Project 2 Report

Christina Pavlopoulou cpavl001@ucr.edu

Andres Calderon acald013@ucr.edu

May 23, 2016

1 DVFS controller implementation

In order to implement the requested controller we start capturing the user parameters for the number of cycles per interval (DVFSInterval) and the target power (DVFSTargetPower) in the file `sim-ourorder.c`. We did that using the `opt_reg_*` functions already provided by the code (listing 1). With these values we implement the controller as it is shown in listing 3. Listing 2 shows the variables we used.

In addition, we did modifications on files `power.c` and `power.h` to adapt the formula for power factor according to the changes in VSF and FSF values (listing 4). Also, we capture the total power per cycle for each interval at the end of the `update_power_stats` function (listing 5).

```
678 /*****************************************/
679 /* CS 203A Catching VSF and FSF from user */
680 /*****************************************/
681   opt_reg_int(odb, "-DVFSInterval", "Number of cycles for power monitoring interval",
682          &DVFSInterval, /* default */100000,
683          /* print */TRUE, /* format */NULL);
684
685   opt_reg_float(odb, "-DVFSTargetPower", "Target power budget controlled at each interval",
686          &DVFSTargetPower, /* default */6000000.00,
687          /* print */TRUE, /* format */"%12.2f");
688
689   opt_reg_float(odb, "-DVFSIncrement", "Increment for scaling of Voltage and Frequency",
690          &DVFSIncrement, /* default */0.1,
691          /* print */TRUE, /* format */"%12.1f");
692
693   opt_reg_flag(odb, "-DVFSTurnOff", "Do not execute the DVFS controller",
694          &DVFSTurnOff, /* default */FALSE,
695          /* print */TRUE, /* format */NULL);
696
697 /*****************************************/
698 /*                                       */
699 /*****************************************/
```

Listing 1: Capturing user parameters.

```
99  /****************************************/
100 /* CS 203A Declaring VSF and FSF        */
101 /****************************************/
102 float VSF = 1.0;
103 float FSF = 1.0;
104
105 /****************************************/
106 /* CS 203A Declaring arguments          */
107 /****************************************/
108 int DVFSInterval;
109 float DVFSTargetPower;
110 float DVFSIncrement;
111 int DVFSTurnOff;
112
113 /****************************************/
114 /* CS 203A Declaring auxiliar varibles  */
115 /****************************************/
116 #define Mhz 600e6
117 float power_this_interval;
118 float total;
119 float previous_total = 0;
120 float avg_power;
121 float power_factor;
122 extern FILE * output;
123
124 /****************************************/
125 /*                                      */
126 /****************************************/
```

Listing 2: Declaration of variables.

```
4940 /*****************************************************************************************/
4941 /* CS 203A DVFS Controller                                                               */
4942 /*****************************************************************************************/
4943
4944   if(sim_cycle % DVFSInterval == 0){
4945     power_this_interval = total - previous_total;
4946     avg_power = power_this_interval / DVFSInterval;
4947
4948     if(DVFSTurnOff == FALSE){
4949       if(power_this_interval > DVFSTargetPower && VSF > 0.3){
4950         VSF -= DVFSIncrement;
4951         FSF -= DVFSIncrement;
4952       }
4953       if(power_this_interval < DVFSTargetPower && VSF < 10.0){
4954         VSF += DVFSIncrement;
4955         FSF += DVFSIncrement;
4956       }
4957     }
4958
4959     fprintf(output,"%f:%f:%f:%f:%f\n", power_this_interval, avg_power, VSF, FSF, FSF*Mhz);
4960     previous_total += power_this_interval;
4961   }
4962
4963 /*****************************************************************************************/
4964 /*                                                                                       */
4965 /*****************************************************************************************/
```

Listing 3: Source code DVFS controller.

```
/***************************************/
/* CS 203A Defining a new Power factor    */
/***************************************/

extern float VSF;
extern float FSF;
#define Powerfactor (FSF)*(Mhz)*(VSF*VSF)*Vdd*Vdd
extern float total;
extern float power_factor;

/***************************************/
/*                                     */
/***************************************/
```

Listing 4: Power factor modification.

```
  /*************************************************************************/
  /* CS 203A Computing total power per cycle                               */
  /*************************************************************************/
  total += total_cycle_power_cc1 + total_cycle_power_cc2 + total_cycle_power_cc3;

  /*************************************************************************/
  /*                                                                       */
  /*************************************************************************/
```

Listing 5: Computing total power per interval.

2 Experiments

To run the experiments we select target power at each case without having the controller working. For Go, we found that the average total power consumption was around 5400000 W. In the case of Anagram the average power consumption was 7400000 W. We modified the upper bound of the scaling factor to 10 in order to get a more wide range of possible values.

2.1 Results for Go

Given time restriction we limit the number of instructions for this experiment to 50 million using the `-max:inst` parameter. Table 1 summarizes the results of the experiment. When we use the controller, we can see that while the total and average power consumption decrease, the execution time increases. Figures 1 and 2 show the change on the scaling factors and the corresponding total power consumption respectively at each interval. The red line on figure 2 marks the target power. We can see that the power remains around the target.

	DVFS Controller	Baseline
Total run-time	0.0731 s	0.0610 s
Total energy consumption	1998777579 W	2008539742 W
Average power consumption	53.75	54.02

Table 1: Results for the Go experiment

2.2 Results for Anagram

We can see a similar behavior to the previous section. Table 2 and Figures 3 and 4 summarize the results for this experiment.

	DVFS Controller	Baseline
Total run-time	0.0444 s	0.0418 s
Total energy consumption	1880593561 W	1893120690 W
Average power consumption	74.33	74.82

Table 2: Results for the Anagram experiment

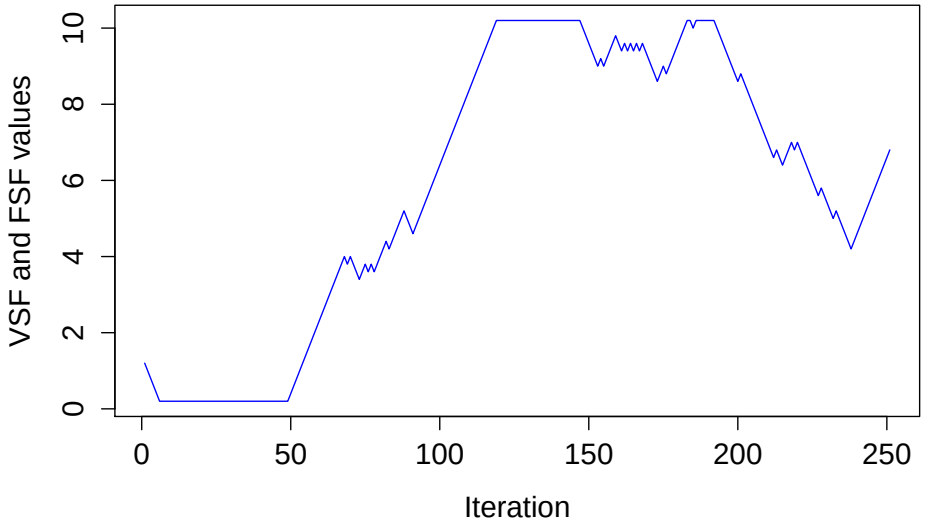

Figure 1: Scaling factors VSF and FSF for each interval for the Go experiment.

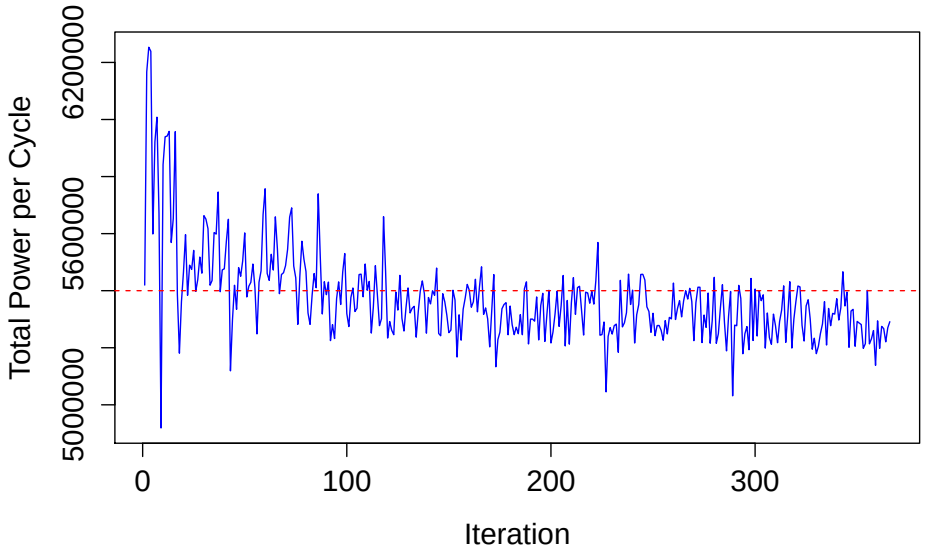

Figure 2: Total power per interval for the Go experiment. Red line is the target power.

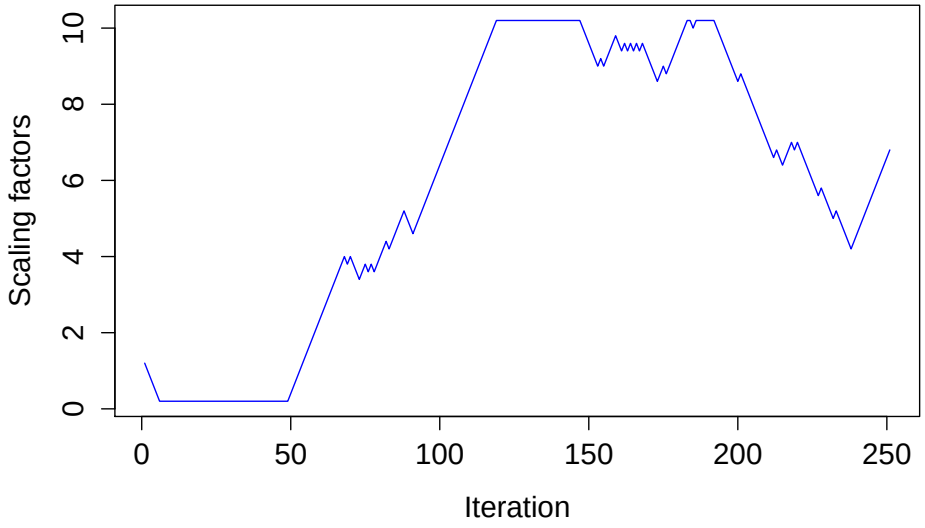

Figure 3: Scaling factors VSF and FSF for each interval for the Anagram experiment.

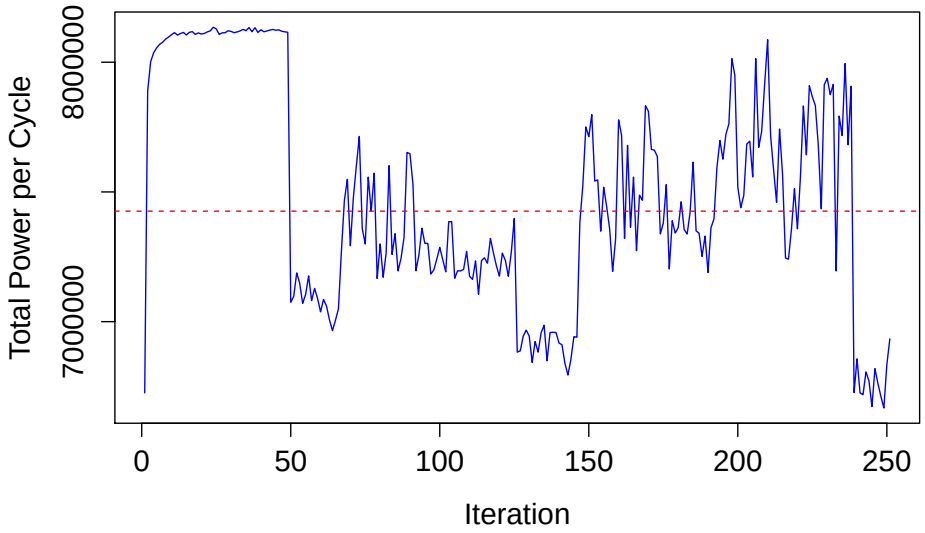

Figure 4: Total power per interval for the Anagram experiment. Red line is the target power.