipation of a resistor can be represented as.

$$P_{CPU} = V_{\text{supply}} * I \quad (8)$$

The supply voltage is known to be a stable 12V so the current is needed to get the power consumption. This will be discussed later in the paper. For the purposes of this paper we will assume that the EATX12V power supply is the only source of power to the CPU. This may in fact not be entirely true and the CPU could be also drawing power from elsewhere on the motherboard.

2.4 Processor Architecture / Power Management

In this paper the Intel i7-4820K processor will be discussed, this processor is part of Intel's Ivy Bridge-E product line. The Ivy Bridge-E processors are part of Intel's new 22nm technology. They say that this new technology is not only smaller but also faster and more energy efficient. These processors are a breakthrough in transistor production with it being

the first to use tri-gate technology for consumer products. The density that these transistors are packed together is around twice that of Intel's previous 32nm technology. With the 32nm transistor technology packing around 1.16 billion transistors into $212mm^2$ the Ivy Bridge technology packs around 1.4 billion transistors in to $160mm^2$ [10].

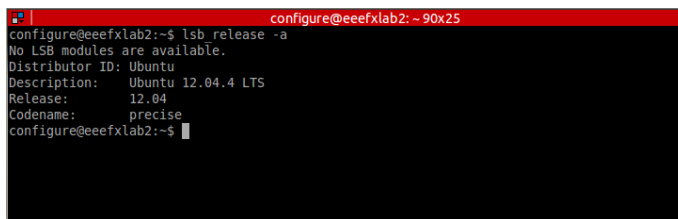
Intel has introduced several new power management systems in the new architecture. These include power gating and Power Aware Interrupt Routing (PAIR). These two systems work together and help keep power consumption down. If the user wants to be more energy conscious then PAIR will be activated. This program will stop interrupts going to idle cores when turned on so that they use no power. This cuts out a large portion of static leakage power but will somewhat reduce the performance of the system with the main activity being interrupted. It cuts off the power to the idle cores using the power gate which essentially turns off a particular core so to reduce the capacitance of the circuit.

3. Experiment Platform

3.1 Software

3.1.1 Operating System

For the following experiments a computer has been built from the ground up. Linux has been chosen as the preferred operating system due to the fact it has superior control over the hardware and sensors when compared to Windows or OSX. Ubuntu which is a Debian-based Linux distribution has been chosen due to its ease of use and functionality. Frequency switching and assignment of processes is made simple in Ubuntu as opposed to Windows therefore it is the best choice. It is also open source and many of the programs used on it are free and very customisable through the terminal. The release version used in the following experiments is Ubuntu 12.04.4 LTS.



```
configure@eeefxlab2:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 12.04.4 LTS
Release: 12.04
Codename: precise
configure@eeefxlab2:~$
```

Figure 1: screenshot showing the current version of Linux being run

3.1.2 Agilent BenchLink Data Logger 3

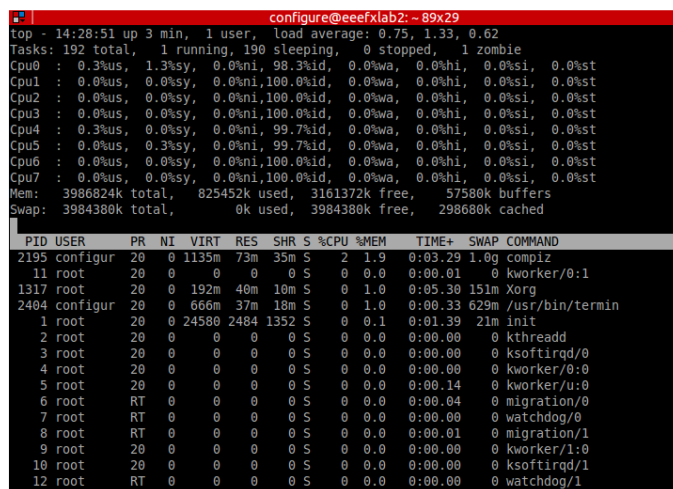
An Agilent 34970A multimeter is used to gather data from a shunt resistor in some experiments. This multimeter has a RS-232 connection so it can be used with a computer to log the data it acquires. The BenchLink data Logger 3 software makes it easier to log this data and graph it. Settings such as the rate it logs the data and averaging the results makes it perfect for calculating power usage. Another useful feature of the program is that it makes it possible to export the raw data to a text file so that it can be imported into a program such as excel for further processing.

3.1.3 Programs Used in Ubuntu's Terminal

Several programs are used in the experiments to either change parameters or detect changes whilst testing. The programs listed below are ones run in Ubuntu's terminal.

Top

This utility is one of many sensor utilities used for the experiments. It is able to display information on current processes running and which cores they are running on. It is also able to see how much each core is being stressed and is displayed as a percentage. For the experiments done later in the paper it is necessary to modify the information being displayed by Top. To do this once top has been opened a couple of command lines are needed. The first being ‘1’ to get individual core information on display and then ‘f p’ to get up information on which core a process is running on.



```

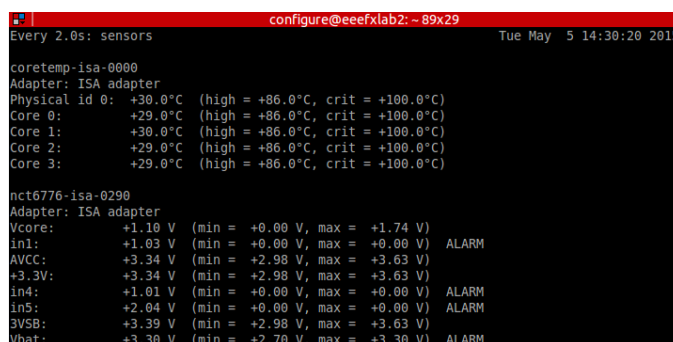
configure@eeefxlab2: ~$ top
top - 14:28:51 up 3 min, 1 user, load average: 0.75, 1.33, 0.62
Tasks: 192 total, 1 running, 190 sleeping, 0 stopped, 1 zombie
Cpu0 : 0.3%us, 1.3%sy, 0.0%ni, 98.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu2 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu3 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu4 : 0.3%us, 0.0%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu5 : 0.0%us, 0.3%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu6 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu7 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3986824k total, 825452k used, 3161372k free, 57580k buffers
Swap: 3984380k total, 0k used, 3984380k free, 298680k cached

  PID USER      PR  NI  VIRT  RES  SHR  %CPU  %MEM    TIME+  SWAP  COMMAND
 2195 configur 20   0 1135m 73m  35m  S    2.1  1.9   0:03.29  1.0g  complz
    11 root      20   0    0    0    0  S    0.0  0.0   0:00.01  0  kworker/0:1
 1317 root      20   0 192m  40m  10m  S    0.1  0.0   0:05.30 151m  Xorg
 2404 configur 20   0 666m  37m  18m  S    0.1  0.0   0:00.33 629m  /usr/bin/termin
    1 root      20   0 24580 2484 1352  S    0.1  0.1   0:01.39 21m  init
    2 root      20   0    0    0    0  S    0.0  0.0   0:00.00 0  kthreadd
    3 root      20   0    0    0    0  S    0.0  0.0   0:00.00 0  ksoftirqd/0
    4 root      20   0    0    0    0  S    0.0  0.0   0:00.00 0  kworker/0:0
    5 root      20   0    0    0    0  S    0.0  0.0   0:00.14 0  kworker/u:0
    6 root      RT   0    0    0    0  S    0.0  0.0   0:00.04 0  migration/0
    7 root      RT   0    0    0    0  S    0.0  0.0   0:00.00 0  watchdog/0
    8 root      RT   0    0    0    0  S    0.0  0.0   0:00.01 0  migration/1
    9 root      20   0    0    0    0  S    0.0  0.0   0:00.00 0  kworker/1:0
   10 root      20   0    0    0    0  S    0.0  0.0   0:00.00 0  ksoftirqd/1
   12 root      RT   0    0    0    0  S    0.0  0.0   0:00.00 0  watchdog/1
  
```

Figure 2: Screenshot showing top displaying the current utilisation of a core straight after the core number (%us) and the core a process is running on (%CPU)

Sensors

This program is used to view information on core voltages and temperatures. The program in default mode is satisfactory for use in the experiments so no modification is needed except for putting “watch” in front of the command so that the information is updated. The resulting information can be viewed below.



```

configure@eeefxlab2: ~$ sensors
Every 2.0s: sensors                               Tue May  5 14:30:20 2015

coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +30.0°C (high = +86.0°C, crit = +100.0°C)
Core 0:         +29.0°C (high = +86.0°C, crit = +100.0°C)
Core 1:         +30.0°C (high = +86.0°C, crit = +100.0°C)
Core 2:         +29.0°C (high = +86.0°C, crit = +100.0°C)
Core 3:         +29.0°C (high = +86.0°C, crit = +100.0°C)

nct6776-isa-0290
Adapter: ISA adapter
Vcore:          +1.10 V (min = +0.00 V, max = +1.74 V)
in1:            +1.03 V (min = +0.00 V, max = +0.00 V) ALARM
AVCC:           +3.34 V (min = +2.98 V, max = +3.63 V)
+3.3V:          +3.34 V (min = +2.98 V, max = +3.63 V)
in4:            +1.01 V (min = +0.00 V, max = +0.00 V) ALARM
in5:            +2.04 V (min = +0.00 V, max = +0.00 V) ALARM
3VSB:           +3.39 V (min = +2.98 V, max = +3.63 V)
Vbat:           +3.30 V (min = +2.70 V, max = +3.30 V) ALARM
  
```

Figure 3: Screenshot showing the sensors utility displaying Vcore voltage

Cpufreq

This program is useful for both monitoring and setting parameters of the CPU. It has two modes of operation which can be called up with different command lines. The first is called up by typing 'sudo cpufreq-set' into terminal. This program has many options to allow you to change frequency and how the computer chooses the correct frequency. For these experiments it is necessary to set the governor to userspace so that only the frequency decided by the user is used. This is a very useful feature and is displayed later on in the paper.

The second feature of this program is called up by typing 'sudo cpufreq-info'. This program is used to monitor and check what frequency each core is running at and is demonstrated later in the paper.

Stress

This simple program can set up stress situations on individual cores. It is completely computational and does not stress the hard drive or memory in the setup used in later experiments. It does this by calculating the square root many times over. It can be set to do this for a pre-defined amount of time. It can be called up by typing 'stress -c 1 -t 30s' into the terminal, in this example it runs 1 process for 30 seconds.

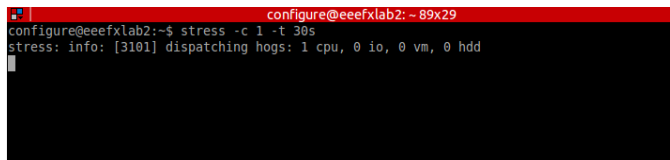


Figure 4: Screenshot displaying the stress utility running for 30 seconds with one instance running

Sys benchmark

This is the primary benchmarking program used within the tests. It works by testing if numbers are prime numbers or not up to a user defined maximum. It is completely computational and uses no memory and therefore is a good test for measuring the power consumption of CPU's. Unlike the stress test it can measure the time it takes to do the calculation and so it is a good test of the performance output as well as the power consumption. It can be called up by typing the following into the terminal 'sysbench – test=cpu –cpu-max-prime=20000 run'. In this example code it tests up to 20000 for prime numbers.

Likwid

Likwid can display a software model of power consumption based on Intel's RAPL counters. It is used later to comment on whether or not they are accurate models. To run the program simply type 'likwid-powermeter -s 1' into the terminal.

3.2 Hardware

3.2.1 Shunt Resistor

To measure the power consumption of the CPU module it is necessary to find out the current traveling through the CPU. One problem arises though and this is the fact that the current can get very high and the multimeter used can only measure up to 1A in DC. Therefore a shunt resistor is placed in the circuit and the voltage is measured across it, from this the current can be calculated when the resistance is known. The shunt resistors resistance is so small that it has negligible effect on the circuit. The specification of the shunt resistor is displayed below.

Primary current	50 A
Voltage drop	50 mV
<u>Power</u>	2.5 W
Fitting type	Top Hat rail mounting
Connection	Screw connector
Accuracy class	0.5
Height	23 mm
Width	30 mm
Length	135 mm
Type	50mV/50A

Table 1: Characteristics of the shunt resistor

3.2.2 Agilent 34970A / USB to RS232 Adapter

The Agilent 34970A is a high precision multimeter which is used in the experiments to gather data from the shunt resistor. It has the ability to connect to a computer by a RS232 connection which enables data logging in real time and then to export that data for use in calculations. It can display the voltage of the subject with up to 6 digit accuracy. The computer used to connect to the multimeter does not accept an RS232 connection therefore a RS232 to USB adapter is needed to make the connection.

3.2.3 Test Computer Specification

The test rig has been built from the ground up with the experiments to be performed on it in mind. The parts are chosen very carefully as to cater to the experiments. The computer has the following specification.

CPU	Intel i7-4820K
Motherboard	Asus P9 X79 LE
CPU Heatsink	Akasa Venom Voodoo
RAM	Crosshair 4GB DDR3
HDD	SanDisk 64GB SSD
Power supply	EVGA 1000G
Graphics card	EVGA GeForce 210 1024MB DDR3

Table 2: computer specification

Certain components have been chosen for specific reasons. The CPU has been selected due to the fact it is a current up to date model Intel is selling, some other factors is that it does not have a built in on board GPU which means the results will be more accurate when there is no power being used up for graphics. It also has features such as hyper threading and turbo boost. One last perk it provides is the unlocked multiplier on this particular model meaning that many different frequency steps can be selected.

The motherboard has been selected due to the fact this processor has to have a specific socket and there are not many on the market at the moment so the choice of motherboards is not the best. This particular one had good BIOS editing abilities and so is perfect for the testing. Some of the BIOS setting which are practical for the testing are things such as Vcore voltage locking, the ability to turn off speed stepping and the ease of use when switching operating frequency multipliers.

3.2.4 Shunt Resistor Setup

To make it feasible to measure the power that the CPU is drawing it is necessary to insert a shunt resistor in the CPU power supply circuit. This is because the multimeter chosen to record the data cannot handle the DC current expected to be going through the circuit. The motherboard chosen for the test setup uses an EATX12V connection to supply power to the

CPU. This connection is a 4-pin design with two live wires and two ground wires. The shunt resistor is placed on the ground route going back to the power supply. The schematic on the following page shows the placement of the Shunt resistor.

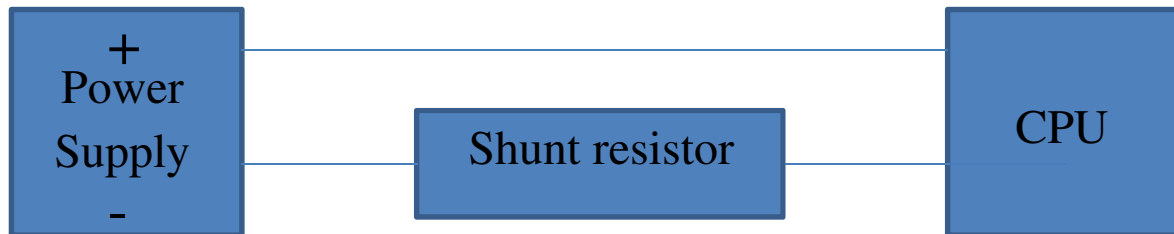


Figure 5: Shunt resistor setup in the CPU power circuit

The wires in the schematic above represent two wires as it is a 4-pin connection. The purpose of this experiment is to measure the power consumed by the CPU. From this setup it can be seen that the current will be the same going through the whole circuit due to the fact everything is in series. One thing to keep in mind is this is not entirely correct as the CPU has many connections and would be better described as a variable resistor but, we will consider more the shape of the curves and the relative power consumption in different operating modes. To calculate the power we can use the simple equation.

$$P_{total} = V_{supply} * I_{total} \quad (9)$$

It is known that the power supply will be supplying a constant very stable 12V through the EATX12V connection, therefore we need the current to calculate the power. As stated earlier we cannot use the multimeter to measure the current and so voltage is measured across the shunt resistor. Because the resistance of the resistor is known we can use the equation below to work out the current.

$$I_{total} = \frac{V_{shunt}}{R_{shunt}} \quad (10)$$

$$P_{total} = V_{supply} * \frac{V_{shunt}}{R_{shunt}} \quad (11)$$

Simple circuit theory will tell us that the power dissipation through the shunt resistor will affect the results. This is why a resistor with such a low resistance compared to the CPU has been chosen as to reduce this effect and make the results more accurate.

4. Investigation of Single-Core and Multi-Core Processing

4.1 Shunt Resistor Thermal Drift

It is necessary to calculate the resistance of the shunt resistor under the operating conditions it will be running in. The voltages that the shunt resistor is running under are very similar to that of the previous test rig conducted by Kaiwen Ju [11]. The swing of voltage is higher though with a peak voltage of around 1.1mV being recorded. Therefore the resistance is recalculated to be 5.73m Ω .

4.2 Preliminary Testing of the Shunt Power

An initial view of how the power changes with frequency is used to check the voltage range of the shunt so that the thermal drift can be calculated correctly. It also helps to view the overall power consumption of a CPU when its frequency is changed. To test this, a simple stress utility is used to stress the system to 100%. The frequency is then changed using Cpufreq-set to one of the available frequencies provided by Intel Speedstep. The following graph is produced over the full frequency spectrum.

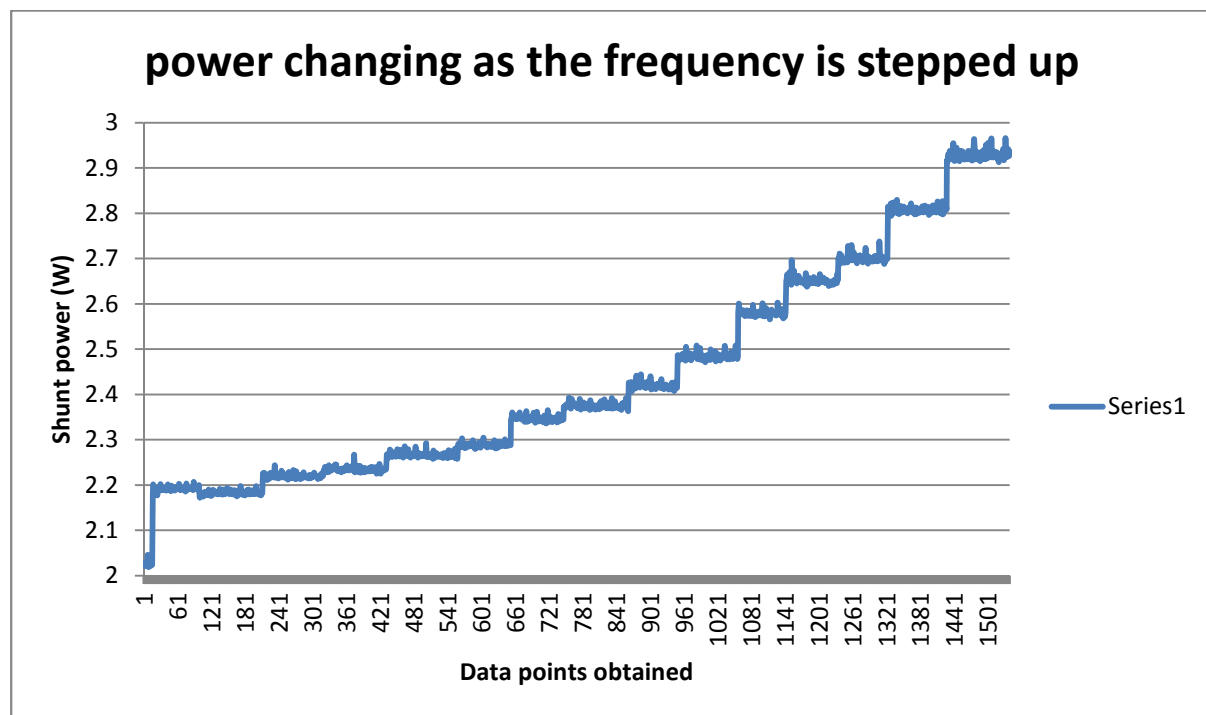


Figure 6: A graph describing the relationship between frequency and shunt power when fully loaded with a stress utility and using Intel Speedstep frequency steps

Here it can be seen that the power is related to the frequency as it goes up every time the frequency is changed. These values are then averaged and plotted with frequency against power.

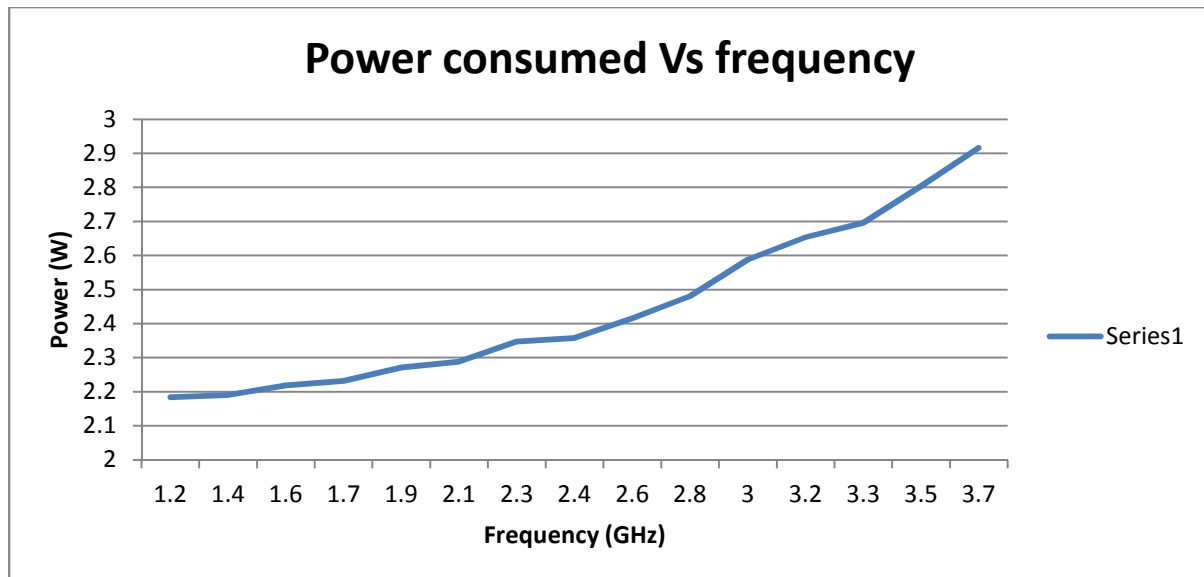


Figure 7: a graph describing the average power used by a frequency when being stress tested to 100% utilisation based on Intel's Speedstep frequencies

This graph gives a good indication of the power range the shunt resistor will be operating under and gives a good overview of the type of results that should be expected. This is a good starting point for the experiments but does not tell us enough information. For a better understanding of the CPU, energy is needed instead of power. Energy is a measure of the power used and the time it uses that power for. Therefore it is a good comparison between different frequencies when the time taken to complete a task goes down by a lot as frequency is increased.

4.3 Initial Frequency Vs Voltage Investigation

There is a need to investigate the characteristics of the CPU voltage when at certain frequencies. This is needed due to the fact that the voltage and frequency of a CPU when selected by the CPU are designated by some form of stepping. The i7-4820K uses an Intel South Bridge C1 stepping as default.

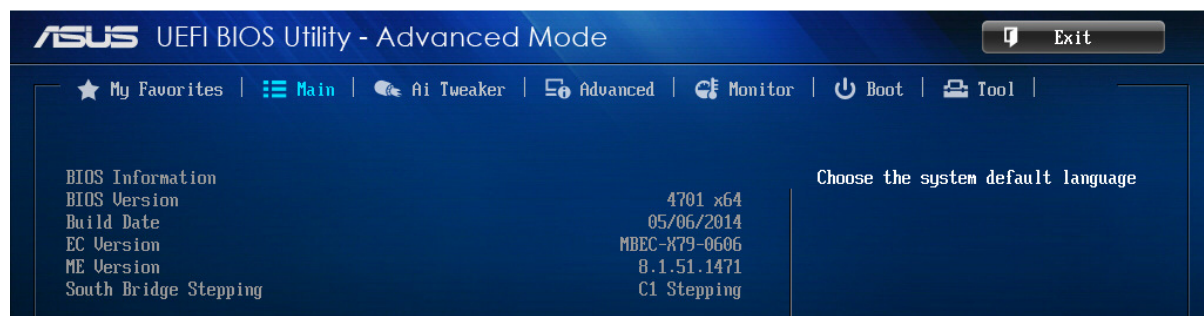


Figure 8: BIOS screenshot of South Bridge C1 stepping

This stepping function has pre-determined voltages for different frequencies. To carry out this experiment it is necessary to lock the frequency to certain values and then fully load the system to read off the voltage needed to obtain this frequency in a stable state. The reason for having to fully load the system is that when Intel SpeedStep is activated it also has something called C-states. These C-states turn off the cores and cache when they are idle which in turn means that the processor can lower the supply voltage and decrease power consumption. Later in the paper these states will be turned off as well as the Intel SpeedStep technology for a more thorough testing procedure. For these tests SpeedStep and C-states are active and in state “auto”.



Figure 9: BIOS screenshot showing SpeedStep enabled and C states in auto mode

Measurement of the CPU core voltage can be acquired through the terminal. The program “Sensors” is used for this measurement. Another program “Top” is used to check that the processor is being fully utilised and to verify which cores are being stressed.

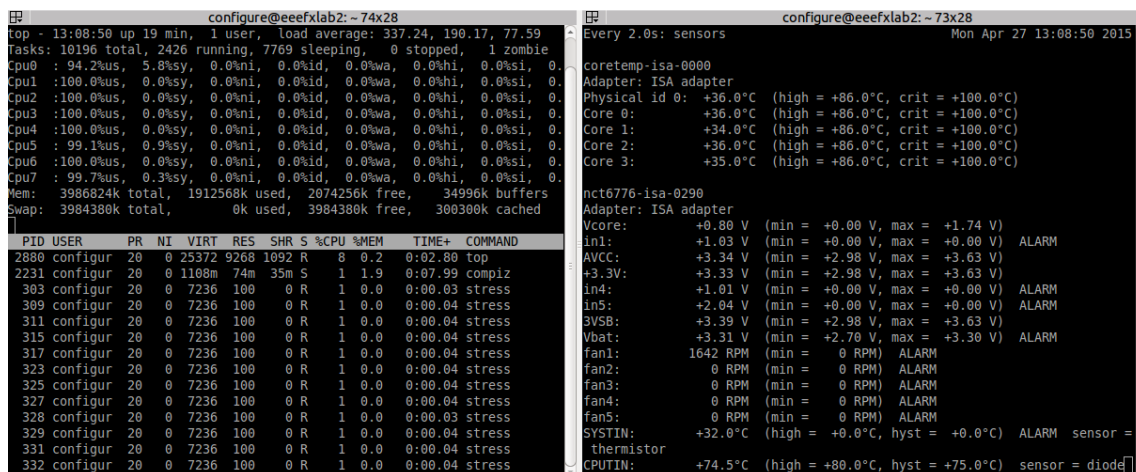
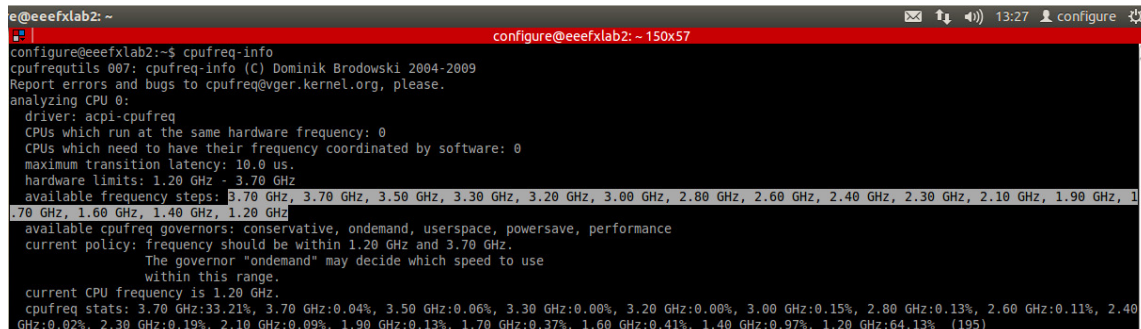


Figure 10: Terminal screenshot with Top on the left showing full CPU utilisation and Sensors on the right displaying Vcore=0.8V

To fix the frequency to specific values the program “cpufreq” is used. This program takes advantage of Intel’s SpeedSteps and therefore only pre-defined values can be used. To view

what frequency the cores are currently operating at and the available steps, the program is put into info mode “cpufreq-info”. This gives a comprehensive view of the current frequency information.



```

e@eeefxlab2: ~
configure@eeefxlab2: ~ 150x57
configure@eeefxlab2:~$ cpufreq-info
cpufrequtils 0.07: cpufreq-info (C) Dominik Brodowski 2004-2009
Report errors and bugs to cpufreq@vger.kernel.org, please.
analyzing CPU 0:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 10.0 us.
  hardware limits: 1.20 GHz - 3.70 GHz
  available frequency steps: 3.70 GHz, 3.70 GHz, 3.50 GHz, 3.30 GHz, 3.20 GHz, 3.00 GHz, 2.80 GHz, 2.60 GHz, 2.40 GHz, 2.30 GHz, 2.10 GHz, 1.90 GHz, 1.70 GHz, 1.60 GHz, 1.40 GHz, 1.20 GHz
  available cpufreq governors: conservative, ondemand, userspace, powersave, performance
  current policy: frequency should be within 1.20 GHz and 3.70 GHz.
    The governor "ondemand" may decide which speed to use
    within this range.
  current CPU frequency is 1.20 GHz.
  cpufreq stats: 3.70 GHz:33.21%, 3.70 GHz:0.04%, 3.50 GHz:0.06%, 3.30 GHz:0.00%, 3.20 GHz:0.00%, 3.00 GHz:0.15%, 2.80 GHz:0.13%, 2.60 GHz:0.11%, 2.40 GHz:0.02%, 2.30 GHz:0.19%, 2.10 GHz:0.09%, 1.90 GHz:0.13%, 1.70 GHz:0.37%, 1.60 GHz:0.41%, 1.40 GHz:0.97%, 1.20 GHz:64.13% (195)
  
```

Figure 11: Terminal screenshot showing frequency information, highlighted area are the available South Bridge C1 steps

To change the frequency the other mode of the program is used “cpufreq-set”. In this mode it is possible to set the governor that selects what frequencies the CPU should use. When the user wants the CPU to use only one particular frequency rather than a range the governor “userspace” is used. After this is activated it is possible to change the value of every core to a pre-defined value.



```

e@eeefxlab2: ~
configure@eeefxlab2: ~ 149x55
configure@eeefxlab2:~$ sudo cpufreq-set -g userspace
[sudo] password for configure:
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 0
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 1
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 2
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 3
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 4
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 5
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 6
configure@eeefxlab2:~$ sudo cpufreq-set -f 1.2GHz -c 7
configure@eeefxlab2:~$
  
```

Figure 12: Setting the governor to userspace and then setting the frequency of each core to 1.2GHz

Once the frequency is chosen a simple stress utility is used to get the cores up to 100% utilisation. Only the CPU wants to be stressed in this example so a purely mathematical stress is used whereby it executes a square root calculation over and over.

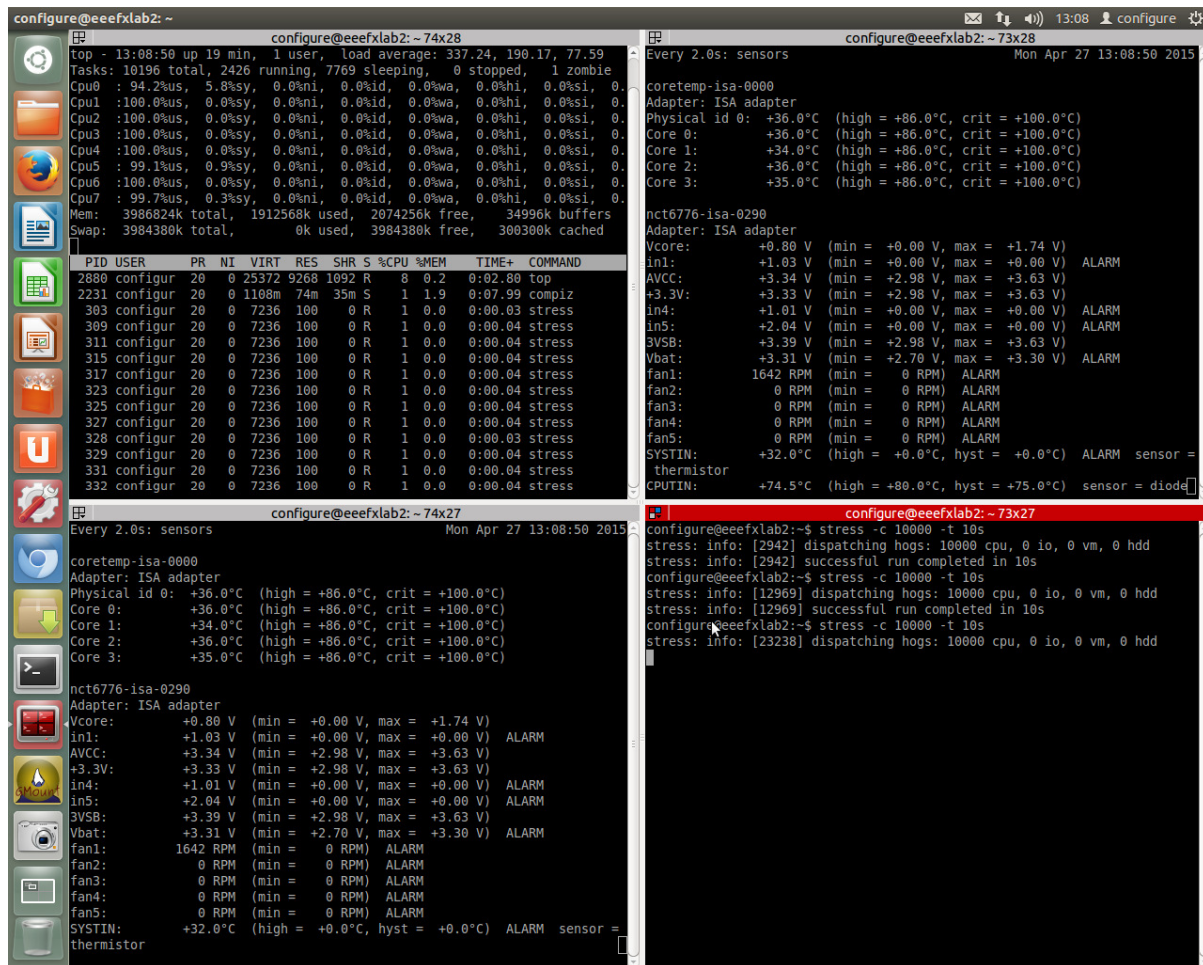


Figure 13: Terminal view of Top, Sensors and stress utility stressing the system to 100%

This shows that the CPU is in its maximum high occupation and the voltage can be read from the program Sensors. The voltage of the CPU is displayed as Vcore. This test is repeated at all the available frequencies and then the voltage against frequency is plotted.

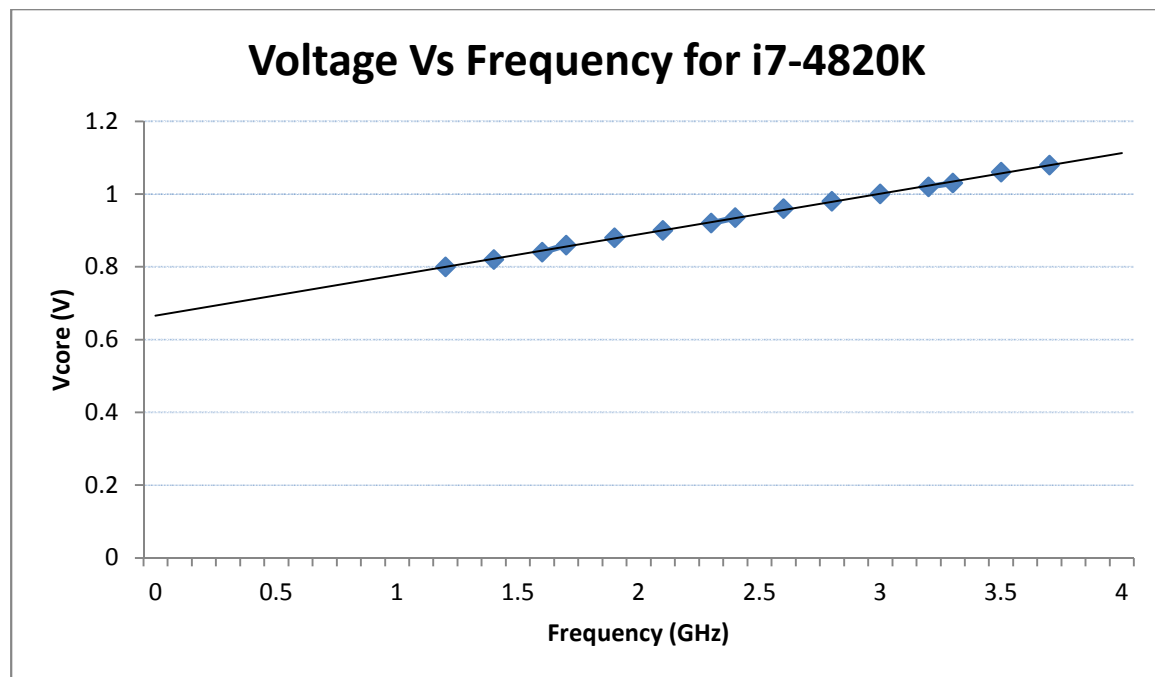


Figure 14: Graph describing the relationship of voltage with frequency increases as set by Intel South Bridge C1 stepping

It can be seen that the voltage scales linearly with respect to the frequency.

4.4 Investigation of Single-Core and Multi-Core Processing

To test the power consumption and the performance throughput of the CPU the program “sysbench” is used. This program checks for prime numbers, it does this by moving up from 0 to a user defined number and dividing each time by every number from 0 to the square root of the number until it reaches the user defined maximum. It outputs the time taken to do this calculation so it is therefore a good statistic for performance along with power consumption. For the purposes of this test it is necessary to have two instances of this program running so that it can be compared when splitting the threads across multiple cores or both on a single core. To fix the affinity of the program to a single core the program “Taskset” is used. This can fix a particular thread to a chosen core.

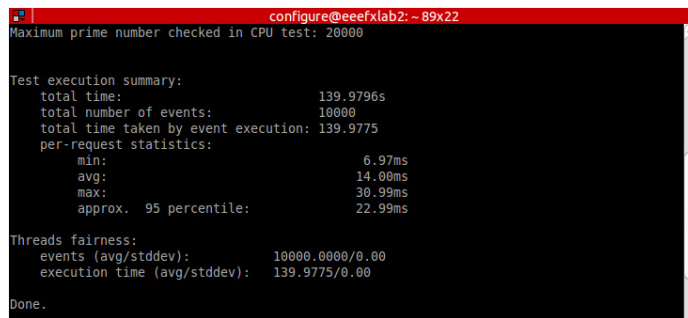
```

configure@eeefxlab2: ~149x55
configure@eeefxlab2:~$ taskset -c 0 sysbench --test=cpu --cpu-max-prime=20000 run & taskset -c 0 sysbench --test=cpu --cpu-max-prime=20000 run

```

Figure 15: two instances of Sysbench running on core 0 using Taskset to select the affinity

Once the program has run it outputs several pieces of information. From this the important piece of information is the total time taken to execute the threads.



```
configure@eeefxlab2: ~89x22
Maximum prime number checked in CPU test: 20000

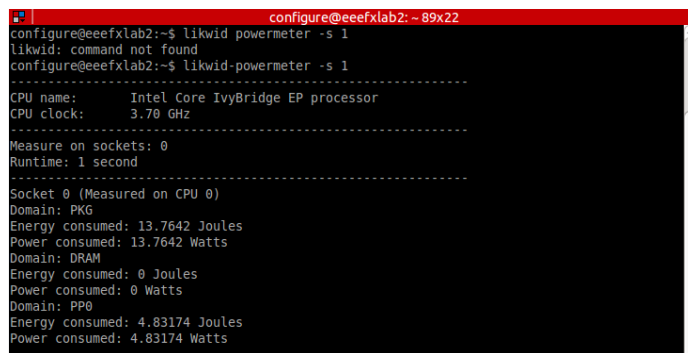
Test execution summary:
  total time:          139.9796s
  total number of events: 10000
  total time taken by event execution: 139.9775
  per-request statistics:
    min:              6.97ms
    avg:              14.00ms
    max:              30.99ms
    approx. 95 percentile: 22.99ms

Threads fairness:
  events (avg/stddev): 10000.0000/0.00
  execution time (avg/stddev): 139.9775/0.00

Done.
```

Figure 16: Sysbench output for 20000 prime number check

Another program called “Likwid” is also run whilst this process is happening. Likwid when run in powermeter mode uses Intel’s RAPL counters to estimate the power usage at a particular point in time. It can be seen below how it outputs this information.



```
configure@eeefxlab2:~$ likwid powermeter -s 1
likwid: command not found
configure@eeefxlab2:~$ likwid-powermeter -s 1
-----
CPU name:      Intel Core IvyBridge EP processor
CPU clock:     3.70 GHz
-----
Measure on sockets: 0
Runtime: 1 second
-----
Socket 0 (Measured on CPU 0)
Domain: PKG
Energy consumed: 13.7642 Joules
Power consumed: 13.7642 Watts
Domain: DRAM
Energy consumed: 0 Joules
Power consumed: 0 Watts
Domain: PP0
Energy consumed: 4.83174 Joules
Power consumed: 4.83174 Watts
```

Figure 17: Likwid-powermeter output for 1 second

From this the PKG power is used due to the fact this is the power for the whole CPU package and is more closely related to the true power usage. This process of testing is repeated over the full frequency spectrum of the CPU defined by Intel’s SpeedStep. The power going through the shunt resistor is also measured throughout the experiments so that the results can hopefully be compared to Intel’s RAPL counter measurements.

From these results several statistics can be analysed. Firstly a look at how frequency effects power consumption recorded by Likwid.

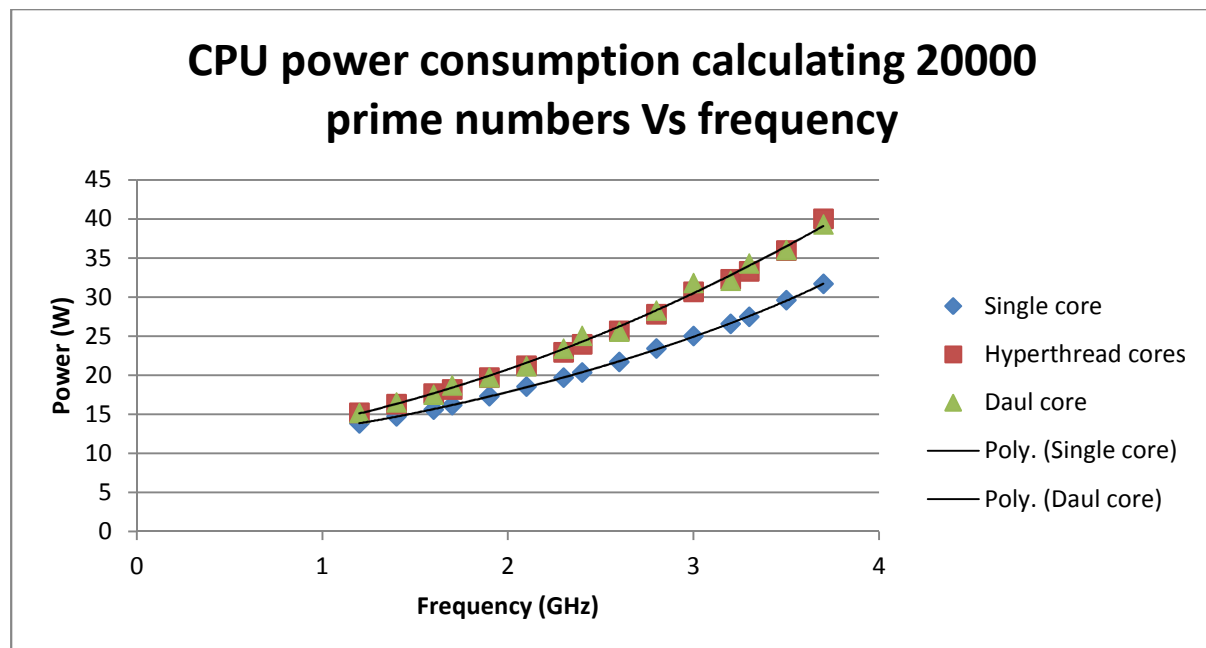


Figure 18: This graph describes how much power is used to complete the task of calculating if numbers are prime or not up to 20000. The results are data collected from Likwid using Intel RAPL counters

There is an overall increase in power consumed by the CPU as frequency is increased. When we split the threads across a real and a Hyper-Thread core as well as 2 real cores it uses more power as frequency is increased. From Figure (18) it can be seen that the power consumption of splitting threads is almost identical on both the Hyper-Thread core and a second real core. From these findings it could be assumed that it is less efficient to use multiple cores as opposed to a single core but power consumed is not the full story. The time taken to complete the tasks is now considered.

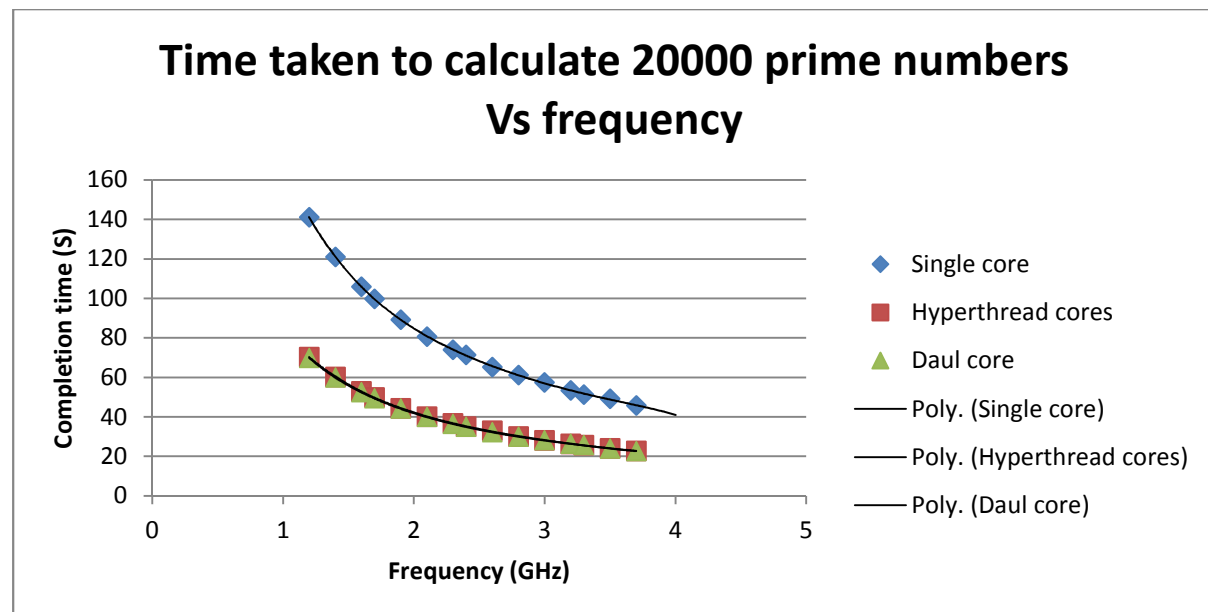


Figure 19: This graph shows the time taken to complete the task of calculating 20000 prime numbers when operated at different frequencies.

It can be seen in Figure (19) that the time taken when splitting the threads is a great deal less than if they are both on the same core. Again the splitting of threads across a real core and a Hyper-Thread or a second real core yields the same results. Splitting the threads across multiple cores reduces the total time taken by an average of 49.6%. This is a massive increase in performance from simply splitting the threads.

From these two pieces of information it is now possible to look at the process in terms of energy used which is the best comparison on which is more energy efficient.

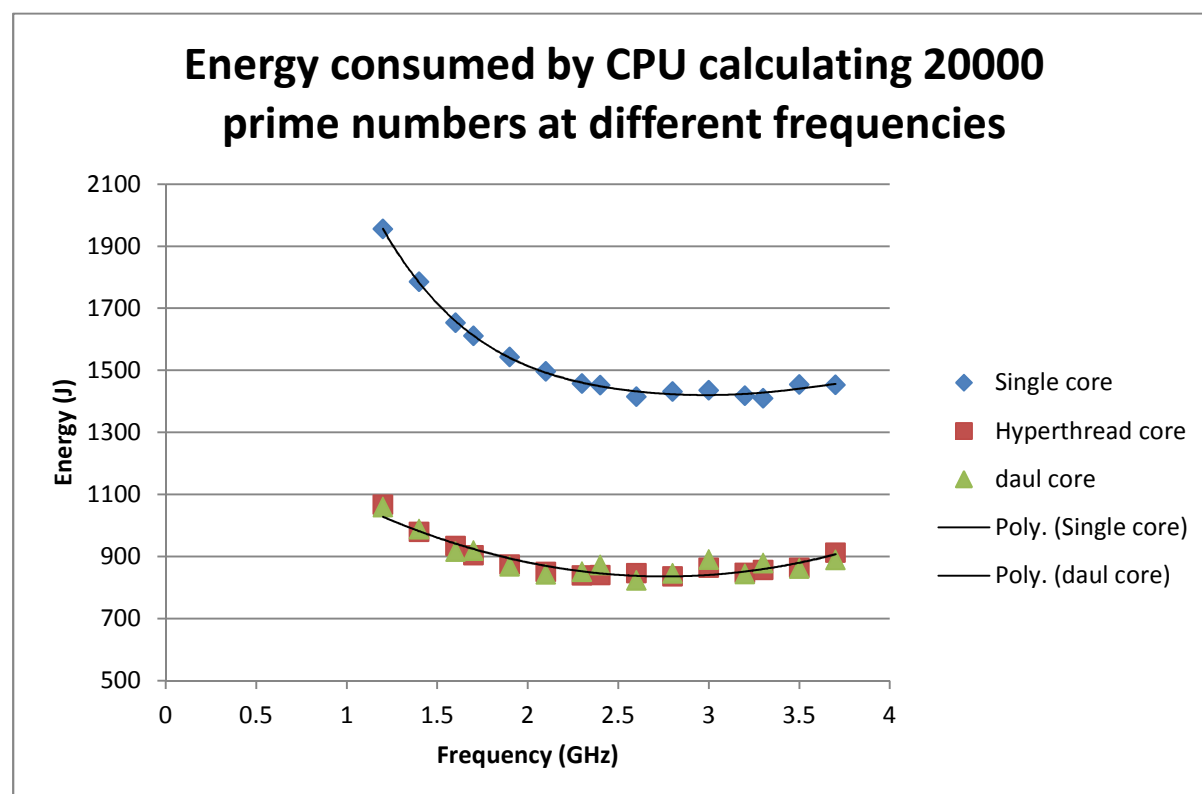


Figure 20: This graph shows the energy consumed when running in different modes of parallelism

The energy consumed by splitting the threads across multiple cores is around 40% less. This is a massive decrease in energy consumed and clearly show how multi core processing can be much more energy efficient than single core processing. The time taken to complete the tasks is also significantly less which is great for the end user who wants a faster system. It can be seen though as the speed increases to its maximum the energy starts to rise again. This can be due to the fact it is reaching its maximum output and the power scales with the square of the voltage or could be due to the fact it is reaching the Thermal Design Power (TDP) limit and is not performing as well as described because all 4 cores are being run at maximum capacity. Therefore when higher voltages are used it does start to become less efficient and more performance orientated. But as a whole this is a good thing as it can balance between being energy or performance conscious.

The data collected from the Shunt resistor follows closely with those gathered from Likwid's RAPL counter results. They do however not give the same power readings as expected but the shape of the trend lines is very similar

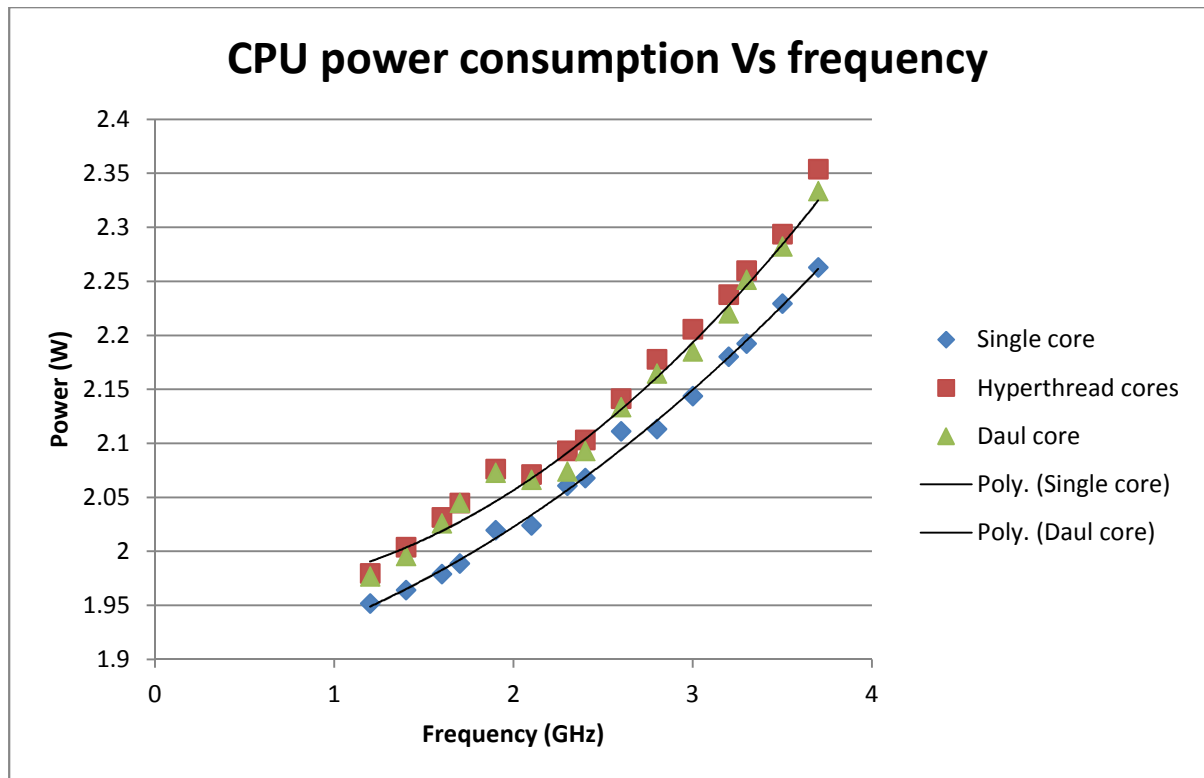


Figure 21: This graph shows the relationship between power calculated by the shunt resistor and frequency when operated in different modes of parallelism

The splitting of threads across two cores again yields an increase in power consumption as expected. But with these results the time taken is the same as before and so it has been found that splitting of cores again reduces the energy needed to complete the tasks.

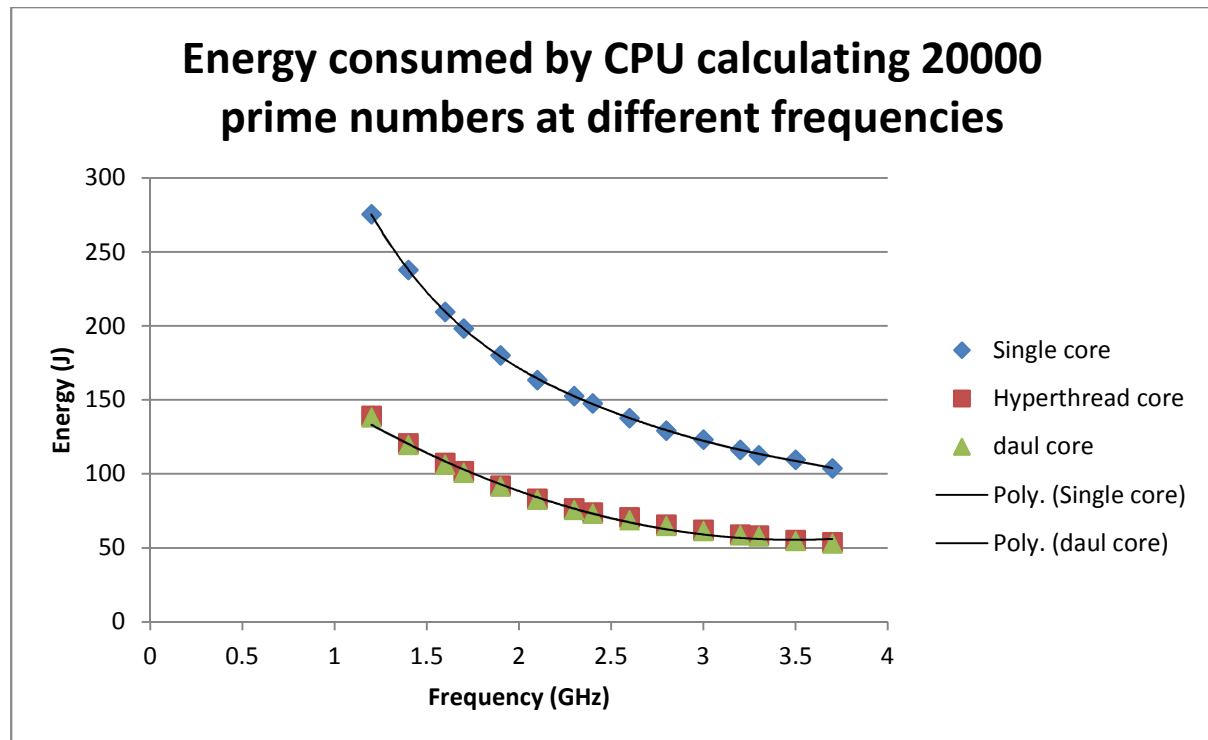


Figure 22: Energy consumed by CPU calculated by the shunt resistor when running at different frequencies

Here it can be seen that they have very similar results to the ones calculated by the RAPL counters, but there is no increase in energy towards the higher end of the frequency spectrum which is apparent in the results obtained Through Likwid.

To get a better feel to the efficiency increase gained by splitting threads at the same frequency we will use the following equation as discussed in section 2.2. This will highlight the similarity between both the RAPL counter and the Shunt resistor results.

$$Energy\ Efficiency = \frac{Power_{serial} * Time}{Power_{parallel} * Time} \quad (7)$$

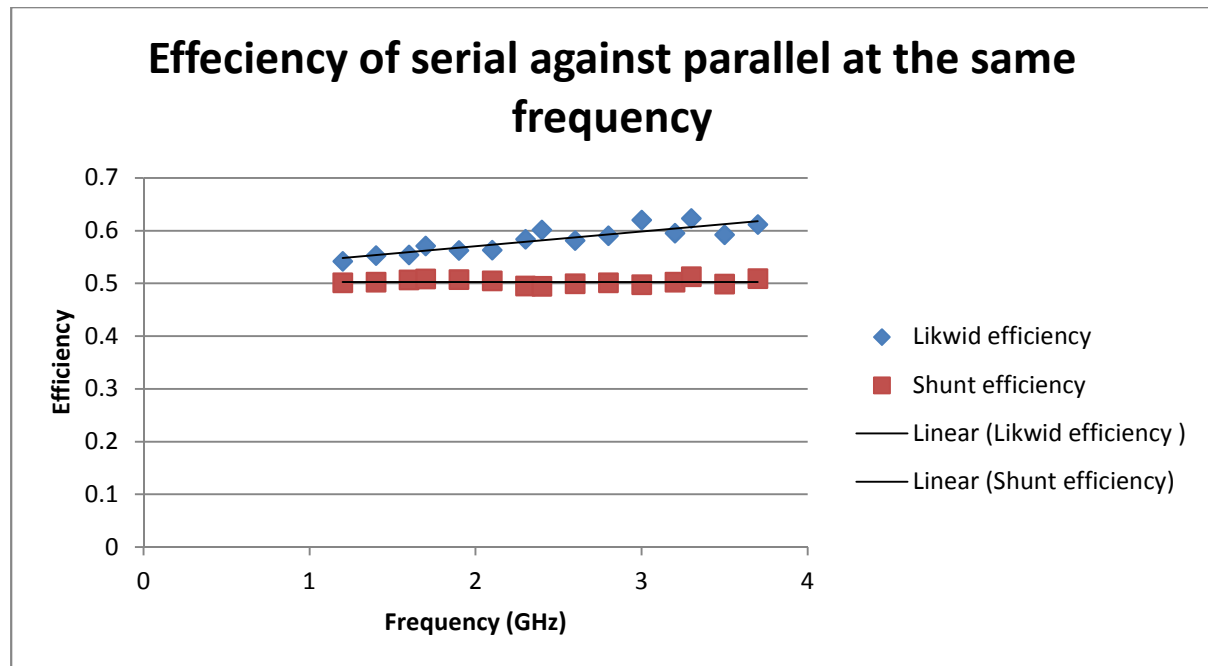


Figure 23: a graph showing how much more efficient it is to split threads across multiple cores as opposed to a single core

It can be seen that the results obtained through the shunt resistor give a very stable 50% reduction in energy consumed with splitting the cores across either two real cores or a core and its imaginary Hyper-Thread core. This heavily backs up the theoretical power savings that multi-core processors claim they can achieve. The RAPL counters however show the processors to be slightly less efficient with it saving only around 38-46% of power. It is interesting to note that the efficiency of the processors decreases with an increase in frequency when looking at the RAPL counter results.

4.5 Power Map of the i7-4820k

A power map of the i7-4820k will show just how much power the device is using over all frequencies and voltages. It will also show how much the total power is made up of both static and dynamic power. To test this theory several tests are conducted that would get the correct equation for the power.

$$P_{CPU} = (m * V) + (\alpha * E_{short-circuit} * f) + (\alpha \frac{1}{2} CV^2 f) \quad (6)$$

The equation for power consumption is split up into three parts; transition power, short-circuit power and static power. Firstly the static power will be obtained; this can be done by keeping the voltage the same and then changing the frequency. The reason for this is that the static power is frequency independent and therefore it should have no effect on the results.

Once the results at 3 different voltages had been obtained, then a graph of power against voltage could be created from the y-intercept values when frequency is zero. To make sure that the CPU is in a fully utilised state, Sysbench is used to put all the cores to 100%. This is done by affinity locking an instance of Sysbench to each core using Taskset.

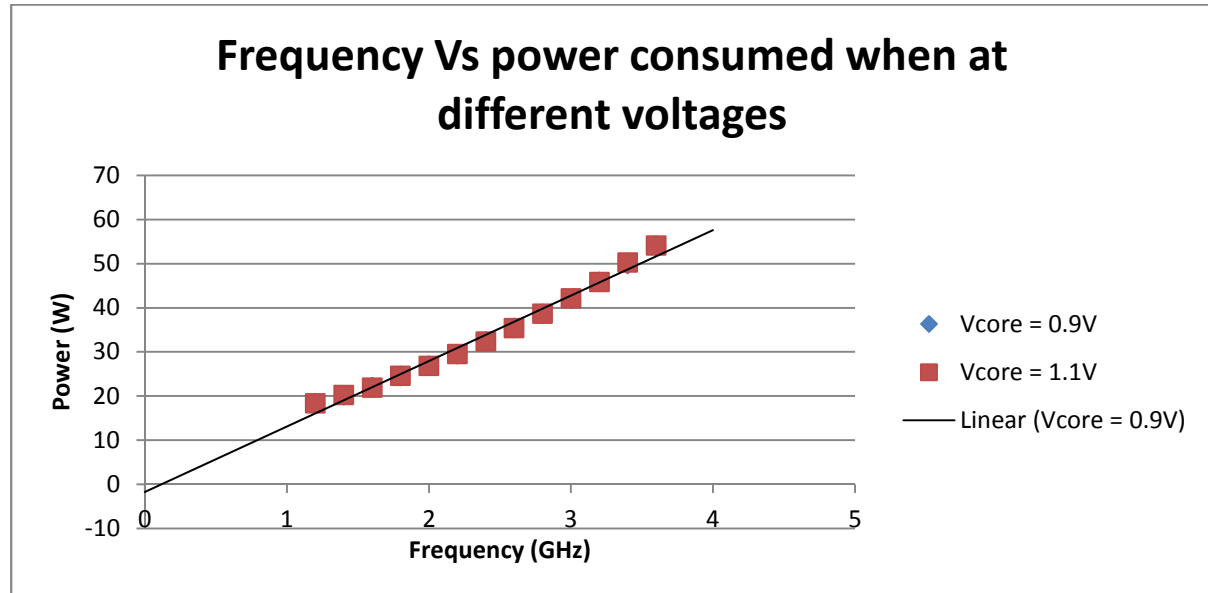


Figure 24: a graph showing how power is related to frequency with power readings collected from Likwid. These results are obtained through Likwid and therefore use Intel's RAPL counters to get the results. As it can be seen in the graph it does not matter what voltage the CPU is running at it gives the same results. This is not expected as this would lead to the conclusion that power is voltage irrelevant, when in fact it relies heavily on voltage. The reasons behind this are discussed later on in section 5.4. Due to this the testing of the full power map cannot be continued but the testing procedure will still be laid out for future reference.

Once the static power has been found the short circuit and transition power also need to be obtained. To find these a similar test is conducted where the frequency is kept stable this time and the voltage is varied. Once again if 3 different frequencies are used then we can obtain the short-circuit power where the voltage is 0 and plot these results on a graph of frequency against power. This will give the voltage irrelevant short-circuit power factor needed for the equation. The transition power is also found using the same graph with plotting the multiplier values of x against voltage. This yields the activity factor and capacitance in one.

From the values obtained the full frequency map can then be plotted to show how the power consumption varies with clock speed.

5. Discussion

5.1 The Need for Higher Voltages with Higher Frequencies

From Figure (14) it can be seen that the higher the frequency gets the voltage also increases linearly. The reason for higher voltages is simple. A CMOS gate can be modelled as a capacitance and when looking at a CMOS gate that is what you will see in measurements. For the CMOS gate to switch states it has to either charge or discharge. For a capacitor to change states from a logical 0 to 1 it has to charge up. Now suppose we can have two supply voltages, one which is significantly larger than the other $V_1 \gg V_2$. The time taken for V_2 to charge the capacitor will be many times greater than the time taken to charge the capacitor with V_1 . The CMOS gate will run into problems if it tries to operate at a frequency higher than the rate at which the capacitor can charge and discharge. Therefore to tackle this problem there has to be an increase in the supply voltage to combat this problem.

Over the years supply voltages have increased to accommodate the need for higher operating frequencies. At the same time the size of transistors has plummeted so that many more transistors can be fit in to the same area. With smaller transistors comes the problem of thinner insulating layers between the gate and sources. This is where the problem of electrons tunnelling through this insulating layer arises. This is one of the major factors of power dissipation in current CPU's. To tackle this problem companies like Intel are no longer going for higher frequencies and supply voltages but are going for a scheme known as many-core. Instead of increasing the frequency anymore and making the problem even worse they are going for a greater level of parallelism with over a 100 cores to a single CPU. This technique takes us back to the beginning of the paper where it is stated that if we split threads along many cores at lower operating frequencies as opposed to a single core at very high frequencies we can reduce the power a lot. With lower supply voltages we can also decrease the effect of tunnelling and therefore have a more efficient CPU. It is also beneficial for mobile devices where cooling of the components is a major factor. With the device not being driven to its maximum capacity there is less of a need for big heat sinks and other devices such as fans which aren't appropriate for mobile devices.

5.2 More Threads Equal More Power

The notion of if you have more threads running this in turn gives a big increase in power consumption is becoming somewhat of a myth with multi-core CPU's. If for a second we ignore Intel's C-states and have a look at the issue of splitting threads across multiple cores we can begin to understand why this is. Modern CPU's do not have a voltage regulator for

each individual core, they operate under a singular master Vcore regulator. This provides the same voltage to each core at a steady voltage. This voltage can be dynamically changed as of when the processor needs to increase frequency. Say we have one thread running on 1 of 8 cores. This thread increases that core up to maximum utilisation and so the frequency is dynamically scaled up to cope with this. To then provide a stable higher frequency the voltage also has to go up as discussed in section 5.1. Now there is the problem that because there is only one master Vcore regulator all the other 7 cores voltage also rises to the same value. This is where the problem arises. Although the other 7 cores are processing no information, they are still on and using up voltage. The power consumption of the other 7 cores will not be as high as the core running the thread but they will still be using up power. To understand this better a look at the power consumption equation for a CPU shown in section 2.1 will be looked at.

$$P_{CPU} = (m * V) + (\alpha * E_{short-circuit} * f) + (\alpha \frac{1}{2} CV^2 f) \quad (6)$$

The activity factor α associated with the dynamic is what makes the power consumption of the other 7 cores less than the one running the thread. This is because the activity factor is a scale of how often the transistors are actually switching every clock cycle. Due to the fact that the idle cores are not processing any data they therefore have an activity factor or around 0 and almost get rid of the dynamic power and the short circuit power. But as stated in section 2.1, the static power can make up a huge proportion of the total power consumption in modern CPU's. This power is therefore still present and a large factor in the total power consumption of the CPU. If more threads are added to each of the remaining 7 idle cores then only the dynamic and short circuit power will be adding on top of the previous power consumption. This therefore means that if we add another 7 threads to the original first threads with one on each core the power consumption does not increase significantly.

Now to tackle this problem Intel has introduced C-states, power gating and power aware interrupt routing (PAIR). These are states that are activated when a core is idle and the CPU wants to lower its power usage. There are many states that can be selected from but the general idea is that if a core is idle turn that core off and therefore reduce the static power consumption and also reduce the overall capacitance of the circuit. Previously these states were not commonly chosen by the user though because to change these states one has to go into the BIOS and select which C state they want their computer to operate under. For the average run of the mill end user this is not something they are familiar with and probably will

never look at. But with Intel's new architecture these changes can be done by selecting whether or not you want your computer in performance or energy saving mode

Another problem associated with this is that the other cores may not ever become idle, due to the fact that modern CPU's will try and split tasks across many cores. Assume we have the same example of 1 thread on 1 of the cores utilising it to its maximum capacity. If background tasks are now needed to be performed then the CPU will see that it is better to run them on the other cores as opposed to slowing down the speed at which the main thread is running at. This in turn means that those cores are no longer idle and therefore are turned on and so are using up energy. Some companies are now employing a technique of having a separate slower running core to handle background tasks so that these faster higher voltage cores are not turning on for any reason and are free to be used by high priority tasks. Intel's PAIR architecture now handles this as well by routing these interrupts to the main core when in energy saving mode at the price of slowing down the main thread.

These are just some of the problems that the CPU's encounter when trying to tackle power consumption and they are a major factor now when designing not only the architecture they are made from but also the programs that use the architecture.

5.3 Validity of Shunt Resistor Setup and Better Testing Methods

The shunt resistor setup as described in Figure (5) may not be the best setup for trying to estimate the power consumption. It is assumed in this paper that the shunt resistor is in series with the CPU in a simple series circuit but this may not be the case. Firstly because the CPU is essentially a big variable resistor it must have two grounds. One leading to the shunt resistor is a definite but the grounds of all the voltage regulators inside the CPU do not go through the shunt resistor for definite. There could be multiple grounds which don't all lead back to the CPU power source. The other trouble is that the CPU power supply we have tapped off may not be the CPU's only source of power; it could be drawing power from several different sources. Some people speculate that the EATX12V supply may only be for surplus energy requirements and other parts of the motherboard may supply the CPU with more power.

Therefore the current that is calculate to be going through the shunt resistor may not be a true representation off how much current the CPU is actually drawing and so it cannot be accurately measured how much power the CPU is using. We can however try and get similar power characteristic curves and see if they match up with the ones Likwid provides us with.

Some other methods for testing have been proposed by several people online. One way in which the power could be tested is to test the power consumption from the plug socket. This will however also include the system power. To get around this problem the system is locked into as stable state that is physically possible. This is done in the BIOS by taking out any automatic offset voltages and speeds. The DRAM is locked to certain voltages and speeds as is the graphics card. Several other options like Intel's SpeedStep and C-states are disabled and the system voltages are locked to specific values. With this in mind the tests can be carried out and then the system power consumption can be subtracted from these tests. This can give very nice results with overall power consumption models. This approach is attempted with the shunt resistor method but failed with the fact that the shunt resistor does not give true power consumption and has a thermal drift.

5.4 How Accurate are Intel's RAPL Counters

Intel first introduced the RAPL counters as a way of making better decisions on when to use turbo boost and also to set power straps to certain bits of hardware like the CPU and DRAM. This feature was introduced in Intel's Sandy Bridge product line. Since then programmers have been using it to see how much energy/power their programs are using and also to set limitations on how much power the system is allowed to use. Intel's RAPL counters work by using specific MSR's, this is to try and get away from a completely modelled system and have some feedback as to what these devices are actually doing.

Whilst testing though the validity of RAPL counters are brought into question. It would appear from testing that the counters still rely on a model of sorts. This model would appear to be Intel's SpeedStep technology. The power curves when the system is running at its intended voltages for the selected frequencies appear to be correct from the RAPL counters. But once the voltage starts to be locked to specific values in the BIOS and the frequency is then changed it appears as though the actual voltage has no effect on the results. This is down to the fact that it is not an actual analogue power meter but is in fact just seeing how fast bits are going through a device and then looking at a predefined SpeedStep table for what voltage it must be running at. However this may not be the correct voltage of the system so it is not a complete power estimation device. This especially becomes a problem when the end user is not using stock voltages on their system and with most programmers being familiar with the BIOS it can be assumed they will want the most out their hardware and overclock the CPU. This is where their energy reading s will not be accurate.

5.5 The Importance of Hyper-Threading

Hyper-threading can be seen to dramatically decrease power consumption. Intel claims that a Hyper-thread capable core uses less than 5% more area on the die than a non Hyper-thread capable core [12]. With the results obtained in the experiments it is clear that the Hyper-Thread core matches the performance and the power consumption when splitting across two real cores. This could lead to a great deal of power saving due to the fact that there is now more cores to split threads across. Therefore if there are more than say 4 threads being processed on a 4-core processor it doesn't have to load more than one thread onto each core it can continue to spread the threads out.

5.6 Future Work

If the time had permitted then a couple of more tests could have been carried out. A new way of testing the total power curve of the CPU without using Likwid could have been done. This is discussed in section 5.3. The type of tests that are done could have been expanded to include access to the hard drive and also the GPU. This could have tested to see if the speed it accesses the hard drive increases power efficiency or not.

These tests then could have been related to real world problems such as decoding a video or streaming a video from a website. These use many attributes of a system but are hard to get concise results from as there is no standard in testing streaming and playback of a video from a website.

6. Critical Analysis of the Project and Societal Impact

6.1 Critical Analysis of the Project

From a critical point of view the project went fairly well. Most of the goals were met but a few were not able to be completed. The main test to be completed was the energy efficiency of splitting threads across multiple cores as opposed to a single core. There was one test that wasn't completed. This test was the full frequency power curve of the processor.

The design specifications were met but due to the fact it needed very specific parts it took a very long time to get the parts chosen and ordered in. This was one of the major holdups of the dissertation. Not all of the parts arrived until after the Christmas holidays due to the fact that some of the parts kept on going out of stock. This greatly reduced the time available for testing. Because of this not all of the tests were completed and so the full frequency power curve was not finished.

The results that were collected offered very good evidence to some of the theories covered in the literature review. This includes energy efficiency when splitting threads across cores to be around 50%. The results presented in section 4.4 clearly show that this is the case.

One of the reasons that the full power curve could not be completed is down to the fact that the Intel RAPL counters did not act as they should under certain conditions. This became apparent in section 4.5 where the power consumed did not change with an increase in voltage. This does not follow the power consumption equations stated in section 2.1. Because of this it could not be used for this test.

If more time had been available then a second method of testing for the full power curve could have been implemented. This could have been done by measuring the power consumed as a whole at the plug socket and then completing the same test procedure except taking away the system power that would also be included. This has been shown to work in some papers online and so if time permitted this could have been verified as to whether or not it works.

With regard to validity of the results obtained this is still up for consideration. The setup of having the shunt resistor in series with the CPU is theoretically the correct way to test power consumption. But it may be the case that not all of the power used by the CPU is going through the shunt resistor and some of the power maybe going to different grounds. Or it may be the case that the CPU can draw power from not only the EATX12V supply but also from other sources on the motherboard if needed. This would affect the results collected and make them slightly less accurate. This could have been tested if more time had been available.

6.2 Societal Impact of the Project

The implications of the results shown in this project are very beneficial to society as a whole. Energy saving is a massive problem that we face at the moment and the work that chip makers are undertaking currently to reduce power consumption in many ways benefits us greatly. This project can be seen to impact society across a broad spectrum of areas some of these are outlined in the following section.

The main aim of the project was to highlight the power consumption of CPU's under different operating conditions and discuss as to whether or not they reduce power consumption in any way. It was shown that splitting threads across multiple cores can achieve better energy efficiency.

With the use of digital electronics and digital data ever more prevalent in today's society there has become the need for faster electronics. This has meant that we have had to increase the size of chips and the frequency at which they operate at. This has led to an increase in power consumption of such devices as explained in section 2.1. The need for these devices to be mobile is ever increasing and with mobile devices comes the problem of not having a constant power supply but instead being supplied by a battery. The need therefore for more energy efficient modules is massive and will help society in being able to process data on the move and not have to charge such devices up as often.

We will start by looking at how this affects our health or how it can help us manage our health. With most of our records of health being digitally stored there is a need to process this data. The data is getting more and more complicated and so the need for faster CPU's to not only obtain this data but also to process it is increasing. This is using up a massive amount of energy with systems usually having to be on all the time. Any advancement in reducing this power consumption will lead to reductions in cost to provide energy to these systems. This can in turn reduce the overheads of the medical practices and then benefit the patient by hopefully providing lower medical bills. This can hopefully in turn provide a better healthcare for society as a whole. A lot of medical devices are also becoming portable so there is a need there to keep energy consumption down to a minimum so that such devices can be used for longer periods of time. This is usually not a problem in 1st world countries where they have multiple copies of such devices and can charge them at will from the mains. A problem arises though if you are not near a constant supply of power for example in 3rd world countries, out at sea or at a mobile camp for example camps such as the army use when fighting wars. The reduction in energy consumption would greatly increase the productivity of medical assistance in these situations with downtime being reduced to a minimum. Another health factor is that many health devices are also being wearable. For example heart rate monitors and after operations sometimes data is needed to be collected about the patient when they're at home. Again if these devices could be charged up less and worn more it would benefit our health and the productivity of the medical system.

With a lot of devices now being portable and using batteries another issue arises. The energy taken to charge up the batteries and to create them is a burden on society. With most energy on the planet being produced by non-green methods it damages our planet more every time we charge such a device up. Therefore if we can reduce this energy consumption we can hopefully tackle the problem of things such as global warming. This is a massive factor at the

moment and is one of the main reasons why companies such as Intel are crating energy efficient modules.

It can be seen that such research can benefit our society's environment and health but it can also affect the wealth of a society as well. With the devices consuming less power things like production and assembly lines can reduce their overheads. This in turn can mean they can make more profit, this can improve the wealth of a society by having companies that can compete for lower prices offered to the end user. Many systems in production lines will be on 24/7 and so even the smallest reduction in energy used can yield massive savings.

7. Conclusions

The final results this paper has outlined carry with them some really surprising outcomes. The fact that Intel's RAPL counters appear not to take voltage into account and most probably use the pre-defined SpeedStep values is surprising. This can lead to false power measurements and therefore can lead to problems. It also hindered the full testing of the CPU's power map and so this could not be taken into account while making this paper.

An important theory proven to be true whilst testing is the energy savings of a multi-core processor. This showed that splitting the threads across multiple cores can reduce the energy by half. This is great for power conscious modules and also still has the option to have a greater performance than a single core. The importance of Intel's PAIR system and power gating system has been shown. The way it reduces the power consumption have been discussed and shown how necessary they are in modern multi-core processors. This is down to the fact that the static power of a CPU now contributes to a large proportion of the total power consumption of the device. With the ability to reduce some of that static power by turning off cores and making sure they're not activated until they are really needed is a massive advancement in multi-core technology.

This paper also highlights the need for programmers to create programs that take advantage of multi thread processing or parallel processing as it is sometimes called. Without programs being made to run with parallel threads there is no need for multi-core processors. This is the real trouble as the technology is here but the programs are lagging behind in the parallelism area. Programs also need to start taking more use of Hyper-Threading, where two cores share resources like the cache. If there are two very similar threads created by a program then sending them to a Hyper-Thread core where they can share resources is a good idea.

8. Acknowledgements

I would like to thank my project supervisor Dr. Fei Xia, he was extremely helpful throughout the whole dissertation providing a lot of back ground theory and helping me with situations when I was stuck. I could have not asked for a better supervisor and I am very grateful for being assigned one so involved with the project. I would also like to thank the PRIME research and the μ Systems groups for their involvement and advice along the way.

9. Works Cited

- [1] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems," *International Journal of Computing Aided Engineering and Technology*, vol. VI, no. 4, pp. 450-459, 2014.
- [2] H. S and M. D, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, Portland, OR, 2007.
- [3] Intel®, "System Programming Guide, Part 1," *64 and IA-32 Architectures*, vol. III, no. A, pp. 14-36, 2011.
- [4] D. Radhakrishnan, "Design of CMOS circuits," *IEE Proceedings G (Circuits, Devices and Systems)*, vol. CXXXVIII, no. 1, pp. 83-90, 1991.
- [5] N. A. T. Kim, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. XXXVI, no. 12, pp. 68-75, 2003.
- [6] C. X. L. C. W. S. M. J. Fang Juan, "Calculation Method of Multi-core Dynamic Reconfigurable Cache Power Consumption," *JDCTA*, vol. VII, no. 9, pp. 157-164, 2013.
- [7] A. K. S. W. Konstantin Moiseev, "Timing-aware power-optimal ordering of signals," *ACM Transactions on Design Automation of Electronic Systems*, vol. XIII, no. 4, p. Article No. 65, 2008.
- [8] nvidia, "The Benefits of Multiple CPU," *Whitepaper*, pp. 12-14, 2010.
- [9] S. P. J. B. P. Ali Khajeh-Saeed, "A Comparison of Multi-Core Processors on Scientific," p. 3, 2011.
- [10] H. Jobling, "IT Pro Portal," 24 April 2012. [Online]. Available: <http://www.itproportal.com/2012/04/24/intel-ivy-bridge-architecture-breakdown/>. [Accessed 29 April 2015].
- [11] K. Ju, "CPU Power Consumption and Results Analysis," Newcastle University, Newcastle, 2014.
- [12] Intel, "Hyper-Threading Technology," *Intel Technology Journal*, vol. VI, no. 1, p. 7, 2002.

10. Appendices

10.1 Table for the graphs in section 4.2

This table is the averaged values collected for power consumption used in Figure (7).

freq (GHz)	Power average (W)
1.2	2.18401
1.4	2.1901
1.6	2.2185
1.7	2.2314
1.9	2.27149
2.1	2.2882
2.3	2.347
2.4	2.3573
2.6	2.4152
2.8	2.4811
3	2.5876
3.2	2.654
3.3	2.6962
3.5	2.8045
3.7	2.9156

10.2 Table for the graph in section 4.3

This table of results is used to make Figure (14).

frequency (GHz)	core voltage (V)
1.2	0.8
1.4	0.82
1.6	0.84
1.7	0.86
1.9	0.88
2.1	0.9
2.3	0.92
2.4	0.93
2.6	0.95
2.8	0.97
3	1
3.2	1.02
3.3	1.04
3.5	1.06
3.7	1.07

10.3 Table for power consumption tests in section 4.4

The results collected by the Agilent Data Logger 3 which are then averaged for each frequency over the time taken to do the test are put in the following table. This table is then used to create the graphs seen in section 4.4.

frequency (GHz)	Time single core (s)	Time hyperthread core (s)	Time dual core (s)	likwid power single core (W)	likwid power hyperthread core (W)	likwid Energy single core (J)	likwid Energy hyperthread core (J)	likwid Energy dual core (J)	likwid efficiency same frequency	performance increase	Shunt power single core (W)	Shunt power hyperthread cores (W)	Shunt power dual core (W)	Shunt energy single core (J)	Shunt energy hyperthread core (J)	Shunt energy dual core (J)	Shunt efficiency same frequency
1.2	141.1255	70.29255	69.81405	13.8666	15.1891	15.167	1955.520296	1067.680571	1058.868966	0.541477221	49.80852156	1.951675204	1.979588945	275.4812365	139.1593201	137.988741	0.500991619
1.3																	
1.4	121.05705	60.16445	59.8298	14.7542	16.2847	16.5068	1786.099927	979.7600189	987.5985426	0.552935772	49.69925337	1.964169996	2.003958579	237.7766254	120.5670657	119.3964841	0.50213718
1.5																	
1.6	105.83955	52.87865	52.3426	15.623	17.6616	17.4902	1653.53129	933.9250972	915.4825425	0.55365299	49.96133298	1.978808913	2.031398233	209.4362449	107.418025	106.0273397	0.506251149
1.7	99.6758	49.7426	49.28445	16.1633	18.1767	18.6605	1611.089858	904.1563174	919.2929692	0.570607074	49.90493003	1.988798174	2.044654627	198.2350491	101.7064372	100.738282	0.508178693
1.8																	
1.9	89.16675	44.35005	44.0755	17.3115	19.7134	19.6901	1543.610193	874.2902757	867.8913673	0.56247756	49.73832735	2.019640497	2.076160393	180.0847793	92.0781722	91.35465597	0.507288602
2																	
2.1	80.66235	40.09635	39.87665	18.5528	21.2203	21.1363	1496.512447	850.8565759	842.8448374	0.563306032	49.70887905	2.024122714	2.071013889	163.2704948	83.04006567	82.38743659	0.504607012
2.2																	
2.3	74.01335	36.6274	36.4063	19.6926	22.9171	23.3717	1457.515296	839.3937885	850.8771217	0.583786067	49.48755866	2.060650026	2.093169588	152.5156116	76.66735904	75.49848312	0.495021345
2.4	71.3979	35.1127	34.88845	20.3373	23.9426	25.0181	1452.040512	840.893331	872.8427309	0.60114586	49.7889742	2.06774538	2.103213666	147.6326779	73.84951153	73.01408384	0.49456587
2.5																	
2.6	65.184	32.97355	32.2018	21.7101	25.6829	25.5537	1415.151158	846.8563873	822.8751567	0.581475083	50.58534303	2.111224966	2.141244511	137.6180843	70.60443296	68.70105045	0.49921528
2.7																	
2.8	61.15205	30.0703	29.9052	23.4127	27.8998	28.2617	1431.734601	836.2490289	845.1717998	0.59031317	49.17900401	2.113166437	2.17866798	129.2244596	65.48910799	64.72918897	0.500905085
2.9																	
3	57.4164	28.15615	28.02105	25.0109	30.6953	31.7672	1436.055839	864.2614711	880.1510296	0.619866354	49.03851513	2.143781808	2.2055805	123.0882338	62.1006554	61.22019538	0.497368381
3.1																	
3.2	53.32285	26.27355	26.3117	26.5977	32.2554	32.0646	1418.265167	847.4638647	843.6741358	0.594863468	49.27259139	2.180101527	2.237649167	116.2492267	58.79098727	58.41797554	0.502523562
3.3	51.3048	25.7358	25.6116	27.4824	33.3131	34.2995	1409.979086	857.339279	878.4650742	0.62304139	50.16257789	2.19222457	2.25989257	112.4715347	58.1601432	57.6884052	0.512648295
3.4																	
3.5	49.11575	24.0305	23.9186	29.6215	35.5635	36.0173	1454.882189	863.9805818	861.4833918	0.59213275	48.92626092	2.229455866	2.293768497	109.5013969	55.12040386	54.58242724	0.498463286
3.6																	
3.7	45.82655	22.8052	22.6273	31.7652	40.0809	39.2753	1452.939933	912.9128807	888.693957	0.61162261	49.7641651	2.262759692	2.353847282	103.6944702	53.67958202	52.79706492	0.509159889
3.8																	
3.9																	
4																	

