BTECH 451

Parallella Epiphany

Mong-Fan Wang
6916087

ABSTRACT

This report is a summary of my progress in the fourth year Bachelor of Technology project. I am working with a senior engineer in Compucon New Zealand, attempting to further the research and development on a relatively new technology named the Epiphany.

Parallel computing is a type of computation where there are many calculations carried out simultaneously. Unlike traditional computing where one problem generally gets solved with one processor, parallel computing takes the problem and divide them into smaller ones if possible, and they get processed at the same time.

Epiphany is a coprocessor with extremely low power consumption, this allows for a very high performance per watt when utilised in parallel computing. My project will verify the claims of the emerging technology, while also attempt in developing some applications for it.

TABLE OF CONTENTS

COMPANY

Compucon New Zealand is a wholly New Zealand owned company since April 2011, that manufactures computer systems and has excellent reputation and quality in the reseller and user communities.

Partnering with world class component manufacturers, Compucon offers high quality and reliable computer system builds.

Compucon New Zealand also specialises in high performance parallel computing and this will be the main focus of this project.

The company has had many University of Auckland students since 2002 completing projects with them in the Bachelor of Technology program.

PROJECT BRIEF

PURPOSE

The aim of this project is to verify the claims of the Epiphany manufacturer and develop a number of applications to drive this new technology. The programs will be coded in the C or C++ language, as these are low level languages, they communicate with the chip much better than a high level language like Python.

PEOPLE

TN Chan is the General Manager of Compucon New Zealand, as well as the full supervisor of this project. David Fielder is the Senior Engineer of the company who assists me and provides hands-on guidance in the Compucon House one day a week.

HARDWARE & BACKGROUND

EPIPHANY

A co-processor manufactured by Parallela, the particular model I am working with has 16 high performance RISC CPU cores, programmable with both C/C++ and OpenCL. Advantages include very low wattage for power consumption and being very flexible in terms of scalability.

PARALLELLA

The co-processor runs on the Parallela board which is a credit card sized board, includes a Gigabit Ethernet connection, HDMI port and 1 GB of SDRAM.

EPIPHANY ARCHITECTURE

As the Epiphany is a co-processor, it cannot do everything that a normal CPU can. Instead, it is a simplified processor that carries out specialised tasks. This is a disadvantage of the Epiphany, however, it is also the advantage at the same time as being specific makes it much more energy efficient than a standard CPU. Memory on the Epiphany uses little-endian. Doubles are not supported.

THREADING

- Multi-Threading improving performance
- CPU utilisation is better
- IO latency hiding

THREADED MPI

Threaded MPI is the go to architecture for the Parallella Epiphany. The fully divergent RISC cores allow high performance with inter-core data movement and maximise data re-use.

Brown Deer Technology claims that the programming is very easy, performance is great and the libraries are readily available. It is also only going to get easier as technology progresses. Part of my project is also to verify the claims of Brown Deer.

The power efficiency of the Epiphany rivals many other processes in the market today and threaded MPI works perfectly aligned with that goal.

COPRTHR SDK

This term stands for the CO-PRocessing THReads. It is a SDK that provides libraries and tools for developers that are developing multi-core applications. It provides support for the Parallella in OpenCL and STDCL for the Epiphany co-processor.

STDCL

This is a portable API for targeting compute offload accelerators and co-processors.

EPIPHANY LAYOUT

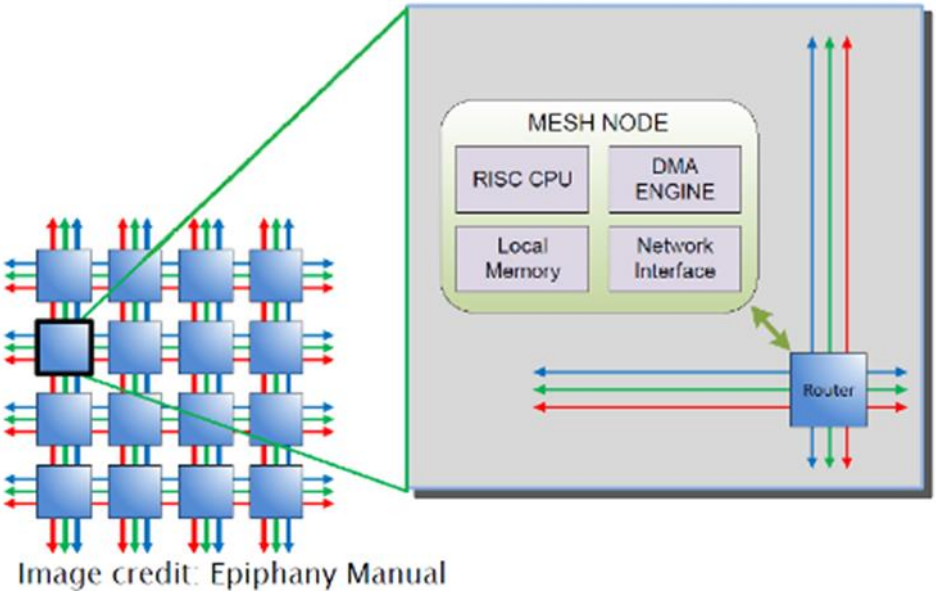


Image credit: Epiphany Manual

DEVICE SETUP

PARALLELLA SETUP

The following hardware are required:

- Parallella Board
- 4-Port Powered USB Hub
- 8 GB Micro-SD Card with an Adapter
- Micro-USB to USB (Female) Cable
- Micro-HDMI to HDMI (Female) Cable
- Crossover Ethernet Cable

The following software is required:

- https://www.parallella.org/create-sdcard/
    - The Manufacturer has included 4 versions of the Ubuntu image:
        - Desktop Headless
        - Desktop with Display
        - Kickstarter Headless
        - Kickstarter with Display

      For the board I am working with, I will be using both the Desktop Headless and with Display.

SD CARD FORMATTING

The SD card houses the Operating System for the Parallella.

Use the SDFormatter to fully erase the SD card. Now use a Win32 Disk Imager program to load the file containing the Parallella image onto the SD card. It may appear that there are no files in the SD card from Windows Explorer, however, this is normal. Safely eject the SD card and pop it into the Parallella.

EXPANDING IMAGE

This command shows that only a small portion of the SD Card is available for use:

```
df -h
```

By entering the following series of commands, the image will be expanded so that the entire SD card's storage size can be utilised correctly.

```
dmesg | grep "root"
root=/dev/mmcblk0p2
```

`/dev/mmcblk0p2` is the root partition, expand this by entering:

```
fdisk /dev/mmcblk0
```

Enter 'm' for help. Delete partition 2 (root partition), then create a new partition 2. Enter 'd' followed by '2' to delete the root partition. Then 'n' followed by 'p' and 2'' to create a new partition 2. For the first and last sector, select default. Enter 'p' to confirm and write it to disk with 'w'.

Machine is then rebooted with:

```
sudo shutdown -r now
```

After reboot, enter:

```
resize2fs /dev/mmcblk0p2
```

This ensures the resize.

SSH SETUP

WINDOWS

Microsoft Windows does not have built in SSH, this means PuTTY for Windows is used. It can be downloaded from:

- http://www.putty.org/

LINUX

In Linux, SSH is built in the Terminal.

NETWORK CONNECTION

There are two ways to connect to the Parallella board. Finding the IP address assigned to the machine, or assigning a static address to it.

DYNAMIC IP

Find the IP address of the Parallella board by using any sort of network tool that displays all devices connected in a Local Area Network.

STATIC IP

Edit the file "/etc/network/interfaces" to contain the following lines:

```
auto eth0
iface eth0 inet static
```

```
    address 10.0.0.3/8
    up route add 10.0.0.2 dev eth0
```

Edit the file "/etc/hostname" and assign the board a hostname, then edit \etc\hosts and add the following line:

```
10.0.0.2 [Hostname]
```

Reboot the board to allow the operating system to process the changes.

Running ifconfig in the terminal should confirm that your board's IP address is now 10.0.0.3.

COPRTHR SETUP

WINDOWS

Run the Windows installer from https://github.com/browndeer/coprthr (libstdcl-1.4.0-win7-install.msi) and set the appropriate paths to use the headers and library.

LINUX

Pre-requistites:

- Linux Ubuntu
- libelf-0.8.13.tar.gz (www.mr511.de/software/libelf-0.8.13.tar.gz)
- libevent-2.0.18-stable.tar.gz (github.com/downloads/libevent/libevent/libevent-2.0.18-stable.tar.gz)
- libconfig-1.4.8.tar.gz (www.hyperrealm.com/libconfig/libconfig-1.4.8.tar.gz)
- m4-1.4.16.tar.gz (http://ftp.gnu.org/gnu/m4/)
- flex-2.5.35.tar.gz (http://flex.sourceforge.net/)
- bison-2.5.tar.gz (http://ftp.gnu.org/gnu/bison/)

Pre-compiled Package:

- coprthr-1.5.0-rc2-parallella.tgz

The libraries are unpacked by entering the following commands:

```
./configure
sudo make install
```

Unpacking the file will produce a directory browndeer/.

Enter following commands to remove previous installations as well as installing the new version:

```
sudo ./browndeer/uninstall_coprthr_parallella.sh
```

```
sudo ./browndeer/install_coprthr_parallella.sh
```

Finally, add the following environmental variables to PATH:

```
export PATH=/usr/local/browndeer/bin:$PATH
export
LD_LIBRARY_PATH=/usr/local/browndeer/lib:/usr/local/lib:$LD_L
IBRARY_PATH
```

PROGRAMMING

The Parallela uses a host/device structure, meaning every application needs a corresponding program for each side.

While the programs are separate, all files are created and stored on the host (in this case the ARM chip)

HELLO WORLD EXAMPLE

DEVICE PROGRAM - E_HELLO_WORLD.C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "e_lib.h"

int main(void) {
    const char         ShmName[] = "hello_shm";
    const char         Msg[] = "Hello World from core
0x%03x!";
    char               buf[256] = { 0 };
    e_coreid_t         coreid;
    e_memseg_t         emem;
    unsigned           my_row;
    unsigned           my_col;


    coreid = e_get_coreid();
    e_coords_from_coreid(coreid, &my_row, &my_col);

    if ( E_OK != e_shm_attach(&emem, ShmName) ) {
        return EXIT_FAILURE;
    }

    snprintf(buf, sizeof(buf), Msg, coreid);

    if ( emem.size >= strlen(buf) + 1 ) {
        e_write((void*)&emem, buf, my_row, my_col, NULL,
strlen(buf) + 1);
    } else {
        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}
```

HOST PROGRAM - HELLO_WORLD.C

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <e-hal.h>

const unsigned ShmSize = 128;
const char ShmName[] = "hello_shm";
const unsigned SeqLen = 20;

int main(int argc, char *argv[])
{
    unsigned row, col, coreid, i;
    e_platform_t platform;
    e_epiphany_t dev;
    e_mem_t    mbuf;
    int rc;

    srand(1);

    e_set_loader_verbosity(H_D0);
    e_set_host_verbosity(H_D0);

    e_init(NULL);
    e_reset_system();
    e_get_platform_info(&platform);

    rc = e_shm_alloc(&mbuf, ShmName, ShmSize);
    if (rc != E_OK)
        rc = e_shm_attach(&mbuf, ShmName);

    if (rc != E_OK) {
        fprintf(stderr, "Failed to allocate shared memory.
Error is %s\n",
                strerror(errno));
        return EXIT_FAILURE;
    }

    for (i=0; i<SeqLen; i++)
    {
        char buf[ShmSize];

        row = rand() % platform.rows;
        col = rand() % platform.cols;
        coreid = (row + platform.row) * 64 + col +
platform.col;
        printf("%3d: Message from eCore 0x%03x (%2d,%2d): ",
i, coreid, row, col);
```

```
        e_open(&dev, row, col, 1, 1);
        e_reset_group(&dev);

        if ( E_OK != e_load("e_hello_world.elf", &dev, 0, 0,
E_TRUE) ) {
            fprintf(stderr, "Failed to load
e_hello_world.elf\n");
            return EXIT_FAILURE;
        }

        usleep(10000);

        e_read(&mbuf, 0, 0, 0, buf, ShmSize);

        printf("\"%s\"\n", buf);
        e_close(&dev);
    }

    e_shm_release(ShmName);
    e_finalize();

    return 0;
}
```

```
linaro@linaro-nano: ~/epiphany-examples/apps/hello-world
File  Edit  Tabs  Help
linaro@linaro-nano:~/epiphany-examples/apps/hello-world$ ./build.sh
linaro@linaro-nano:~/epiphany-examples/apps/hello-world$ ./run.sh
  0: Message from eCore 0x8ca ( 3, 2): "Hello World from core 0x8ca!"
  1: Message from eCore 0x84b ( 1, 3): "Hello World from core 0x84b!"
  2: Message from eCore 0x84b ( 1, 3): "Hello World from core 0x84b!"
  3: Message from eCore 0x888 ( 2, 0): "Hello World from core 0x888!"
  4: Message from eCore 0x849 ( 1, 1): "Hello World from core 0x849!"
  5: Message from eCore 0x88b ( 2, 3): "Hello World from core 0x88b!"
  6: Message from eCore 0x88b ( 2, 3): "Hello World from core 0x88b!"
  7: Message from eCore 0x8ca ( 3, 2): "Hello World from core 0x8ca!"
  8: Message from eCore 0x80a ( 0, 2): "Hello World from core 0x80a!"
  9: Message from eCore 0x808 ( 0, 0): "Hello World from core 0x808!"
 10: Message from eCore 0x8c8 ( 3, 0): "Hello World from core 0x8c8!"
 11: Message from eCore 0x8c9 ( 3, 1): "Hello World from core 0x8c9!"
 12: Message from eCore 0x88a ( 2, 2): "Hello World from core 0x88a!"
 13: Message from eCore 0x88b ( 2, 3): "Hello World from core 0x88b!"
 14: Message from eCore 0x8cb ( 3, 3): "Hello World from core 0x8cb!"
 15: Message from eCore 0x84a ( 1, 2): "Hello World from core 0x84a!"
 16: Message from eCore 0x88a ( 2, 2): "Hello World from core 0x88a!"
 17: Message from eCore 0x84b ( 1, 3): "Hello World from core 0x84b!"
 18: Message from eCore 0x848 ( 1, 0): "Hello World from core 0x848!"
 19: Message from eCore 0x8ca ( 3, 2): "Hello World from core 0x8ca!"
linaro@linaro-nano:~/epiphany-examples/apps/hello-world$
```

DOT PRODUCT EXAMPLE

DEVICE PROGRAM - E_TASK.C

```
#include <stdio.h>
#include <stdlib.h>
#include "e-lib.h"
#include "common.h"

int main(void)
{
  unsigned *a, *b, *c, *d;
  int i;

  a    = (unsigned *) 0x2000;//Address of a matrix
(transfered here by host)
  b    = (unsigned *) 0x4000;//Address of b matrix
(transfered here by host)
  c    = (unsigned *) 0x6000;//Result
  d    = (unsigned *) 0x7000;//Done

  //Clear Sum
  (*(c))=0x0;

  //Sum of product calculation
  for (i=0; i<N/CORES; i++){
    (*(c)) += a[i] * b[i];
  }

  //Raising "done" flag
  (*(d)) = 0x00000001;

  //Put core in idle state
  __asm__ __volatile__("idle");
}
```

HOST PROGRAM - MAIN.C

```
#include <stdlib.h>
#include <stdio.h>
#include <e-hal.h>
#include "common.h"

#define RESULT 85344 //recognize /Sum_{i=0}^{n-1} i^2 =
\frac{N(N-1)(2N-1)}{6}

int main(int argc, char *argv[]){
  e_platform_t platform;
  e_epiphany_t dev;

  int a[N], b[N], c[CORES];
```

```
  int done[CORES],all_done;
  int sop;
  int i,j;
  int sections = N/CORES; //assumes N % CORES = 0
  unsigned clr = 0;

  //Calculation being done
  printf("Calculating sum of products of two integer vectors
of length %d initalized from (0..%d) using %d Cores.\n",N,N-
1,CORES);
  printf(".......\n");

  //Initalize Epiphany device
  e_init(NULL);
  e_reset_system();
//reset Epiphany
  e_get_platform_info(&platform);
  e_open(&dev, 0, 0, platform.rows, platform.cols); //open
all cores

  //Initialize a/b input vectors on host side
  for (i=0; i<N; i++){
    a[i] = i;
    b[i] = i;
  }

  //Load program to cores
  e_load_group("e_task.elf", &dev, 0, 0, platform.rows,
platform.cols, E_FALSE);

  //1. Copy data (N/CORE points) from host to Epiphany local
memory
  //2. Clear the "done" flag for every core
  for (i=0; i<platform.rows; i++){
    for (j=0; j<platform.cols;j++){
      e_write(&dev, i, j, 0x2000,
&a[(i*platform.cols+j)*sections], sections*sizeof(int));
      e_write(&dev, i, j, 0x4000,
&b[(i*platform.cols+j)*sections], sections*sizeof(int));
      e_write(&dev, i, j, 0x7000, &clr, sizeof(clr));
    }
  }

  // start cores
  e_start_group(&dev);

  //Check if all cores are done
  while(1){
    all_done=0;
    for (i=0; i<platform.rows; i++){
      for (j=0; j<platform.cols;j++){
```

```
    e_read(&dev, i, j, 0x7000, &done[i*platform.cols+j],
sizeof(int));
    all_done+=done[i*platform.cols+j];
      }
    }
    if(all_done==CORES){
      break;
    }
  }

  //Copy all Epiphany results to host memory space
  for (i=0; i<platform.rows; i++){
      for (j=0; j<platform.cols;j++){
    e_read(&dev, i, j, 0x6000, &c[i*platform.cols+j],
sizeof(int));
      }
  }

  //Calculates final sum-of-product using Epiphany results as
inputs
  sop=0;
  for (i=0; i<CORES; i++){
    sop += c[i];
  }

  //Print out result
  printf("Sum of Product Is %d!\n",sop);
  //Close down Epiphany device
  e_close(&dev);
  e_finalize();

  if(sop==RESULT){
    return EXIT_SUCCESS;
  }
  else{
    return EXIT_FAILURE;
  }
}
```

EPIPHANY BSP - HELLO WORLD

DEVICE PROGRAM - E_CORE_HELLO.C

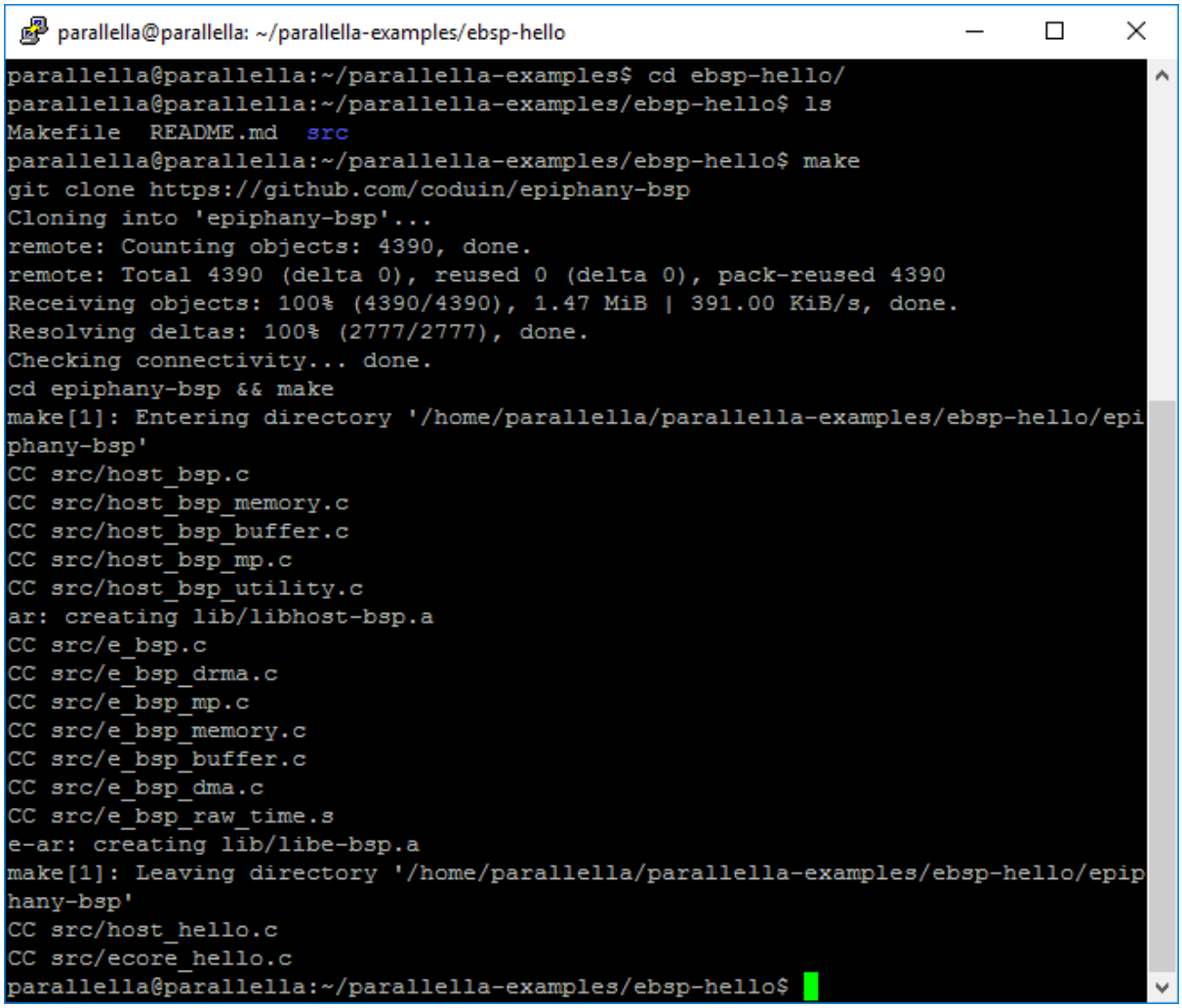
```
#include <e_bsp.h>

int main()
{
    bsp_begin();

    int n = bsp_nprocs();
    int p = bsp_pid();
```

```
    ebsp_message("Hello world from core %d/%d", p, n);

    bsp_end();

    return 0;
}
```

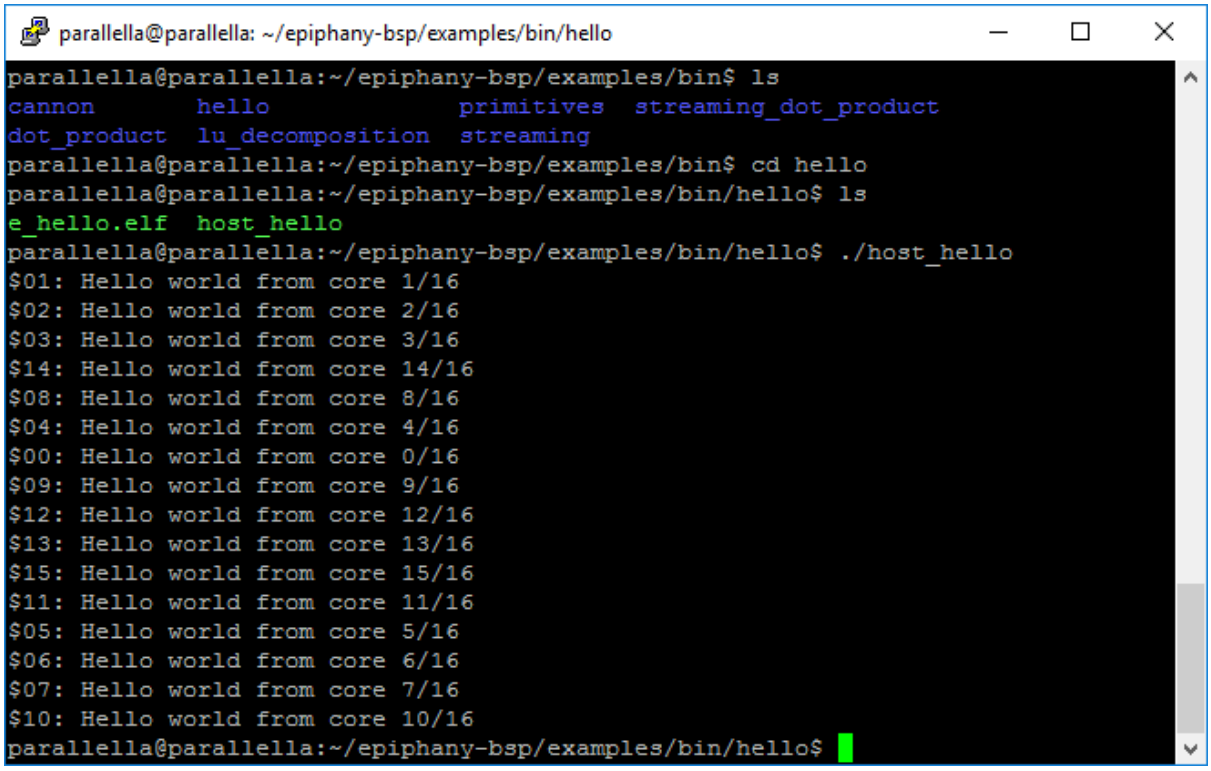
HOST PROGRAM - HOST_HELLO.C

```
#include <host_bsp.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    bsp_init("ecore_hello.srec", argc, argv);

    bsp_begin(bsp_nprocs());

    ebsp_spmd();

    bsp_end();

    return 0;
}
```

RUNNING MAKEFILE

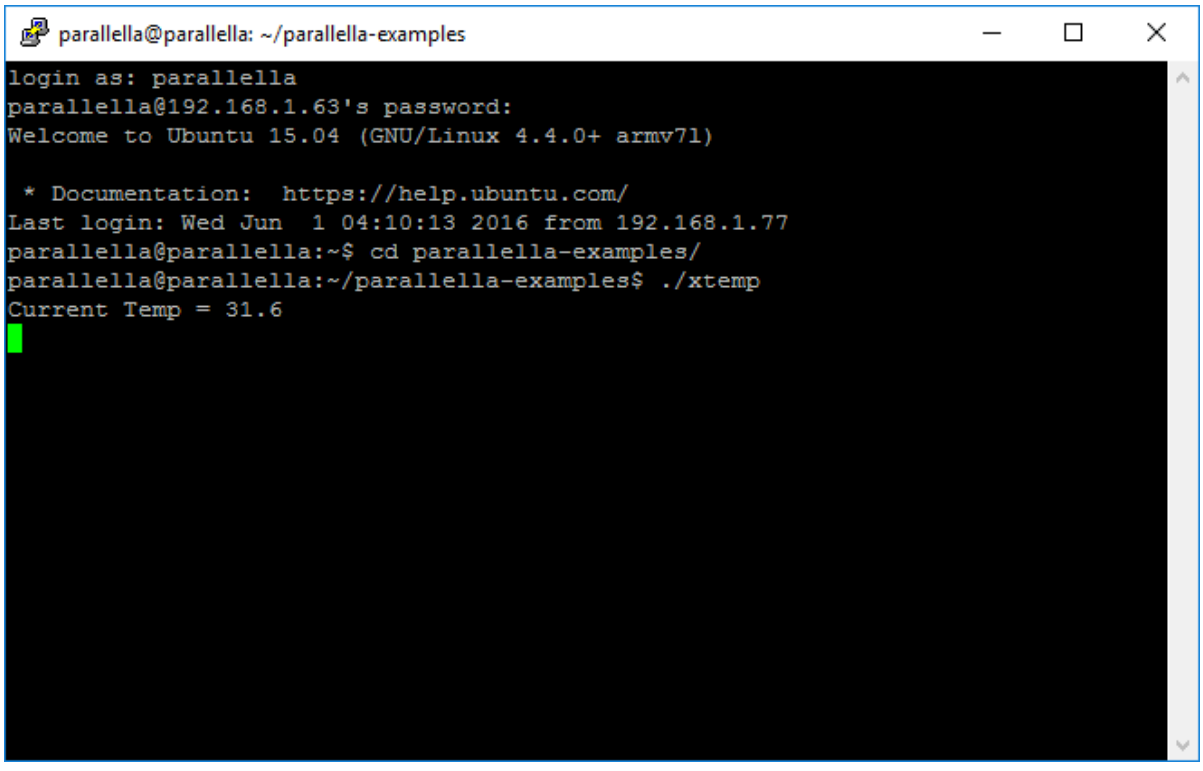


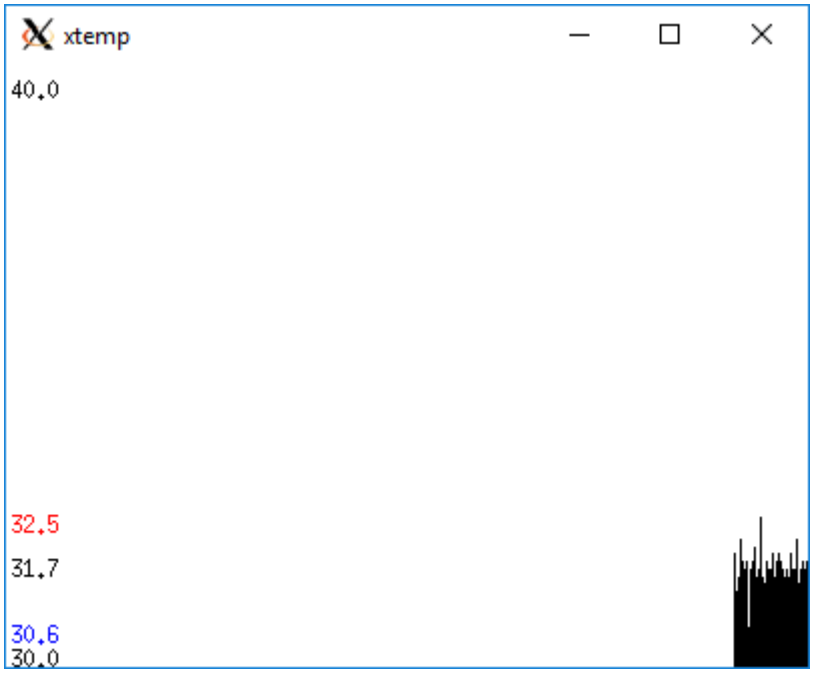
PROGRAM OUTPUT



XTEMP

The xTemp utility is a program under the Parallella Utility package, where the temperature of the board can be visualised. With SSH access, X11 forwarding is needed to see the graphical output on the remote connection.





PARALLELA EPIPHANY WORKSPACE CREATION

The following commands in the Terminal or PuTTY SSH connection will allow the workspace creation. This will allow for easier programming on the Epiphany.

```
cd ~/Downloads
wget
ftp://ftp.parallella.org/esdk/old/esdk.5.13.07.10_linux_x86_64_armv7l.tgz
sudo mkdir -p /opt/adapteva
sudo mv esdk.5.13.07.10_linux_x86_64_armv7l.tgz /opt/adapteva
cd /opt/adapteva
sudo tar xvf esdk.5.13.07.10_linux_x86_64_armv7l.tgz
sudo ln -sTf esdk.5.13.07.10 /opt/adapteva/esdk
sudo apt-get install libmpfr-dev libgmp3-dev libmpc-dev
openjdk-6-jre tcsh csh g++ -y
sudo nano /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin:/usr/games:/usr/local/games:/opt/adapteva/esdk/tools/e-
gnu/bin"
EPIPHANY_HOME="/opt/adapteva/esdk"
LD_LIBRARY_PATH="/usr/lib:/usr/lib/x86_64-linux-gnu"
cd /usr/lib/x86_64-linux-gnu
sudo cp libmpc.so libmpc.so.2
sudo ldconfig
sudo cp libmpfr.so libmpfr.so.1
sudo cp libgmp.so libgmp.so.3
sudo nano /opt/adapteva/esdk/tools/host/bin/echo-process
Save empty file
sudo chmod 777 /opt/adapteva/esdk/tools/host/bin/echo-process
```

```
e-eclipse
```

Create a workspace and name the project. Since the Epiphany is 16 core in this case, we must have the settings of:

Number of rows = 4
Number of columns = 4
Row number in first core = 32
Column number of first core = 8

This creates a master project for all projects that will be created.

To program the Epiphany, we now change the host name to the Epiphany IP address or host name in our network. The 'stop at main' checkbox should be unticked and both 'Resume' and 'Verbose mode' should be ticked.

The final step is to right click the first core project and complete the following:

C/C++ Build → Settings → Epiphany Linker → Linker Description File

Change Select LDF to: ${EPIPHANY_HOME}/bsps/current/legacy.ldf

In the Epiphany Linker, add e-lib to the libraries, then apply these settings to all of the projects. A dialog should pop up showing success messages if completed correctly.

EPIPHANY PROGRAM EXECUTION

Type e-server in the Terminal.
The Epiphany listens on the port 51000 by default.

CUURENT & FUTURE WORK

As outlined, the first half of the project is research based, with the second half being much more practical. I have already moved towards the practical side of things as I attend seminars and do in house experiments with the senior Engineer. Currently I am in the middle of experimenting with Threaded MPI from Brown Deer Technology and looking at Bulk Sync Parallel computing as well.

Future work will include finishing up the experimentation and move on to utilising some of the core functions to develop applications that may be of use to the Epiphany architecture.

REFERENCES

Compucon.co.nz. (2009). Compucon Computers NZ - Quality Servers and Workstations - Company Profile. [online] Available at: http://www.compucon.co.nz/content/view/27/242/.

Suzannejmatthews.github.io. (2016). Technical Musings : Parallella Setup Tutorial. [online] Available at: http://suzannejmatthews.github.io/2015/05/29/setting-up-your-parallella/.

Adapteva, (2011). Epiphany Architecture Reference. [online] Available at: http://www.adapteva.com/docs/epiphany_arch_ref.pdf.

Adapteva, (2016). Epiphany Datasheet [online] Available at: http://adapteva.com/docs/e16g301_datasheet.pdf.

Brown Deer Tecnology, (2013) COPRTHR API Reference. [online] Available at: http://www.browndeertechnology.com/docs/coprthr_api_ref.pdf.