CS203A - ADVANCED COMPUTER ARCHITECTURE
PROJECT PHASE - 2

Mohit Morarka (SID: 861130666) **Ambuj Lawania (SID: 861130271)**

Q1.(a) Stream Buffer

To create a stream buffer we decided to adopt the cache structure since the built in functions in Simplescalar would ease in its implementation. We modified the code such that the stream buffer (instruction and data) in the form of a cache structure were placed between the L1 and L2 caches respectively.

The stream buffer is initiated using command line parameters to determine block size. The stream buffer is called upon when a miss occurs in L1 cache. The buffer fetches blocks based on spatial locality and fills it when a miss a occurs in L1. In case a miss occurs against it flushes the buffer and refetches blocks

Modifications in code:

1. Declared data and instruction stream buffer , stream buffer options and stream buffer latency variables. The stream buffer is initiated as a cache structure.

2. Modified the L1 instruction and data cache mishandler function to first search in their respective stream buffers on a miss before searching in L2 cache and finally memory. Caches are accessed using the built-in function 'cache_access'

3. Created a separate miss handler function for the stream buffers where on a miss they would fetch 4 sequential blocks from L2 cache.

4. Created Data and Instruction Buffer based on block size parameters passed by the user via command line. The buffers are initiated using the 'cache_create' function and the legality of parameters is first checked.

5. Implemented a flush functionality in the stream buffer such that every time a miss occurs in the stream buffer , it is first flushed before fetching from L2.

Note: Line by Line Changes with explanation along with commands are present in the Read-Me File.

Results:
Anagram Benchmark

	Buffer Size: 0	BufferSize: 2	Buffer Size: 4	Buffer Size: 8
Miss Rate: il1	.07383	.06998	.04322	.01132
Miss Rate: dl1	.1389	.1229	.1044	.0887

Go Benchmark

	Buffer Size :0	Buffer Size: 2	Buffer Size: 4	Buffer Size: 8
Miss Rate: il1	.0148	.01398	.00845	.00236
Miss Rate: dl1	0.0209	0.0186	.01231	.0069

Observation and Inference:

1. **Miss rate of the given experiment decreases with the increase in buffer size because stream buffer takes advantage of principle of locality. Whenever a miss occurs in L1 cache successive cache lines are placed in the stream buffer starting at the miss address.**

Q1.(b) Memory Pre-fetching

We modified the command line parameters to enable pre-fetching in unified L2 cache and obtained the following the results.

Anagram Benchmark

	0	1	2	4
Run Time	20	19	18	19
Miss Rate (%)	0.2481	0.2481	0.2481	0.2481

Go Benchmark

	0	1	2	4
Run Time	**23**	**22**	**26**	**22**
Miss Rate (%)	**0.2491**	**0.2491**	**0.2491**	**0.2491**

Observation and Inference:

1. The run time is observed to have decreased and this could be because for every miss occurring in L1 cache , the data that it needs to fetch from main memory is probably already pre-fetched in L2 thus speeding it up. However sporadic jumps are observed slightly as number of prefetches increase probably because of the increased time required to pre-fetch blocks.

2. The cache miss rate is for the unified L2 cache and since the original miss rate was very low to begin with and major number of misses are concentrated in L1, thus L2 cache miss rate remains relatively similar

Part 2. DVFS Controller
We have implemented a DVFS controlling mechanism which monitors power consumption at regular intervals. The controller increases/decreases the voltage and frequency of the CPU to meet a specific power budget.

<u>Anagram-</u>

<u>When DVFS control is OFF</u>

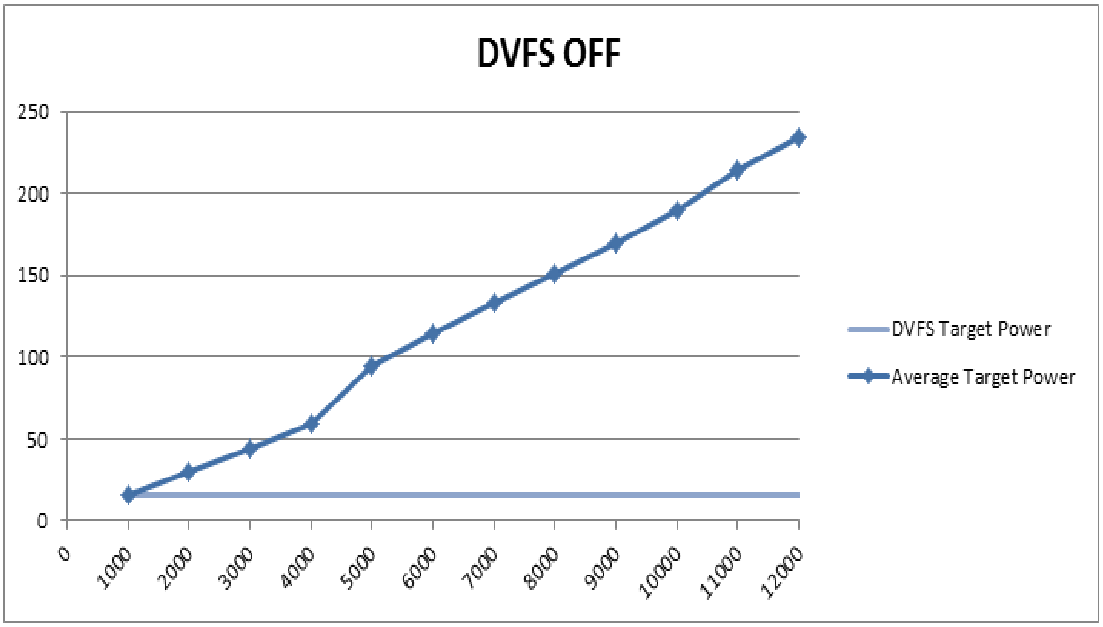
- **Here the default value of N (DVFS Interval) is 1000 cycles**
- **Default value of VSF and FSF are taken as 1.0**
- **DVFS Target Power taken is 15.87**

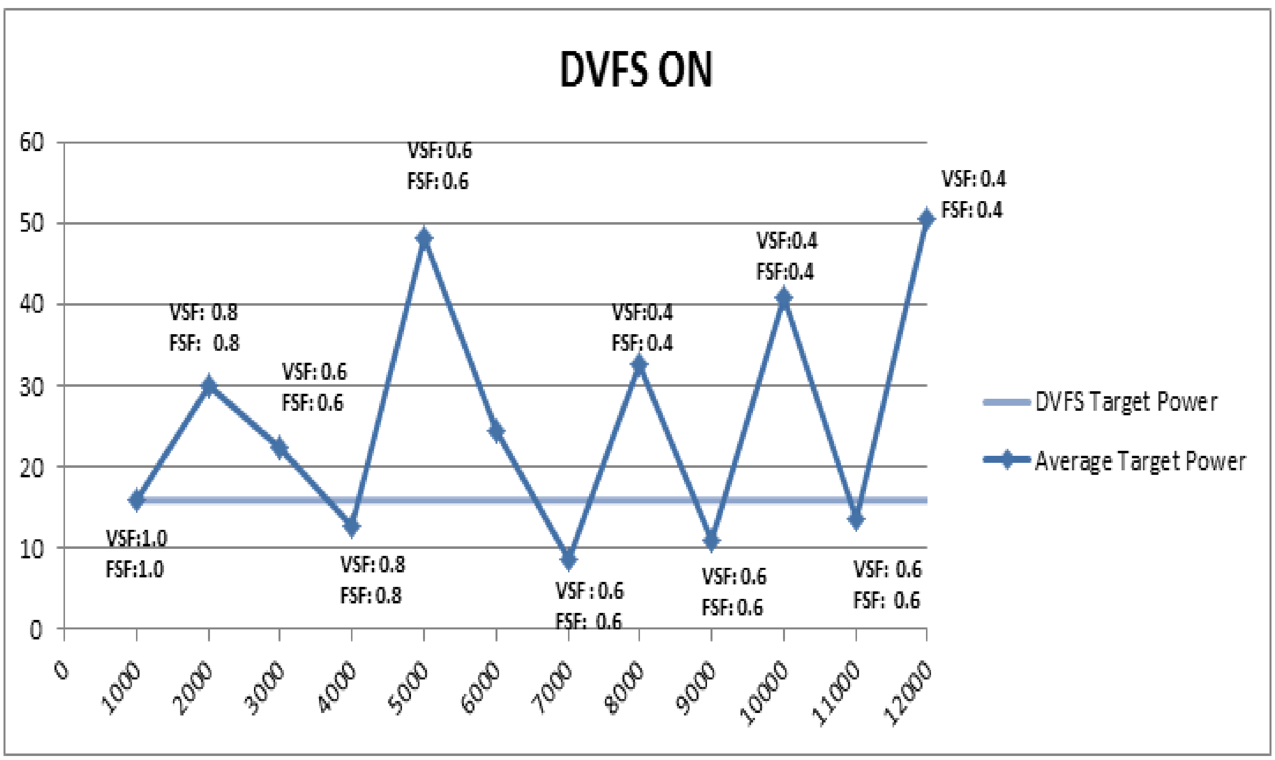
DVFS Interval Cycles	VSF	FSF	DVFS Target Power	Average Target Power	Total Power (cc1 +cc2 +cc3)
1000	1.0	1.0	15.87	15.87	15878.41
2000	1.0	1.0	15.87	29.94	29942.90
3000	1.0	1.0	15.87	43.81	43817.93
4000	1.0	1.0	15.87	58.99	58991.96
5000	1.0	1.0	15.87	94.62	94622.24
6000	1.0	1.0	15.87	114.07	114072.15
7000	1.0	1.0	15.87	132.77	132776.78
8000	1.0	1.0	15.87	150.93	150939.95
9000	1.0	1.0	15.87	169.21	169214.62
10000	1.0	1.0	15.87	189.57	189578.68
11000	1.0	1.0	15.87	213.91	213917.51
12000	1.0	1.0	15.87	234.15	234158.48

<u>When DVFS control is ON</u>

- **Here the default value of N (DVFS Interval) is 1000 cycles**
- **Default value of VSF and FSF are taken as 1.0**
- **DVFS Target Power taken is 15.87**
- **By Default value of FSF and VSF is 1.0 and we vary it between 0.2 and 2.0.**
- **Here in the second interval Average Target Power (29.87) is greater than DVFS Target Power (15.87) so we decrease the FSF and VSF by 0.2 and set them to 0.8 each. In the next cycle Average Target Power (22.38) is still greater than DVFS Target Power (15.87) so we again decrease the VSF and FSF by 0.2 and set them to 0.6 each.**
- **In the fourth cycle Average Target Power (12.71) becomes less than DVFS Target Power (15.87) so we increase the FSF and VSF by 0.2 and set them to 0.8 each.**

DVFS Interval Cycles	VSF set to	FSF set to	DVFS Target Power	Average Target Power	Total Power (cc1 +cc2 +cc3)
1000	1.0	1.0	15.87	15.87	15878.41
2000	0.8	0.8	15.87	29.87	29875.07
3000	0.6	0.6	15.87	22.38	22389.76
4000	0.8	0.8	15.87	12.71	12717.06
5000	0.6	0.6	15.87	48.29	48293.97
6000	0.4	0.4	15.87	24.56	24560.24
7000	0.6	0.6	15.87	8.46	8469.97
8000	0.4	0.4	15.87	32.49	32498.52
9000	0.6	0.6	15.87	10.79	10795.30
1000	0.4	0.4	15.87	40.81	40817.11
1100	0.6	0.6	15.87	13.64	13644.19
1200	0.4	0.4	15.87	50.40	50405.42


DVFS OFF
250
200
150
100
50
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
DVFS Target Power
Average Target Power


DVFS ON
60
50
40
30
20
10
0
VSF:1.0
FSF:1.0
VSF: 0.8
FSF: 0.8
VSF: 0.6
FSF: 0.6
VSF: 0.8
FSF: 0.8
VSF: 0.6
FSF: 0.6
VSF : 0.6
FSF: 0.6
VSF:0.4
FSF: 0.4
VSF: 0.6
FSF: 0.6
VSF:0.4
FSF:0.4
VSF: 0.6
FSF: 0.6
VSF: 0.4
FSF: 0.4
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
DVFS Target Power
Average Target Power

<u>Go-</u>
<u>When DVFS control is OFF</u>

Here the default value of N (DVFS Interval) is 1000000 cycles

Default value of VSF and FSF are taken as 1.0

DVFS Target Power taken is 58.21

DVFS Interval Cycles	VSF	FSF	DVFS Target Power	Average Target Power	Total Power (cc1 +cc2 +cc3)
1000000	1.0	1.0	58.21	58.21	58212268.00
2000000	1.0	1.0	58.21	112.30	112306040.00
3000000	1.0	1.0	58.21	164.99	164997600.00
4000000	1.0	1.0	58.21	216.87	216870064.00
5000000	1.0	1.0	58.21	268.66	268668480.00
6000000	1.0	1.0	58.21	321.22	321225792.00
7000000	1.0	1.0	58.21	373.80	373807552.00
8000000	1.0	1.0	58.21	426.51	426510592.00
9000000	1.0	1.0	58.21	478.196	478196448.00
10000000	1.0	1.0	58.21	529.57	529579168.00
11000000	1.0	1.0	58.21	581.27	581275072.00
12000000	1.0	1.0	58.21	632.72	632724800.00
13000000	1.0	1.0	58.21	684.48	684487552.00
14000000	1.0	1.0	58.21	736.00	736001472.00
15000000	1.0	1.0	58.21	787.63	787637376.00
16000000	1.0	1.0	58.21	838.88	838887808.00
17000000	1.0	1.0	58.21	890.03	890033472.00

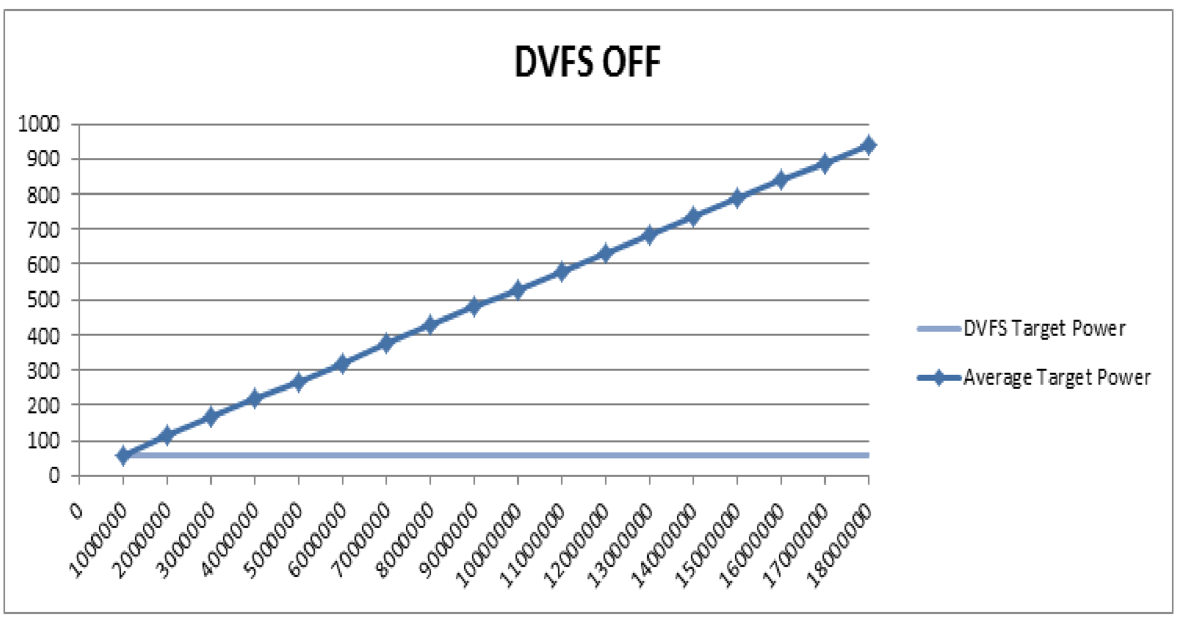
18000000	**1.0**	**1.0**	**58.21**	**940.86**	**940861120.00**

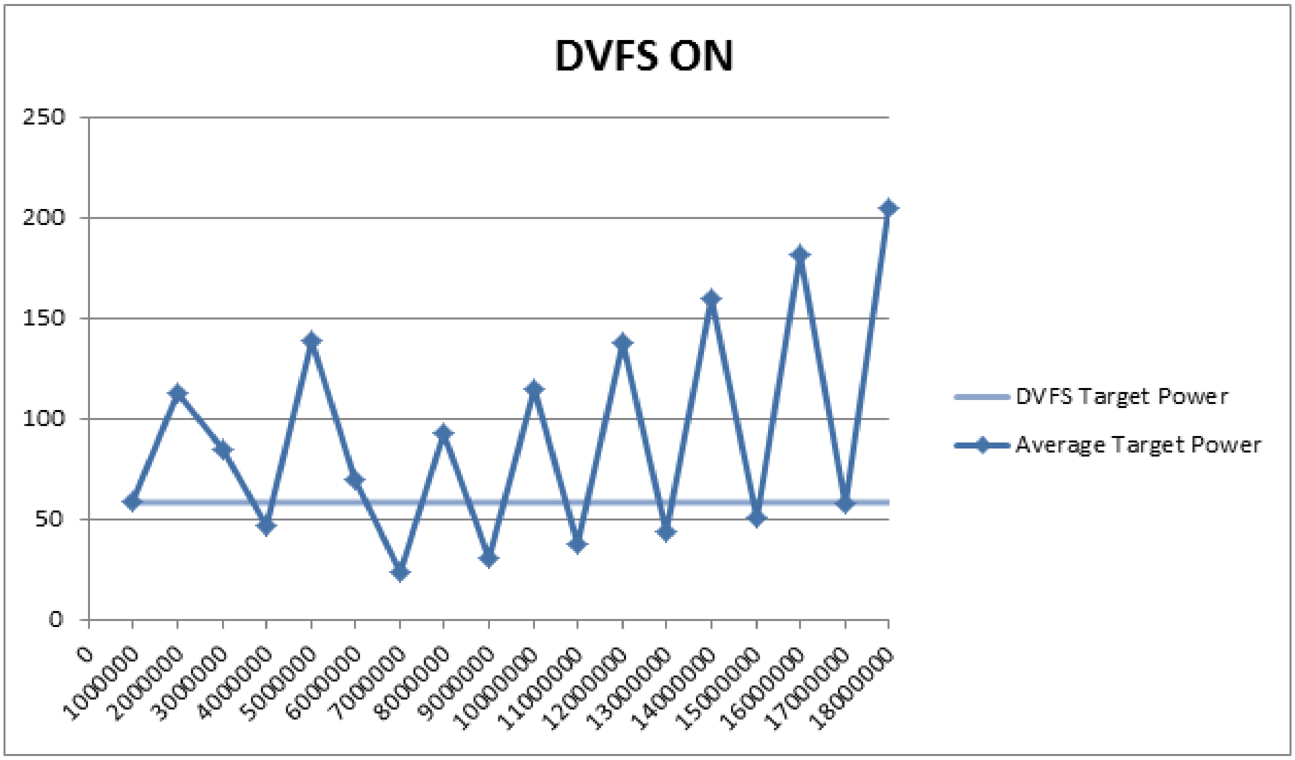
When DVFS control is ON

- **Here the default value of N (DVFS Interval) is 1000000 cycles**
- **Default value of VSF and FSF are taken as 1.0**
- **DVFS Target Power taken is 58.21**
- **By Default value of FSF and VSF is 1.0 and we vary it between 0.2 and 2.0.**
- **Here in the second interval Average Target Power (112.89) is greater than DVFS Target Power (58.21) so we decrease the FSF and VSF by 0.2 and set them to 0.8 each. In the next cycle Average Target Power (84.91) is still greater than DVFS Target Power (58.21) so we again decrease the VSF and FSF by 0.2 and set them to 0.6 each.**
- **In the fourth cycle Average Target Power (47.08) becomes less than DVFS Target Power (58.21) so we increase the FSF and VSF by 0.2 and set them to 0.8 each.**

DVFS Interval Cycles	VSF set to	FSF set to	DVFS Target Power	Average Target Power	Total Power (cc1 +cc2 +cc3)
1000000	1.0	1.0	58.21	58.21	58212268.00
2000000	0.8	0.8	58.21	112.89	112895920.00
3000000	0.6	0.6	58.21	84.91	84918136.00
4000000	0.8	0.8	58.21	47.08	47085108.00
5000000	0.6	0.6	58.21	138.26	138262544.00
6000000	0.4	0.4	58.21	69.73	69739424.00
7000000	0.6	0.6	58.21	24.04	24045634.00

8000000	0.4	0.4	58.21	92.59	92595600.00
9000000	0.6	0.6	58.21	30.76	30760114.00
10000000	0.4	0.4	58.21	114.96	114968776.00
11000000	0.6	0.6	58.21	37.38	37389848.00
12000000	0.4	0.4	58.21	137.35	137358736.00
13000000	0.6	0.6	58.21	44.02	44028052.00
14000000	0.4	0.4	58.21	159.77	159775728.00
15000000	0.6	0.6	58.21	50.66	50661752.00
16000000	0.4	0.4	58.21	182.10	182108032.00
17000000	0.6	0.6	58.21	57.24	57247460.00
18000000	0.4	0.4	58.21	204.24	204242784.00





For Anagram-

../sim-outorder anagram.alpha words 1 1000 15.87 < anagram.in >../../OUT.std 2>../../OUT.stat

Command line parameters-

DVFS On/OFF is 1/0

DVFS Interval is 1000

DVFS Target Power is 15.87

Steps-

- **First give the command line parameters as stated above.**
- **Command line parameters goes to main.c line 237 where we first check whether we are running Anagram or Go.**
- **Then we take the command line inputs and store them in the variable DVFS Interval, DVFSTargetPower and DVFS flag.**
- **Then it goes to sim-outorder.c to line 4882 to check whether we need to run with DVFS on/off and then trigger the DVFS controller every N cycle.**
- **In the DVFS controller we check if Average_Power is less than DVFS_Target_Power then we increase the VSF and FSF values by 0.2 each and if it is greater we decrease the VSF and FSF values by 0.2 each which dynamically updates the VSF and FSF values.**
- **Inside Power.c we check the update power stats function and calculate the**

Average_Power (cc1+cc2+cc3/DVFS Interval) and Total_Power (cc1+cc2+cc3).

Explanation-

In sim-outorder.c
Line 4882:we check if((sim_cycle % DVFS Interval == 0) && (DVFS_flag == 1))
i.e if the sim_cycle is equal to the DVFS Interval cycles then we call the DVFS controller
We also check for the DVFS flag if is 0/1.If it is 1 we consider DVFS as on and trigger the controller.

In power.c
Line 610 inside update_power_stats we update the power factor to Power Factor 1 = FSF*pow(VSF,2); with the dynamic values of VSF and FSF
from the controller.
Line 616 and 618 we calculate the Average_Power(cc1+cc2+cc3/DVFS Interval)
and Total_Power(cc1+cc2+cc3).

In DVFS.c
Line 12 we update the VSF and FSF in lock steps by comparing the Average_Power and Target_Power.

In main.c
Line 237 and 243 we take the command line values for Go and Anagram.

For Go-
../sim-outorder go.alpha 2 8 2stone9.in 1 1000000 58 >../../ OUT.std 2>../../ OUT.stat
Command line parameters-
DVFS On/OFF is 1/0
DVFS Interval is 1000000
DVFS Target Power is 58.21

Steps-

- **First give the command line parameters as stated above.**
- **Command line parameters goes to main.c line 237 where we first check whether we are running Anagram or Go.**
- **Then we take the command line inputs and store them in the variable DVFS Interval, DVFSTargetPower and DVFS flag.**
- **Then it goes to sim-outorder.c to line 4882 to check whether we need to run with DVFS on/off and then trigger the DVFS controller every N cycle.**
- **In the DVFS controller we check if Average_Power is less than DVFS_Target_Power then we increase the VSF and FSF values by 0.2 each and if it is greater we decrease the VSF and FSF values by 0.2 each which dynamically**

updates the VSF and FSF values.

- **Inside Power.c we check the update power stats function and calculate the Average_Power (cc1+cc2+cc3/DVFS Interval) and Total_Power (cc1+cc2+cc3)**