**Flash, CCM ve SRAM içerisindeki kod parçasının performans karşılaştırması**

Birçok makalede işlemciniz için yazdığınız kodu ram bloğuna (SRAM) yerleştirdiğinizde yazdığınız kodun hızlandığı söylenmekte. Bazı işlemci çeşitlerinde işlemciye daha yakın bir şekilde konumlandırılmış “Close Coupled Memory- CCM” isimli bir ram bloğu da bulunmakta. Peki gerçekten flashta ,SRAM ve CCM ‘de çalışan kod arasında ne kadar performans farkı oluşuyor. Aşağıda bununla ilgili yaptığım test sonuçlarını bulabilirsiniz.

Test sonuçlarına direkt olarak geçmeden kullanılan fonsiyon ve değişkenlerin konfigürasyonu üzerinde duracağım. Değişkenler **private static** olarak tanımlandı kullanılan test fonsiyonu ise **64 bit karekök** fonksiyonu. Kod geliştirme ortamı olarak ise KEIL IDE uVision 5 (Arm Compiler V5.06) kullanılmıştır.

|  |
| --- |
| 1. uint64\_t gapsqrt64(uint64\_t a) { 3. **static**     uint64\_t rem  \_\_attribute\_\_((section(VARS\_SPACE)))=0; 4. **static**     uint64\_t root \_\_attribute\_\_((section(VARS\_SPACE)))=0; 5. **static**     uint8\_t  i    \_\_attribute\_\_((section(VARS\_SPACE)))=0; 7. rem  =0; 8. root = 0; 9. i=0; 11. **for** (i = 64 / 2; i > 0; i--) 12. { 13. root <<= 1; 14. rem = (rem << 2) | (a >> (64 - 2)); 15. a <<= 2; 16. **if** (root < rem) { 17. rem -= root | 1; 18. root += 2; 19. } 20. } 21. **return** root >> 1; 22. }   Algoritma Jack W.Crenshaw’ın 1998 yılındaki embedded.com da yayınlanan makalesinden alınmıştır  <http://www.embedded.com/electronics-blogs/programmer-s-toolbox/4219659/Integer-Square-Roots> |

Ana program içerisinden aşağıda görüldüğü gibi karekök fonksiyonu çağrılmıştır.

|  |
| --- |
| 1. **while**(**true**){ 2. COMMON\_PORT->ODR ^= COMMON\_PIN; 3. RESULT = gapsqrt64(0x20000000000); 4. RESULT = 0; 5. } |

COMMON\_PIN push pull çıkış olarak ayarlayarak oluşan pin değişim frekansını karşılaştırma amaçlı kullandım.

Kodun üç farklı lokasyonda ( Flash,SRAM ve CCM ) olma durmuna göre değişkenler CCM ve SRAM içerisine konumlandırılarak 6 farklı test yapılarak oluşan pin frekansı ölçüldü.

|  |  |
| --- | --- |
|  |  |
| **Resim 1** : Setup ve osiloskop görüntüsü | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Kod Yeri** | | |
| **FLASH** | **CCM** | **SRAM** |
| Değişkenler CCM içerisinde | 47 | **39.34** | 33.37 | Khz |
| Değişkenler SRAM içerisinde | 39.35 | 48.98 | **33.37** | Khz |
| **Grafik 1**: Karşılaştırma sonuçları | | | | |

|  |
| --- |
|  |
| **Resim 2** : Karşılaştırma sonuçları |

Sonuç:

Herhangi bir ayarlama yapmaksızın derleyici tarafından kod flash içerisinde, değişkenler ise SRAM içerisine atanır. Bu ayarlamada 39.35Khz lik bir çıkış frekansı okuyoruz. En yüksek çıkış frekansını (48,98 Khz ) veren konfigürasyon ise Resim 2’den görüldüğü üzere kodun CCM içerisinde, değişkenlerin ise SRAM içerisinde bulunduğu konfigürasyon olarak görünüyor. İkisi arasındaki performans artışını da %27.7 olarak hesaplayabiliriz.

Kodu flash ve ram lokasyonlarına geçirmek için gerekli konfigürasyon ayarlamalarını AN4296 uygulama notundan(1) bulabilirsiniz.

Not:

AN4296 uygulama notundan sayfa 3’te okuyabileceğiniz gibi kod ve değişkenlerin aynı ram lokasyonuna konulması işlemcinin ramden data ve kodun aynı aynı çekilmesi ihtimalinde çarpışmaya sebep olabileceğinden önerilmemektedir. İşlemcinin böyle bir durumda harward konfigürasyonundan Von Neumann konfigürasyonuna geçiş yapacağı ve bunun da performans kaybına sebep olabileceği belirtilmektedir. Testlerde bu uyarıya rağmen ram ve kod aynı bölgelere konularak test yapılmıştır. Grafik 1’de ilgili test sonuçları kırmızı olarak işaretlenmiştir.

Kaynaklar:

1. **AN4296** Use STM32F3/STM32G4 CCM SRAM with IAR™ EWARM, Keil® MDK-ARM and GNU-based toolchains :

<https://www.st.com/resource/en/application_note/dm00083249-use-stm32f3stm32g4-ccm-sram-with-iar-ewarm-keil-mdkarm-and-gnubased-toolchains-stmicroelectronics.pdf>