

Störst och snabbast

Toshiba ligger långt framme i utvecklandet av nya minnes-typer. Vid ISSCC-kongressen i San Francisco nyligen presenterade man två viktiga framsteg inom FeRAM och MRAM.

Det ena är ett nytt ferroelektriskt RAM (FeRAM) som inte bara är världens hittills största FeRAM med sin storlek på 64 Mbit, utan också det snabbaste med sin läs- och skrivhastighet på 200 Mbyte/s. Kretsen tillverkas i en 130 nm CMOS-process.

Det nya minnet baseras på Toshibas arkitektur chainFeRAM, som väsentligt minskar minnescellernas storlek (till 0,7171 μm^2). Kretslösningarna är optimerade för att minimera squelch-brus under läsning. Detta uppnås genom att cellraderna arbetar i en sekvens där varannan rad är aktiv när mellanliggande rader är inaktiva, och tvärtom. Härigenom går det att minska avstånden mellan raderna.

Inbyggda funktioner för felkorrigering ökar tillförlitligheten vid snabba operationer, även under svåra driftförhållanden. För att uppnå höga dataöverföringshastigheter arbetar minnet med skurvis överföring.

mram

Tillsammans med NEC har Toshiba också utvecklat ett nytt magnetoresistivt RAM (MRAM) som även det uppges vara störst med 16 Mbit och snabbast med 200 Mbyte/s, och som dessutom kan matas med 1,8 V.

Ett problem i MRAM-minnen är att den strömkälla som används för att generera magnetfältet vid skrivning degraderar läsoperationerna från minnescellerna. I det nya minnet används en konstruktion med skilda strömvägar för läsning och skrivning, vilket ger 38 procent lägre resistans i skrivledningarna. Härmed uppnås snabbare läsning, och en cykeltid på 34 ns, vilket uppges vara världsrekord. Chipytan blir också ca 30 procent mindre.

ANDERS LJUNGSTRÖM



I Toshibas nya FeRAM är varannan cellrad aktiv när mellanliggande rader är passiva, och tvärtom.

Samma minne lagrar kod och data

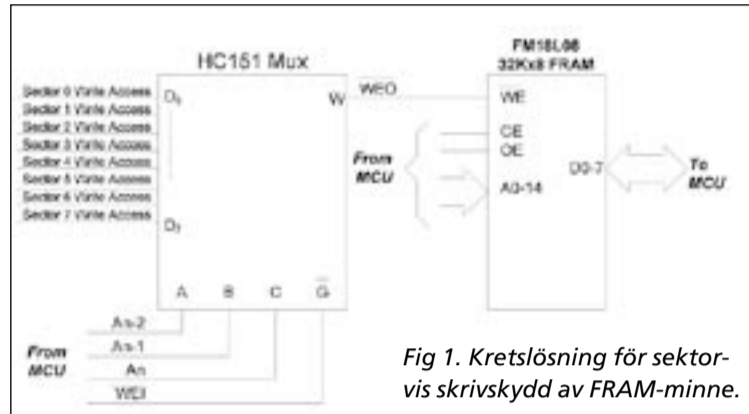


Fig 1. Kretslösning för sektorvis skrivskydd av FRAM-minne.

I många fall kan det vara attraktivt att använda ett och samma minneschip för lagring av både kod och data. Men det skapar vissa problem eftersom det handlar om två uppgifter som har motstridiga krav. Mike Alwais från Ramtron berättar här om några av problemen och ger en enkel lösning för ett av de viktigaste.

Minnesapplikationer kan generellt delas upp i lagring av exekverbar kod eller lagring av data. ROM-baserade teknologier är ickeflyktiga och lämpar sig därför för kodapplikationer. EEPROM är en variant av ROM som huvudsakligen används för ickeflyktig lagring av data. EEPROM är helt klart en kompromiss eftersom det används för lagring av både kod och data, men ger låga prestanda jämfört med alternativen.

Flashminnen kan användas för vissa datalagringsuppgifter, men de passar bäst för lagring av kod och ger begränsade prestanda vid lagring av data. Den viktigaste fördelen med flash i sådana applikationer är den låga kostnaden, snarare än lättanvändbarhet eller höga prestanda vid datalagring.

RAM-baserade teknologier fungerar både som arbetsrymd för exekvering av kod och som dataminne. De viktigaste typerna är DRAM och SRAM. Dessa teknologier ger en utmärkt blandning av goda egenskaper för både kod- och datalagring. Tyvärr möjliggör de traditionella alternativen bara tillfällig lagring. Om de inte backas upp med ett batteri måste de användas tillsammans med någon form av ickeflyktigt minne som kan ge permanent lagring.

Applikationer med begränsat utrymme behöver maximal funktionalitet i en och samma krets. Det idealiska vore om man kunde

lagra både kod och data i en och samma minneskrets. Eftersom detta måste vara ickeflyktigt kan vanligt RAM inte komma i fråga. Kvar finns då ROM-teknologierna, som ger dåliga prestanda vid datalagring, samt batteriuppbakad RAM och ferroelektriskt RAM (FRAM).

Det finns få ickeflyktiga minnesteknologier som lånar sig bättre för lagring av både kod och data är FRAM. En enda FRAM-krets uppfyller kraven hos båda applikationerna. Den praktiska gränsen sätts av vilka minnestätheter som finns tillgängliga. Men detta håller på att bli ett mindre problem i och med att det kommer FRAM med högre tätheter.

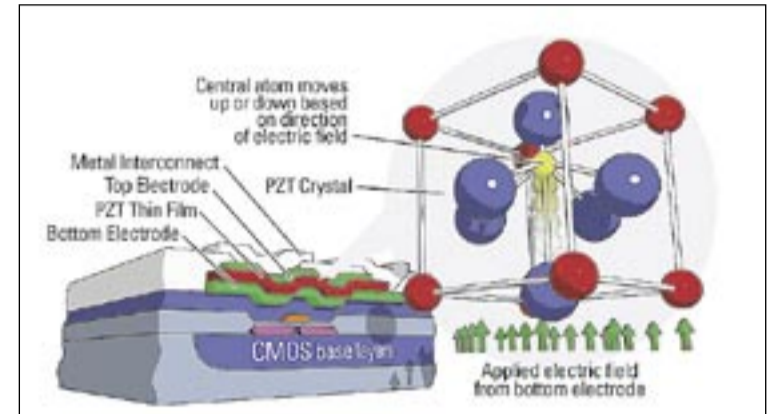
Ett FRAM med hög täthet är en idealisk lösning för lagring av både kod och data i en och samma krets. Men det finns ett antal faktorer att ta hänsyn till. Denna artikel handlar om vilka krav som ställs, och vad man skall tänka på när man kombinerar kod- och datalagring i ett och samma FRAM-chip.

KODLAGRING

Innan man väljer en krets för lagring av både kod och data kan det vara till hjälp att se på vad som är idealiskt för vardera applikationen separat. I många system används ROM eller OTP-EPROM för kodlagring, och det kräver ingen närmare omtanke. Dessa minnes-typer är ickeflyktiga och kan inte ändras i systemet. Därför behöver man bara tänka på basparametrar som minnesstorlek, accesstid och arbetsspänning.

Framför allt bibehåller dessa minnes sitt tillstånd under alla omständigheter. Kodlagring är dock till sin natur enbart läsning. Temporära variabler och parametrar som behöver ändras måste därför lagras i RAM.

I och med att vi fått fältprogrammerbara lagringmöjligheter uppstår frågan om vilka metoder som skall användas när man skriver till kodminne. Flashminne anses allmänt som det bästa valet som fältprogrammerbart minne. Men de första flashminneskretsarna



FRAM-teknologin är kompatibel med vanliga CMOS-tillverkningsprocesser. Den tunna ferroelektriska filmen placeras över CMOS-bas-skikten och läggs in mellan två elektroder. Processen avslutas med metallförbindningar och passivering.

var inte programmerbara direkt i systemet eftersom de krävde höga externa spänningar, och dessutom måste raderas helt innan de kunde programmeras.

I de fall där man bara har ett enda chip kanske det program som styr uppgraderingsprocessen lagras i och exekveras från den minneskrets som programmeras. Detta försvårar möjligheterna till bulkvis radering.

Senare typer av flashminne kunde programmeras i systemet, och de var sektoriserade så att man kunde radera delar av minnet separat före programmeringen. Detta gjorde dem mer användbara för fältuppgradering. Dagens flashminnen är fortfarande svåra att uppgradera i enchipssystem, men i övrigt betraktas de som relativt idealiska för kodlagring. Men flash har väsentliga brister som dataminne, vilket behandlas senare.

Det finns ett antal faktorer som spelar in vid kodlagring. Vissa av dessa har speciellt stor betydelse i system med bara en eller några få minneskretsar:

- Icke-flyktig lagring
- Tillräcklig täthet
- Accesstid vid läsning
- Möjlighet att skydda sig mot obehörig skrivning
- Fältprogrammerbarhet som helhet eller sektionsvis
- Programmerbara, med samtidig läsaccess

DATALAGRING

I måna avseenden är kraven vid datalagring helt motsatta de vid kodlagring. Datalagring är en mycket mer diversifierad uppgift, vilket ställer krav på flexibilitet och enkel skrivaccess. Minnena kan vara flyktiga eller ickeflyktiga, beroende av vad det är för slags data.

I många fall är dataminnets skrivfunktioner minst lika viktiga som dess läsfunktioner. Ett extremt exempel är en den svarta lådan i flygplan (flight data recorder), som förhoppningsvis aldrig behöver läsas av. I de flesta fall skall en skrivoperation inte kräva några

komplexa protokoll, och den skall definitivt inte ta lång tid. Dessutom måste det gå att skriva ett mycket stort antal gånger för att man skall kunna samla in nästan obegränsade mängder data.

I rent flyktiga applikationer används normalt RAM för datalagring. För relativt små datamängder, snabb dataaccess och låg effektförbrukning används SRAM, medan man i relativt stora databuffertar använder DRAM. Dessa minnestyper utför sina uppgifter tämligen väl, så länge som det handlar om flyktig lagring. Det går att göra en RAM-krets till ett ickeflyktigt minne genom att lägga till batterier, men detta skapar nya problem.

Icke-flyktig datalagring är en utmaning för de flesta vanliga minnesteknologier. ROM-baserade teknologier som flash är dåliga som dataminne, utom för de mest statiska konfigurationerna. Bara det att man talar om "programmering" visar att det är mycket svårare att lägga in nytt innehåll i ett flash än i RAM-baserade minnen.

Komplexa algoritmer för skrivning, fördröjningar vid radering och relativt långa skrivcykler är typiska egenheter hos flashminnen. Sektorer programmeras med relativt låg hastighet, men de måste först raderas, vilket sker med ytterst låg hastighet. EEPROM är det traditionella valet för ickeflyktigt minne eftersom det är enklare att använda. Men det har mycket långsamma skrivtider, och både flash och EEPROM klarar inte speciellt många skrivcykler innan de slitits ut.

FRAM

FRAM är den första minnestypen som optimerats för ickeflyktig datalagring. Det går att skriva lika snabbt till FRAM-minnen som det går att läsa. FRAM är ickeflyktigt och klarar ett mycket stort antal accesscykler ($>10^{12}$). För 3 V FRAM är värdet 10^{16} . Inga algoritmer eller protokoll krävs för att skriva till minnet, och det kan adresseras bytevis. FRAM ger den

mest flexibla lösningen med minst antal nackdelar för datalagring.

Att räkna upp vilka egenskaper som har betydelse vid datalagring är svårare än för kodlagring. Applikationerna är långt från homogena. I många system har frånvaron av lösningar hos äldre minnesgenerationer dessutom begränsat vilka typer av och hur stora mängder data som samlats in.

Viktiga egenskaper vid datalagring är.

- Snabb skrivaccess
- Möjlighet att skriva många gånger
- Enkla protokoll för skrivning (bäst är inget)
- Bitadresserbar skrivning
- Ickeflyktigt lagring (i vissa applikationer)
- Möjlighet att uppfylla behoven för både flyktig och ickeflyktig lagring.

datalagring med ett chip

Att kombinera kod- och dataapplikationer i en enda krets kan innebära att man försöker få ett minne att utföra tjänster som nästan helt utesluter varandra. Kod kräver ickeflyktig lagring, med möjlighet att kunna skriva vid enstaka tillfällen. Uppgradering av kod kommer aldrig att kräva ett stort antal skrivoperationer. Skrivtiden saknar som regel betydelse. I viss mån kan man säga



Ramtrons 1 Mbit FRAM.

att ju svårare det är att skriva till kodminne, desto bättre är det eftersom en ofrivillig skrivoperation kan vara katastrofal.

Datalagring kräver ofta en blandning av flyktigt och ickeflyktigt minne, eller åtminstone ett ickeflyktigt minne som det går att skriva till utan några restriktioner. Datalagring kräver ett relativt stort antal skrivoperationer, och skrivhastigheten kan också ha betydelse.

FRAM-teknologin håller nu på att optimeras för minnesapplikationer inom datalagring. FRAM-minnets mycket goda skrivegenskaper gör det mer attraktivt än flash och EEPROM, och att det är ickeflyktigt gör det mer attraktivt än SRAM. Och eftersom det är ickeflyktigt kan det användas som kodminne också.

Den viktigaste begränsningen för att använda FRAM i enchip-

lösningar är de tillgängliga tätheterna. Idag har det största minnet från Ramtron (FM18L08) storleken 32K x 8.

En viktig faktor vid konstruktionen är att förhindra oavsiktlig skrivning till kodarean, samtidigt som det skall gå att uppgradera koden vid behov. Att skriva till ett FRAM påminner annars mycket om att skriva till ett SRAM.

SKRIVSKYDD

FRAM kan enkelt anpassas för kodlagring genom att man skapar en enkel funktion för skrivskydd som förhindrar oavsiktlig skrivning till kodareor, och i somliga fall även vissa dataareor. Ett exempel på en sådan kretslösning visas i fig 1. Denna ger programmerbart skydd mot blockvis skrivning till ett FRAM-minne.

Om Sector Write Access-ingångarna hårdviras får man ett

permanent skydd. Om de istället ansluts till en MCU eller styrlogik får man möjlighet att dynamiskt ändra skrivskyddet.

Kretslösningen baseras på en vanlig CMOS-multiplexer som HC151, vilken används för att skapa en Write Enable-signal som är beroende av adressen.

I denna krets är Write Enable-ingången (/WE) aktivt låg, vilket FRAM-minnet kräver. /WE-signalen från MCU:n eller dekodern betecknas /WEI. /WE-utsignalen från kretsen till FRAM-minnet betecknas /WEO. Så länge som /WEI är hög kommer /WEO-utgången på HC151 att vara hög. När /WEI går över till låg kommer den aktuella adressen och motsvarande Write-Access-ingång att bestämma om /WEO-signalen till FRAM-minnet tillåts att gå över till låg.

Vi antar att de tre mest signifikanta minnesadresslinjerna ansluts till HC151, och att minnesrymden är uppdelad i åtta lika sektorer. För varje sektor bestämmer D_n -signalen till HC151 om sektorn skall vara skrivskyddad. Den inkommande adressen kommer att välja en av dessa åtta sektorer. Om motsvarande D_n -signal är hög kommer /WEO från HC151 att gå över till låg om /WEI är låg. Då kan de aktuella sektorerna skrivas. Om den valda

D_n är låg kommer /WEO att förbli hög för alla adresser i den aktuella sektorn, oavsett vilket tillstånd /WEI har.

Denna enkla kretslösning kan användas för att ge programmerbart skrivskydd till alla slags RAM-minnen. En möjlighet är att helt enkelt hårdvira inställningarna för skrivaccess till fasta värden för kod resp för data. Detta förhindrar oavsiktliga skrivningar, men samtidigt också möjligheten att senare uppgradera koden utan att göra ändringar på kretskortet. En variant kan vara att använda byglingar, som medger förändringar men fortfarande kräver manuella ingrepp.

En mer flexibel metod är att ansluta Write Access-ingångarna till en MCU, kanske via annan logik. Om dessa ingångar sätts låga vid power-on-reset kommer hela kretsen att vara skrivskyddad. Programmerbart kan systemet därefter ändra dessa inställningar, och antingen ge full skrivaccess till vissa sektorer eller dynamiskt styra accessen till alla sektorer. På så vis går det att skydda inte bara kod, utan även kritiska områden med data.

MIKE ALWAIS, RAMTRON