Introduction

Direct Memory Access (DMA) allows devices to access the memory directly without the involvement of the CPU so that the memory access is faster and also the load on CPU is reduced. This project aims to analyze the change in performance of the cortex_m0 Soc with inclusion of a DMA. As performance metrics, the memory transactions per cycle and dynamic power per cycle are considered.

Evaluation Method

The idea is to create test cases (programs) to run on the SOCs and observe the transactions per cycle and the power consumed by the soc. Identical test cases must be used for comparing the two cases.
The challenge in creation of the test cases is the code that could run utilizing the DMA. Any source address other than the address on the dma.c file resulted in an error. The existing dma.c code was used on the cortex_mo processor with DMA and a similar arraycopy.c was created in order to run on the cortex_mo processor without the DMA. Two programs, fibonacci and an array copy program were chosen to be simulated.

The output observed for the array copy program is as follows:

Gate-Level Power (W)													
		Sim Time (ns)			**Transfer Count**			**DMA**		**CPU**		**MEMCTL**	
system	Workload	Start	Stop	cycles	Reads	Writes	xfers/cyc	Dynamic	Leakage	Dynamic	Leakage	Dynamic	Leakage
u_cortex m0ds	dma.c(load)	1005	178495	17749	2088	347	0.1371908277	2.78E-03	9.74E-05	1.71E-03	6.44E-04	4.06E-03	6.71E-04
u_cortex m0ds	burstarray.c	1015	241655	24064	2944	344	0.1366356383	0	0	1.69E-03	6.44E-04	3.57E-03	6.71E-04

It can be observed that there is an increase in transfers per cycle with the use of DMA however, the dynamic power consumed has also increased. DMA adds hardware into the Soc thereby increasing the power consumed. Since there is more wires from the DMA, complexity increase at the memory controller may owe to the increase in dynamic power seen at the memory controller.

The output observed for the fibonacci with and without inclusion of the DMA are as follows:

Gate-Level Power (W)													
		Sim Time (ns)			Transfer Count			DMA		CPU		MEMCTL	
System	Workload	Start	Stop	cycles	Reads	Writes	xfers/cyc	Dynamic	Leakage	Dynamic	Leakage	Dynamic	Leakage
u_cortexm0ds	dma.c(fib)	1005	433175	43217	5204	907	0.1414026888	2.62E-03	9.74E-05	1.71E-03	6.44E-04	4.15E-03	6.71E-04
u_cortexm0ds	fibbonaci.c	1015	433185	43217	5204	907	0.1414026888	0	0	1.69E-03	6.44E-04	3.76E-03	6.71E-04

The fibonacci.c program doesn't include the direct memory access although the DMA is present. So the there is no improvement in the transfers per cycle. However, it can be seen that there is an increase in the dynamic power consumed with addition of DMA. The complexity of the memory system increases with additional hardware so the complexity of Memory Controller also increases (additional wires and circuits) leading to increase in power consumed.

The Stub is a system which supports burst transactions. As a third case, the soc with the stub is considered. The addresses and data that the stub has to deal with are loaded into the xact transaction file.
in order to compare , similar transactions were loaded into the Soc with DMA and the Soc without DMA .
The Soc with DMA was loaded with the appropriate format and the output observed in the sim.out, the memory transactions was then loaded into the Soc without DMA and simulated.

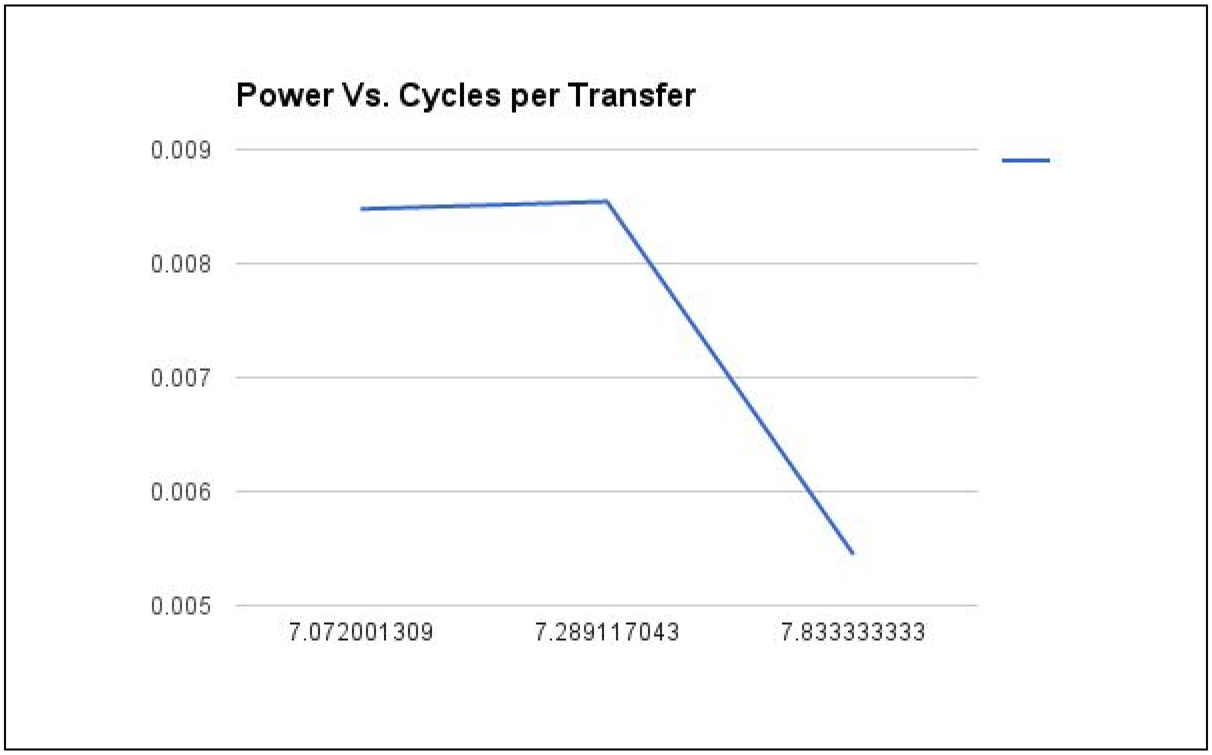
The output observed for the transactions using the stub with and without inclusion of the DMA are as follows:

Gate-Level Power (W)													
		Sim Time (ns)			Transfer Count			DMA		CPU		MEMCTL	
system	Workload	Start	Stop	cycles	Reads	Writes	xfers/cyc	Dynamic	Leakage	Dynamic	Leakage	Dynamic	Leakage
Stub_SOC	with_dma	1005	4765	376	25	23	0.1276595745	2.34E-03	9.74E-05	1.94E-04	6.44E-04	2.91E-03	6.71E-04
Stuc_SOC	without_dma	1015	5935	492	27	23	0.1016260163	0	0	1.94E-04	6.44E-04	2.01E-03	6.71E-04

We observe an increase in the transfers per cycle in this case with the use of DMA and we also see an increase in power consumed as seen earlier. But relatively the difference of increase in transfers is more compared to the array copy program, this may be owing to increased number of memory access'.

Performance Observed for the Soc with DMA is as follows:

With DMA		
System	transfers/cycle	Dynamic Power
u_cortexm0ds	7.072001309	0.0084776
u_cortexm0ds	7.289117043	0.00854281
Stub_SOC	7.833333333	0.0054452



BUGS OBSERVED:

The following are a few bugs observed during the course of the project.

#1

An attempt to change the code of the DMA from (i) to (ii) resulted in an infinite loop

```
*dma_sr = 0x00000040;
*dma_dr = 0x00004000;
*dma_len = 0x0000000f;
      (i)
```

```
*dma_sr = 0x00000040;
*dma_dr = 0x00004000;
*dma_len = 0x00000008;
      (ii)
```

#2
An attempt to use a different pointer to feed a known value into the memory location 0x00000040 prior to pointing it to the DMA location in order to observe known values propagate through the register also failed and resulted in a simulation stop requested by the CPU.

#3
Xact.txt.
Transaction with writes followed by a read fed at the same time resulted in an infinite loop.
#4
Any changes to the source and destination sent to the DMA in dma.c resulted in a simulation stop. In some cases the access resulted in “ neither RAM nor Console”.

PREDICTIONS TO IMPROVE THE Soc PERFORMANCE:

- The power consumed by the Soc can be reduced by decreasing the complexity of the memory controller and increasing the complexity on the DMA’s end. The management of the DMA can be more reliant on the DMA.
- The memory instructions can be bypassed in the CPU when the DMA is involved.
- For a program that doesn’t require the use of DMA, the DMA can be disabled in order to conserve power and improve performance.

CONCLUSION:

As seen in the three cases considered, there is at least an increase in transfers per cycle with the use of DMA. Increase in memory transaction through the DMA may increase the speed of transfers. The downside of using DMA is the increase in dynamic power consumption observed.