Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Möglichkeiten der Nutzung von RRAM in Low-Power Microcontrollern

Frank Vater [1], Goran Panic [2], Mario Schölzel [3]

Abstract: Die Energieeffizienz aktueller Microcontroller wird durch die aggressive Nutzung von verschiedenen Ruhe- und Schlafmodi erreicht. Eine vollständige Abschaltung des Microcontrollers ist dabei aufwendig, da die Zustände in flüchtigen Speichern nicht beibehalten werden. Nichtflüchtige Speicher bieten die Möglichkeit auf einfach Weise die Daten zwischenzuspeichern. In diesem Paper werden die verschiedenen Speichertechnologien in Hinblick auf ihre Energieeffizenz unter Berücksichtigung von Schreib-/Lesegeschwindigkeit sowie Lebensdauer untersucht. Dabei findet eine sehr genaue und technologienahe Simulation auf Basis von Transistormodellen statt.

Keywords: Microcontroller, Low Power, RRAM, Schattenregister, Registerfile

1 Einleitung

Microcontroller und Mikroprozessoren finden sich vielerorts wieder. Das kann für einfache Steuerungs- und Regelungsaufgaben sein bis hin zu komplexen Prozesssteuerungen. Häufig werden dafür Prozessoren mit 8, 16 oder 32 Bit Datenbreite benutzt. Während es z.B. in der Prozessautomatisierung keinerlei Probleme bereitet den Prozessor permanent mit elektrischer Energie zu versorgen, ist das bei einem Einsatz z.B. in einem Sensornetzwerk auf dem freien Feld als Bodensensor schwieriger. Gerade wenn eine Vielzahl von Geräten ausgebracht wurde, können diese unmöglich in kurzen Zeitabständen eingesammelt und die Batterien ausgetauscht oder Akkus aufgeladen werden. Daher ist es unabdingbar Stromsparmechanismen in solchen Geräten einzusetzen. Obwohl gerade kleinere Prozessoren der Entwicklung nach dem Mooreschen Gesetz hinterherhinken, sind hier nun auch Strukturgrößen von 250nm und deutlich kleiner erreicht. Dabei tritt verstärkt der Effekt der Leckströme auf.

In Microcontrollern werden unterschiedliche Speichertypen eingesetzt. Die Abb. 1 zeigt dabei die Speicherarchitektur eines typischen Microcontrollers. Der Flash dient dabei zum Ablegen des auszuführenden Programms. So kann Prozessor von der Energieversorgung getrennt werden ohne dass das Programm verloren geht. Teure, im Sinne von energieintensiven, Schreibzugriffe sind recht selten und treten nur beim Programmieren des Microcontrollers oder bei einem Code-Update auf. Da im Feld ein Code-Update selten vorkommt, sind vornehmlich die Leseoperation und der Leckstrom aus energetischer Sicht interessant.

[1] IHP GmbH, Systeme, Im Technologiepark 25, 15236 Frankfurt(Oder), vater@ihp-microelectronics.com
[2] IHP GmbH, Systeme, Im Technologiepark 25, 15236 Frankfurt(Oder), panicl@ihp-microelectronics.com
[3] IHP GmbH, Systeme, Im Technologiepark 25, 15236 Frankfurt(Oder), schoelzel@ihp-microelectronics.com

Neben dem Programm- wird auch Datenspeicher benötigt, der als SRAM umgesetzt wird. In diesem werden Variableninhalte abgelegt, die während der Abarbeitung komplexerer Programme benötigt werden und die nicht mehr im Registerfile abgelegt werden können.

Im unteren Speicherbereich befinden sich ein “memory-mapped” IO. An diesen sind die peripheren Komponenten angeschlossen, um somit einen einfachen Zugriff darauf zu ermöglichen. Diese Adressen beherbergen keinen “echten” Speicher, vielmehr sind sie eine Schnittstelle zu anderen Komponenten. Beispielsweise können so GPIOs programmiert oder serielle Schnittstellen für den Datenaustausch bedient werden.

Ein weiterer Speicher im Microcontroller, das Registerfile, ist nicht direkt adressierbar und daher auch nicht in Abb. 1 dargestellt. Das Registerfile ist typischerweise aus Flipflops aufgebaut.

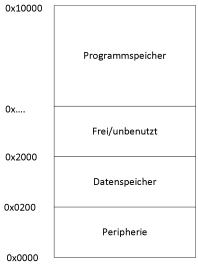


Abb. 1: Speicheraufteilung eines Microcontrollers

Als Beispiel für einen Microcontroller kann der MSP430 der Firma Texas Instruments (TI) dienen [Te15a]. Dieser hat einen 16 Bit breiten Datenpfad und verfügt über einen Adressraum von 64kByte mit getrennten Programm- und Datenspeicher. Aufgrund der niedrigen Taktraten und dem Einsatzgebiet als Sensorknoten ist das Design ohne Cache ausgeführt. Dem Anwendungsgebiet entsprechend sind verschiedene Stromsparmodi implementiert. Diese werden anhand des MSP430 von TI, Familie 5, hier kurz vorgestellt [Te15b]. Neben dem Arbeitsmodus (“active”) stehen die Schlafmodi LPM0 bis LPM4 zur Verfügung. Diese unterscheiden sich untereinander im wesentlichen durch das Abschalten der drei Clock Domänen.

Darüber hinaus existieren noch die Modi LPM 3.5 und 4.5. In diesen Tiefschlaf-Modi verlieren alle flüchtigen Speicher ihren Inhalt. Damit sind RAM und Registerinhalte beim Erwachen aus dem Tiefschlaf gelöscht. Die folgende Startsequenz entspricht damit also dem Ablauf wie bei einem regulären Power-On oder Reset. Messwerte oder Variablen, die nicht im Flash gesichert wurden, sind damit verloren. Ebenso muss somit die Initialisierungsphase wieder durchlaufen werden.

In der Entwurfsphase eines Microcontrollers wird dieser in einer Hardwarebeschreibungssprache wie VHDL implementiert. Daraus wird später eine Netzliste generiert und ein Layout überführt. Der benötigte Speicher wird mit Hilfe eines technologieabhängigen Speichergenerators erzeugt. Dieser fertige Block wird vom Chipdesigner in das Layout eingebunden. Eine Simulation der Leistungsaufnahme des Gesamtsystems, also Digitalzellen, Speicher und I/O Pads, ist sehr zeitaufwendig und kann etliche Tage bis Wochen

in Anspruch nehmen. Darüber hinaus verfügt der Chipdesigner oft nicht über alle notwendigen Bibliotheken (Transistormodelle) der jeweiligen Technologie, die für die Chipfertigung benutzt wird. In einer 0.25 μm Technologie des IHP [IH15] sind sowohl die Speicher als auch die Transistormodelle vorhanden, so dass eine Simulation möglich ist. Die Simulation der einzelnen Speicherblöcke, insbesondere in Hinblick auf die Leistungsaufnahme, kann als Grundlage für eine genauere Bestimmung des Energieverbrauchs eines ASIC genutzt werden. Auf dieser Basis wird eine Exploration der verschiedenen Speichertypen vorgenommen.

2 Related Work

2.1 Stromsparmechanismen

Im Gegensatz zu anderen Halbleitertechnologien ist die CMOS Technologie von Hause aus sehr energieeffizient. Die dynamischen Ströme fließen kurzzeitig bei Schaltvorgängen und die statischen Ströme beschränken sich auf Leckströme. Anfänglich arbeitete die CMOS Technologie bei einer Spannung von 5 V. Mit kleineren Strukturgrößen war es möglich, die Betriebsspannung immer weiter abzusenken, so dass bei Strukturgrößen um 65 nm Spannungen um 1 Volt genügen. Da die Spannung quadratisch in die Leistungsaufnahme eingeht, ist das Einsparpotential bei dynamischen Umschaltvorgängen erheblich. In Designs mit unterschiedlichen Anforderungen an die Verarbeitungsgeschwindigkeit, können die Blöcke einzeln regulierbare Spannungsversorgungen erhalten. So kann z.B. ein langsamer Peripherieblock mit niedrigerer Spannung arbeiten als die Ausführungseinheit der CPU, welche auf schnelle Befehlsabarbeitung optimiert ist.

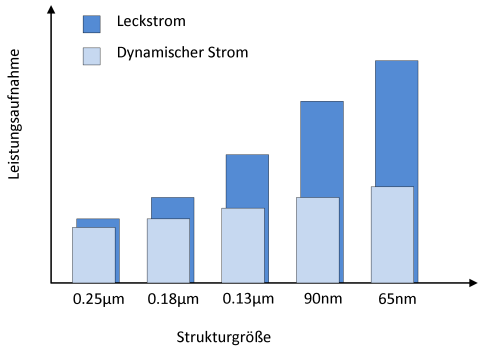


Abb. 2: Dynamischer und statischer Strom in Abhängigkeit von der minimalen Strukturbreite

Gerade der Bereich des dynamischen Schaltens bietet Potential zur Energieeinsparung, welches z.B. durch Clock Gating oder Data Gating umgesetzt werden kann. Mithilfe von Clock Gating können nicht genutzte Partitionen in einem ASIC gezielt von den Umschaltvorgängen ausgeschlossen werden. Da das Clocknetzwerk an jedes Flipflop angeschlossen ist, finden einerseits für das Clocknetzwerk (Treiber) und andererseits in den Flipflops Umschaltvorgänge statt, die Energie verbrauchen. Sollte nun ein Teil eines Design nicht an der Datenverarbeitung beteiligt sein, so wird das Teilnetz der Clock nicht geschaltet. An dieser

Stelle ist die Unterstützung durch die Syntheseprogramme soweit fortgeschritten, dass der Clock Gating Mechanismus fehlerfrei automatisch implementiert wird.

Jedoch steigen im Submicron-Bereich die Leckströme erheblich an. Grund dafür sind die geringeren Strukturgrößen sowie die geringen Thresholdspannungen. Die Abb. 2 zeigt die Entwicklung von dynamischen und statischen Strömen bei kleiner werdenden Strukturgrößen. Vor allem im Einsatzbereich von Sensorknoten, welche über einen langen Zeitraum energieautark fungieren müssen, ist ein hoher Stromverbrauch im Standby nicht praktikabel. In kurzer Zeit wäre das Lebensende bei einem batteriebetrieben Gerät erreicht. Aktuell sind die Strukturgrößen von Microcontrollern für Sensorknoten im Bereich von 180 bis 130 nm, wobei auch hier der Trend aus Kostengründen zu immer kleineren Strukturen geht. Ab etwa 90nm werden die Leckströme so dominant, dass dafür geeignete Mittel gefunden werden müssen, diese auf ein Minimum zu beschränken.

Ein effizientes Mittel gegen hohe Leckströme ist der Einsatz der Power Gating Technik [Pa14]. Bei dieser werden die Blöcke in einem Design separat mit Energie versorgt. Im Gegensatz zum Clock Gating ist beim Power Gating ein aktives Eingreifen durch den Nutzer/Programmierer sinnvoll. Da dieser den Anwendungszweck kennt, kann er aus seiner Position heraus gut abschätzen, welche Blöcke komplett abgeschaltet werden können, so dass der Leckstrom auf Null sinkt.

Der Nachteil des Power Gating begründet sich darin, dass Registerinhalte von flüchtigen Speichern verloren gehen können. Soll ein Microcontroller mit Power Gating in einen sehr energiearmen Schlafmodus versetzt werden, so ist darauf zu achten, dass das Registerfile in einem nichtflüchtigen Speicher gesichert wird. Beim Erwachen, ausgelöst durch einen Interrupt, verweist der Programmzähler auf eine Adresse zur Interruptbehandlung. Innerhalb dieser Routine muss ein gesicherter Zustand aufwendig wiederhergestellt werden. Dieses kostet Zeit und der zusätzliche Aufwand ist nicht erwünscht.

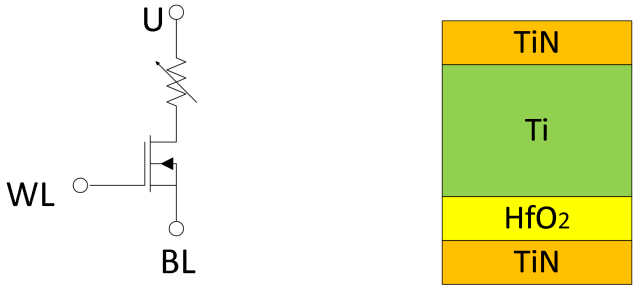
2.2 RRAM

Die Möglichkeit eine Information mit Hilfe eines änderbaren Widerstands zu speichern, ist schon seit den 1960er Jahren bekannt [NB64]. Jedoch erst mit der in [Le08] beschriebenen Lösung war es möglich, diese Art Speicherelement in die siliziumbasierten Chipproduktion zu integrieren.

In Abb. 3a wird der Schaltplan eines Bits in RRAM Technologie gezeigt. Dem Transistorausgang ist ein änderbarer Widerstand nachgeschaltet. Dieser Widerstand kann durch einen kurzen elektrischen Impuls seinen Wert verändern. Die Widerstandsänderung bleibt auch nach Abschalten der Versorgungsspannung erhalten. Auf diese Weise kann entweder eine logische Null oder Eins nichtflüchtig abgespeichert werden. Konstruktiv wird das mittels eines speziell präparierten Vias gelöst. In Abb. 3b ist der vertikale Aufbau eines Transistors mit dem dazu gehörigen modifizierten Via zu sehen. Das Standard-Via, bestehend aus Titan (Ti) und Titannitrit (TiN), befindet sich zwischen Metal2 und Metal3. Das unteren Ende des Vias wird durch das Einbringen von Hafniumoxid (HfO_2) modifiziert, welches so die Funktionalität des veränderbaren Widerstands ermöglicht. Da der RRAM

im Backend, also erst in der Verdrahtungsebene, eingebracht wird, ist diese Technologie vollständig CMOS kompatibel.

Nach der Produktion des Speichers, muss dieser einmalig initialisiert werden. Dazu wird ein Strom von 10μA an jede Speicherzelle angelegt. Da diese Initialisierungsphase auf einem Tester durchgeführt werden kann, wird dieser Energieverbrauch nicht weiter im funktionalen Betrieb des Microcontrollers betrachtet.



(a) Schaltplan einer RRAM Zelle

(b) Querschnitt durch ein Via, dass als RRAM Zelle ausgeführt ist

Abb. 3: RRAM Zelle

3 Anwendungsmöglichkeiten

Unabhängig vom verwendeten Microcontroller, ist der Ablauf der Programmabarbeitung stets ähnlich. Nach dem Power-On oder Reset wird das erste Datenwort von einer Basisadresse abgeholt. Diese Adresse liegt typischerweise innerhalb eines nichtflüchtigen Speichers wie einem Flash. Im weiteren wird das dort abgelegte Programm abgearbeitet, das zunächst aus einem Abschnitt mit Konfigurationsdaten besteht. Dabei werden Timer, GPIOs etc. programmiert. Daran schließt sich die Abarbeitung einer Funktion an, die der Hauptbestandteil des Programms ist. Z.B. können Messwerte aufgenommen, verarbeitet und drahtlos an benachbarte Knoten gesendet werden.

Aus Energiespargründen ist es sinnvoll, Messwerte nicht direkt weiter zu leiten. Gegebenenfalls werden Daten in Registern oder im RAM abgelegt, kumuliert und erst nach einer gewissen Zeit gesammelt weiter versendet. Nach dem Erfassen von Daten wird der Microcontroller in einen Schlafmodus versetzt, um so die Lebensdauer der Batterie zu verlängern. Dieser Energiesparmodus (z.B. LPM 0 bis LPM 4 beim MSP430) wird durch ein Ereignis an einen GPIO Pin oder nach Ablauf eines Timers beendet.

In Abhängigkeit des Anwendungsszenarios kann der Zustand des Schlafmodus für Sekunden, Minuten, Stunden oder noch länger beibehalten werden. Je länger die Schlafzeit ist, umso mehr lohnen sich die Tiefschlafmodi wie der LPM 3.5 und LPM 4.5 beim MSP430. Bei diesen wird die Leistungsaufnahme durch das Abschalten der Spannungsversorgung für Registerfile und RAM sehr weit heruntergefahren. Es können jedoch Fälle eintreten, in denen der Wechsel in diese Modi sehr aufwendig ist. Werden in den Registern oder

im RAM Daten vorgehalten, die beim nächsten Erwachen zur Verfügung stehen sollen, müssen diese in einem nichtflüchtigen Speicher gepuffert werden. Hierbei kann der Einsatz von RRAM die Sicherungsoperation deutlich vereinfachen.

4 Ersetzungsszenarien

Wie in Abschnitt 1 beschrieben, enthält ein Microcontroller verschiedenartige Speichertypen. Da es sich beim RRAM um einen nichtflüchtigen Speicher mit schnellen Zugriffen vor allem beim Lesen handelt, können Überlegungen angestellt werden, alle Speicher durch Speicher diesen Typs zu ersetzen oder zumindest zu ergänzen. Dieser Ansatz kann dann zum Einsatz kommen, wenn der Microcontroller z.B. auf Basis des openMSP430 [Gi09] für ein bestimmtes Projekt gezielt entworfen und gefertigt wird. Sofern seitens der Foundry verschiedene Speichertechnologien angeboten werden, kann der Designer entscheiden, ob z.B. RRAM letztendlich zum Einsatz kommt.

In der Abb. 4 sind die möglichen Szenarien dargestellt, den RRAM in einem Microcontroller einzusetzen. Zunächst ist naheliegend den, als Programmspeicher genutzten, Flash (Abb. 4(a)) durch RRAM (Abb. 4(b)) zu ersetzen. Beispielsweise hat Panasonic bereits den Microcontroller MN1010L im Angebot [Pa15], bei welchem der Programmspeicher in RRAM Technologie ausgeführt ist. Da die notwendige Leistung zum Speichern eines Bits deutlich unter dem des Speicherns in einem Flash liegt und darüber hinaus auch das aufwendige Löschen von Flashsegmenten entfällt, kann das als Fortschritt betrachtet werden.

In der nächsten Stufe kann auch der SRAM durch einen RRAM ersetzt werden (Abb. 4(c)). Dafür sind jedoch neben der Zugriffszeit und der notwendigen Enerige auch die Anzahl der Schreibzyklen entscheidend. In der Literatur wird die Schreibzeit im Nano- bis Mikrosekunden angegeben und die Anzahl der möglichen Schreibzyklen liegt zwischen 10^6 und 10^9. Läuft also der Microcontroller mit einer Taktfrequenz von 10 MHz und es finden in der aktiven Phase nur wenige Schreibzyklen auf die gleiche Adresse im RAM statt, so kann der SRAM durch RRAM ersetzt werden. Allerdings eventuell muss ein höherer Energiebedarf in der aktiven Phase mit in die Betrachtung einfließen.

In letzter Konsequenz kann auch das Registerfile durch einen RRAM ersetzt werden (Abb. 4(d)), um den Bootprozess nach dem Aufwachen aus einem Tiefschlafmodus zu vermeiden. Jedoch sind die Anzahl der Schreibzyklen während der Abarbeitung eines Programms recht hoch, so dass nach kurzer Zeit keine zuverlässige Speicherung von Daten im RRAM mehr möglich ist. Der Programmzähler ist dabei der kritischste Punkt. Bei einem Dutycycle, also einer aktiven Zeit des Microcontrollers, von 0,1% und möglichen 10^9 Schreibzyklen, ergibt sich eine Lebensdauer von etwa 28 Stunden für den Microcontroller. Anschliessend ist der RRAM in seiner Funktion nicht mehr zuverlässig. Daher ist dieser Ansatz den RRAM als Ersatz für ein Registerfile zu integrieren nicht praktikabel.

Alternativ soll daher betrachtet werden, den RRAM als Schattenregister zu verwenden, wie in Abb. 4(e) gezeigt. Dieses ist parallel zum Registerfile in dem Microcontroller eingebaut. Jeder Schreibzugriff auf das Registerfile wird nicht unmittelbar auf das Schat-

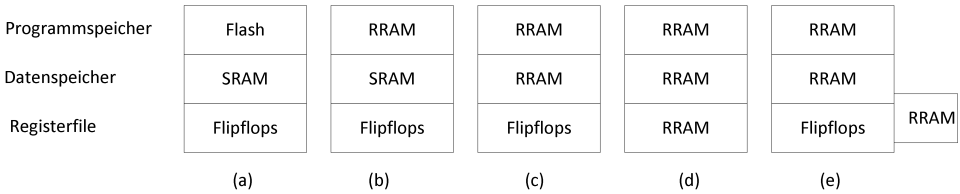


Abb. 4: Ersetzungsstrategien von Speicher in einem Microcontroller durch RRAM

tenregister in RRAM Technologie übertragen. Vielmehr erfolgt dies erst beim Übergang in einen der Schlafmodi, die auch die Spannungsversorgung für SRAM und Registerfile abschalten.

Um auch das Registerfile von der Spannungsversorgung trennen zu können und dennoch die Programmabarbeitung an definierter Stelle fortsetzen zu können, müssen die Registerinhalt also in einem nichtflüchtigen Speicher gesichert werden. Das Ablegen von Werten in nichtflüchtigen Speichern, und ebenso das Zurückschreiben in die Flipflops, benötigt den Einsatz einer gewissen Energiemenge. Bevor also der Prozessor in den Tiefschlafmodus versetzt wird und die Registerinhalte in einen permanenten Speicher überführt werden, muss abgeschätzt werden, ob der Energieaufwand für das Speichern und Wiederherstellen der Werte diesen Schritt rechtfertigt. Das heißt also, dass die Energie die innerhalb des Registerfiles für die voraussichtliche Schlafdauer für Leakage verbraucht wird höher sein muss, als für das Backup. Da diese Werte erheblich von der verwendeten Technologie und der Technik des nichtflüchtigen Speichers abhängen, kann keine pauschale Aussage getroffen werden, ab welcher Schlafdauer sich der Einsatz eines solchen Ansatzes lohnt. Der Nutzer muss also anhand von Programm und den verfügbaren Speichern eine Abschätzung treffen.

Das Schattenregister und der zugehörige Controller können mit Power Gating in das Design integriert werden. So tritt im aktiven Modus keine Belastung der Energiebilanz durch die zusätzliche Komponente auf. Zusätzlich kann das Schattenregister eingesetzt werden, um Checkpoints zu setzen. Damit es möglich, den Programmablauf an einer definierten Stelle fortzusetzen falls z.B. ein Watchdog eine Endlosschleife erkennt und eigentlich ein Reset auslösen müsste.

Die notwendige Energie zum Speichern eines Bitwertes in einer RRAM Zelle liegt aktuell deutlich über der nötigen Energie für ein Flipflop (vgl. Tab. 1). In der Simulation wurde eine um Faktor 10 höhere Energie für das Schreiben eines Wertes und ein Faktor 400 für das Lesen ermittelt. Die Register sind in beinahe jedem Takt des Microcontrollers mit Lese- oder Schreibzugriffen belastet. Während die Anzahl der Lesezugriffe keinen Einfluss auf die Lebensdauer der RRAM Zelle hat, verkürzt jeder Schreibzugriff diese. Dieser Sachverhalt ist bei der Implementierung zu beachten.

5 Simulation der Leistungsaufnahme

Grundsätzlich sind zwei Arten der Leistungsaufnahme zu berücksichtigen, der aktive Modus und der Ruhemodus. Bei Sensorknoten zählt insbesondere der Ruhemodus, der durch Leckströme bestimmt ist, da die Geräte sehr lange in Standby sind, jeweils unterbrochen von kurzen Wachphasen. Allerdings soll auch in den kurzen Wachphasen der dynamische Stromverbrauch möglichst gering sein.

Ein Microcontroller ist in der Anzahl der Transistoren, verglichen mit einem High-End Prozessor, überschaubar. Dennoch ist es nur mit sehr großen Aufwand möglich, diesen Prozessor analog zu simulieren. Eine solche Simulation wäre zwar sehr genau in bezug auf die Leistungsaufnahme, doch ist mit einer Simulationszeit von mehreren Tagen zu rechnen. Für die Bewertung, ab welcher Schlafdauer es sich lohnt auf einen nichtflüchtigen Zwischenspeicher zurückzugreifen, genügt es die Leistungsaufnahme der jeweiligen Speicher zu kennen.

In einer gut charakterisierten Technologie ist eine analoge Simulation des Speichers in Bezug auf Zeitverhalten und Energieverbrauch sehr nahe an den realen Messwerten für die Leistungsaufnahme von dem gefertigten ASIC. Daher kann auf diese Simulationsergebnisse für die Betrachtung des Energieverbrauchs zurückgegriffen werden.

5.1 Digitale Standardzellen

Ein typisches Registerfile kann aus Flipflops aufgebaut werden. Dieses wird zunächst in einer Hardwarebeschreibungssprache beschrieben und anschließend in der Synthese in eine Gatternetzliste überführt. In diesem Schritt ist eine sehr grobe Abschätzung der Leistungsaufnahme möglich. Dabei schätzt das Synthesetool die Anzahl der Umschaltvorgänge von Gattern oder es lässt sich ein Umschaltwert vom Nutzer angeben.

Eine sehr genaue Abschätzung der Leistungsaufnahme ist mit dem Tool “PrimeTime” der Firma Synopsys möglich [Sy15]. Dazu wird zunächst der Prozessor mit dem Programm simuliert, das später im Feld eingesetzt werden soll. Im Gegensatz zu einer üblichen Simulation werden bei dieser Art von Simulation alle Signaländerungen in einer Datenbank gespeichert. Da hierbei einige tausend bis zehntausend Signale und deren Änderungen mit Zeitstempel protokolliert werden müssen, kann die Datenbank eine Größe von mehreren Gigabyte schnell überschreiten. Diese Datenbank kann in das Programm “PrimeTime” eingelesen werden. Dazu wird noch eine technologieabhängige Datenbank eingelesen, die für jeden Gattertyp die Umschaltcharakteristik einschliesslich der Leistungsaufnahme kennt. So kann eine sehr exakte Abschätzung der Leistungsaufnahme erfolgen, deren Grundlage das Anwenderprogramm ist.

Da der Leakagestrom auch von den Eingangswerten eines Gatters abhängig ist, wird dieser mit Hilfe von “PrimeTime” ebenso genauer bestimmt als im Synthesetool. Während für Strukturgrößen bis 250nm diese Unterschiede kaum zum Tragen kommen, sind diese bei 130nm und kleiner unter Umständen erheblich (vgl. Abb. 2).

5.2 Speicher

Microcontroller besitzen flüchtigen und nichtflüchtigen Speicher. Ist der flüchtige Speicher nicht mit Flipflops implementiert, wird mit Hilfe eines technologieabhängigen Generators jeweils ein Hardmacro[4] vom Speicher erzeugt. Dazu wird ein Verhaltensmodell, ein Layout sowie eine Netzliste für die Verifikation des Layouts generiert. Diese Netzliste, z.B. im SPICE Format, kann für eine analoge Simulation des Speichers benutzt werden. Auf diese Art kann sehr genau die Energie z.B. für einen Schreib- oder Lesezugriff ermittelt werden. Der zeitliche Aufwand dafür ist sehr hoch, so dass tatsächlich nur wenige Testfälle simuliert werden können und von diesen aus auf die durchschnittliche Leistungsaufnahme je Operation geschlossen werden. In Abb. 5 ist das Ergebnis einer beispielhaften Simulation eines SRAM Modells zu sehen.

Die nichtflüchtigen Speicher funktionieren nach dem Prinzip, dass ein physikalischer Effekt wie die permanente Widerstandsänderung ausgenutzt wird. Dieses ist nicht mit SPICE simulierbar. Anstelle dessen muss ein VerilogA Modell benutzt werden, welchse diese Aufgabe übernimmt. Dieses VerilogA Modell wird ausschließlich für die Simulation des veränderlichen Widerstands eingesetzt. Alle anderen Komponenten des Speichers werden, wie gehabt, als SPICE Netzliste simuliert. Somit ist auch eine genaue Simulation der Leistungsaufnahme dieses Speichertyps möglich.

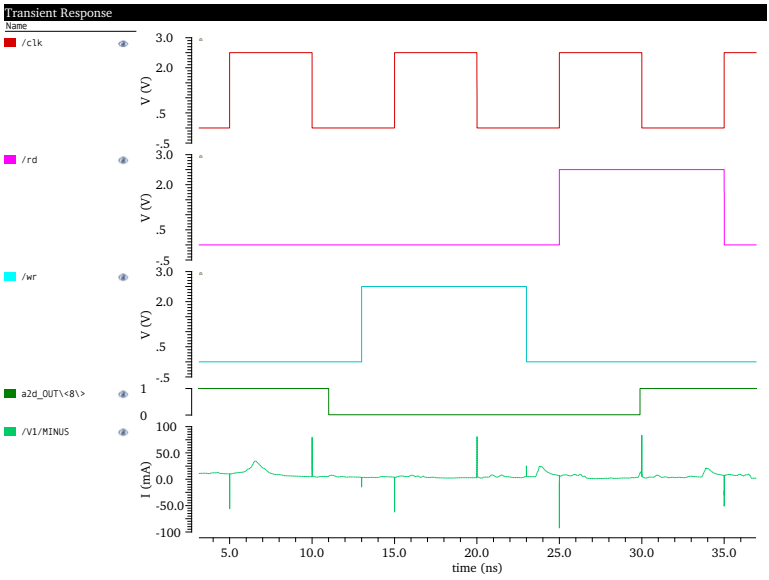


Abb. 5: Analoge Simulation eines SRAM

[4] Fertiges Modul auf Layoutebene

6 Simulationswerte

Da, wie in Abschnitt 5, beschrieben die Simulation der Leistungsaufnahme eine sehr gute Abschätzung des Energieverbrauchs liefert, kann auf dessen Grundlage der Speicher ermittelt werden, der am besten für den geplanten Anwendungsfall geeignet ist.

In Tab. 1 sind die Simulationswerte für die verschiedenen Speichertypen dargestellt. Für alle Speichertypen wurden mit auf Basis der SPICE Netzliste je ein Schreib- und Lesevorgang simuliert. Die Abb. 5 zeigt beispielhaft die grafische Darstellung des Simulationsergebnisses für einen SRAM. Die oberen vier Vektoren sind Clock, Lese- und Schreibsignal sowie der Ausgang eines ausgewählten Datenbits. Der untere Vektor ist die Leistungsaufnahme des Schaltkreises. Zur besseren Vergleichbarkeit sind alle Werte, unabhängig von der eigentlichen Speichergröße, für ein Bit angegeben. Für Flash und RRAM sind keine Werte für die Leckströme angegeben, da diese im Schlafmodus vollständig von der Spannungsversorgung getrennt werden können und im Betrieb die Leckströme eine untergeordnete Rolle spielen.

Speichertyp	Energie (Schreiben)	Energie (Lesen)	Leakage
Flipflop in Registerfile	0,195pWs	0,07pWs	250 pW
SRAM (aus 2kB)	10,7pWs	10,03pWs	45 pW
Flash (aus 64kb)	13400pWs	31,25pWs	0 pW
RRAM (aus 4kbit)	2pWs	31pWs	0 pW

Tab. 1: Energie zum Lesen/Speichern eines Bit für verschiedene Speichertypen in einer 0.25μm Technologie

Die Evaluierung aus Tab. 1 ergibt, dass in jedem Fall der Programmspeicher in Flash-Technologie durch einen RRAM ersetzt werden kann zugunsten der Energiebilanz. Während die Energie zum Lesen etwa gleich ist, benötigt der Flash ein vielfaches an Energie zum Abspeichern eines Bits. Dies ist durch die unterschiedlichen Ansätze bei der Speichertechnologie bedingt. Beim Flash ist der Transistor mit einem doppelten Gate ausgestattet und für die Platzierung der Elektronen zwischen den beiden Gates ist eine hohe Spannung für einen definiert langen Zeitraum notwendig. Beim RRAM dagegen wird ein Widerstand mit geringer Spannung und in vergleichsweise kurzer Zeit verändert.

Ein mögliches Szenario ist auch der Ersatz des SRAM, der als Datenspeicher dient. Während die Energie zum Schreiben eines Werte im SRAM höher ist als für den RRAM (vgl. Tabelle 1), ist die Energie zum Lesen deutlich geringer. Es gibt Anwendungen, wie Verschlüsselungsalgorithmen, welche eine Lookup-Tabelle benutzen. Diese ist im Datenspeicher abgelegt. Auf diese Daten wird stets lesend zugegriffen, so dass ein SRAM energiesparender sein kann als ein RRAM. Im Gegensatz zu Anwendungen, die weitgehend lesend auf den Datenspeicher zugreifen, gibt es Anwendungen die wechselweise lesend und schreibend den Datenspeicher nutzen, wie z.B. Sortieralgorithmen. In diesem Fall ist der Einsatz des RRAM sinnvoll. Neben der Betrachtung der Energie für die Lese- und Schreibzyklen ist auch der Leckstrom zu berücksichtigen. Unter Umständen ist der Energieaufwand für das Schreiben und Lesen des RRAM höher als beim SRAM. Dieses kann über die Zeit, die der Knoten im Schlafmodus ist, kompensiert werden. Anhand der Formel 1 kann der Anwender abschätzen, ob der Aufwand für den Einsatz eines RRAM gerecht-

fertigt ist, dabei ist der Speicher Typ 1 der SRAM und Typ 2 der RRAM. Mit der Annahme, dass einmal pro Aktivphase auf den Datenspeicher lesend und schreiben zugegriffen wird, ergibt sich eine Zeit von etwa 0,3s, die der Knoten in Ruhemodus sein muss, damit der höhere Energieaufwand für die Nutzung des RRAM kompensiert ist. Finden zum Beispiel in der Aktivphase ein Schreibzugriff und fünf Lesezugriffe statt, so ist eine Schlafzeit von etwa 2s notwendig. Ist die notwendige Dauer des Ruhemodus zur Kompensierung des höheren Energieaufwands größer als die Anwendung in ihrem Einsatzgebiet zulässt, sollte auf den RRAM verzichtet werden, da so die Energieeffizienz nicht verbessert werden kann.

Wenn der Knoten in den Schlafmodus übergeht, muss der Inhalt des Registerfiles im Schattenregister gespeichert werden. Das bedeutet das zunächst das Datum aus dem Registerfile gelesen (1) und ins Schattenregister geschrieben werden muss (2). Beim Aufwachen wird der Wert aus dem RRAM gelesen (3) und in das Registerfile zurückgeschrieben (4). Also müssen bei der Betrachtung der Energiebilanz alle vier Vorgänge berücksichtigt werden. Dagegen steht die Einsparung des Leckstroms. Die Formel 2 beschreibt diesen Vorgang, wobei Speicher Typ 1 das Flipflop-basierte Registerfile und Typ 2 das Schattenregister in RRAM Technologie ist. Mit dem Einsetzen der Parameter ergibt sich die Zeit für den Ruhemodus, ab welcher sich der Einsatz eines Schattenregister aus energetischer Sicht lohnt.

Mit den Werten aus der Simulation (Tab.1) benötigen alle Vorgänge zusammen ca. 33,3 pWs um ein Bit zu speichern und wieder abzurufen. Unter Berücksichtigung des Leckstroms eines Flipflops von 250pW, lohnt sich es ab einer Schlafdauer von 0,13s auf den nichtflüchtigen Speicher zurückzugreifen. Nimmt man den Energieaufwand für einen Controller dazu, welcher das Management des Datentransfers übernimmt, kann man annehmen, dass sich ab einer Schlafdauer im Sekundenbereich der Einsatz von RRAM als Zwischenspeicher sinnvoll erscheint.

$$T = \frac{(n \times E_{Lesen_{Typ_1}} + m \times E_{Schreiben_{Typ_1}}) - (n \times E_{Lesen_{Typ_2}} + m \times E_{Schreiben_{Typ_2}})}{P_{Leckstrom_{Typ_1}}} \tag{1}$$

$$T = \frac{E_{Lesen_{Typ_1}} + E_{Schreiben_{Typ_2}} + E_{Lesen_{Typ_2}} + E_{Schreiben_{Typ_1}}}{P_{Leckstrom_{Typ_1}}} \tag{2}$$

wobei:

n = Anzahl der Lesezyklen in der Aktivphase
m = Anzahl der Schreibzyklen in der Aktivphase
$E_{Lesen_{Typ_1}}$ = Energie zum Lesen eines Bits von Speichertyp 1
$E_{Schreiben_{Typ_1}}$ = Energie zum Schreiben eines Bits von Speichertyp 1
$E_{Lesen_{Typ_2}}$ = Energie zum Lesen eines Bits von Speichertyp 2
$E_{Schreiben_{Typ_2}}$ = Energie zum Schreiben eines Bits von Speichertyp 2
$P_{Leckstrom_{Typ_1}}$ = Leckstrom eines Bit von Speichertyp 1

7 Zusammenfassung

In dieser Arbeit wurde die Möglichkeit betrachtet, neue nichtflüchtige Speicher in Microcontrollern vor dem Hintergrund von verlängerter Batterielebensdauer einzusetzen. Gerade in Anwendungsgebieten mit langen Standby-Zeiten kann so die Energiebilanz verbessert werden. Ab einer Schlafdauer von etwa einer Sekunde ist der zusätzliche Energieaufwand für das Puffern in einem nichtflüchtigen Speichern kompensiert. Wichtig dafür ist eine Applikation, bei der sich die Wach- und Schlafphasen gut abschätzen lassen. Somit werden negative Auswirkungen des zusätzlichen Speichers in RRAM Technologie auf den Energieverbrauch vermieden. Darüber hinaus ist es notwendig mit Hilfe von geeigneten Modellen und den zugehörigen Simulationswerkzeugen den Energieverbrauch für Schreib- und Leseoperationen sowie des Leckstroms der verschiedenen Speichertypen bestimmen zu können.

Literaturverzeichnis

[Gi09] Girard, Olivier: , openMSP430 microcontroller - compatible with the original MSP430 architecture. `http://opencores.org/project,openmsp430`, 2009. [Online; accessed 2015-03-05].

[IH15] IHP Microelectronics: , Desgin Kit. `http://www.ihp-microelectronics.com/en/services/mpw-prototyping/design-kit/design-kit.html`, 2015. [Online; abgefragt 24.4.2015].

[Le08] Lee, H.Y. and Chen, P.S. and Wu, T.Y. and Chen, Y.S. and Wang, C.C. and Tzeng, P.J. and Lin, C.H. and Chen, F. and Lien, C.H. and Tsai, M.J.: Low Power and High Speed Bipolar Switching with a Thin Reactive Ti Buffer Layer in Robust HfO2 Based RRAM. In: Electron Devices Meeting, 2008. IEDM 2008. IEEE International. S. 1–4, Dec 2008.

[NB64] Nielsen, P.H.; Bashara, N.M.: The Reversible Voltage-Induced Initial Resistance in the Negative Resistance Sandwich Structure. Electron Devices, IEEE Transactions on, 11(5):243–244, May 1964.

[Pa14] Panic, Goran: A Methodology for Designing Low Power Sensor Node Hardware Systems. Dissertation, Brandenburgische Technische Universität Cottbus-Senftenberg, 2014.

[Pa15] Panasonic: , MN101L series microcontroller. `http://www.semicon.panasonic.co.jp/en/products/microcomputers/mn101l`, 2015. [Online; abgefragt 27.4.2015].

[Sy15] Synopsys: , PrimeTime. `http://synopsys.com/Tools/Implementation/SignOff/PrimeTime/Pages/default.aspx`, 2015. [Online; abgefragt 27.4.2015].

[Te15a] Texas Instruments: , Low-power MCUs. `http://www.ti.com/msp430`, 2015. [Online; abgefragt 27.4.2015].

[Te15b] Texas Instruments: , MSP430x5xx and MSP430x6xx Family User's Guide. `http://www.ti.com/lit/ug/slau208n/slau208n.pdf`, 2015. [Online; abgefragt 27.4.2015].