Onchip RAM erhöht die Logikkomplexität in FPGA

Steven K. Knapp, Xilinx, und Jutta Ramatschi, Metronik, München

126 Heutzutage werden digitale Systeme in immer kleineren Bauformen hergestellt bei wachsender Integrationsdichte. Darum favorisieren Entwickler zunehmend Masken-Gate-Arrays oder die im System programmierbaren Gate Arrays (FPGA). Sowohl die Gate Arrays als auch die FPGA bieten die besonderen Systemvorteile hochintegrierter VLSI-Bauelemente, die der Entwickler bei den niederintegrierten Bauteilfamilien vermisst: hohe Packungsdichte, geringe Stromaufnahme und stark verbesserte Zuverlässigkeit. Manche Logikfunktionen sind mit Standard-SSI/MSI-Bauteilefamilien auch wirtschaftlich gar nicht zu realisieren. Solche speziellen Logikfunktionen enthalten oft auch Speicherelemente wie Latches, Flipflops und Register.

Register und Zähler werden mit Flipflops aufgebaut. In einem Gate Array wird ein Flipflop mit fünf 2-Eingangs-NAND-Gattern realisiert. RAM-Strukturen von einigen 100 Bit Komplexität, aufgebaut in konventioneller Logik, PLD oder auch in Gate Arrays sind dagegen äusserst unwirtschaftlich. Mit Hilfe von über den Chip verteilten RAM-Zellen wird die Komplexität von Flipflop-intensiven Funktionen auf ein vernünftiges Mass reduziert und der Aufbau von grossen Zählern und Registerfeldern einfacher und wirtschaftlicher. Im typischen Logikdesign werden für die Aufgabe eigentlich überdimensionierte RAM-Bausteine eingesetzt, da RAM-IC mit kleinen Speicherkapazitäten meist teurer und schlechter verfügbar sind als deren grosse Brüder. Die dritte FPGA-Generation bietet RAM-Zellen auf dem Chip, die pro Speicherelement weniger als ein Gatteräquivalent zählen.

Aufbau von RAM-Bereichen

Wie man RAM-Funktionen in unterschiedlichen FPGA realisiert, hängt von der Architektur des Logik-Grundbausteins ab. In den Baustein der XC4000-Serie von Xilinx wird ein RAM-Bereich mit demselben Look-up Table aufgebaut, der auch die logischen Funktionen definiert. Jeder Logikblock der XC4000-Serie stellt entweder zwei 16 x 1-RAM oder ein 32 x 1-RAM (bzw. dementsprechend auch zwei 16 x 1-PROM oder ein 32 x 1-PROM) bereit.

Designbeispiele

Einige Schaltungsbeispiele können am ehesten verdeutlichen, wie kleine, über den Chip verteilte RAM-Funktionen anstelle von Flipflops die Logik einfacher, weniger komplex machen.

Das erste Design, das hier angeführt wird, ist ein einfaches Schieberegister, das zweite ein Zähler-Array. Im nächsten Heft der «Elektronik-Informationen» (Nr. 8/91) wird anhand des dritten und vierten Beispiels gezeigt, wie man Fifo und Lifo mit Hilfe von Single-Port-RAM aufbaut. Das letzte Beispiel (ebenfalls in Nr. 8), zeigt, wie man ein RAM als Register-File in einem High-Speed-DMA-Controller einsetzen kann.

Beispiel 1: Schieberegister

Das erste Beispiel **(Bild 1)** ist ein einfaches 32-Bit-Schieberegister. Die Daten werden seriell eingegeben und 32 Taktzyklen später ausgegeben. Bisher wurden zum Aufbau eines solchen Schieberegisters 32 D-Flipflops verwendet. In Gattern entspricht ein D-Flipflop gleich fünf 2-Input-NAND-Äquivalenten. Somit erfordert ein 32-Bit-Schieberegister 160 Gatter.

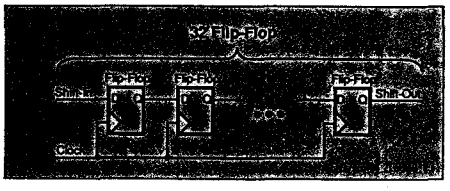


Bild 1 Bei herkömmlichem Aufbau benötigt ein 32-Bit-Schieberegister 32 Flipflops, entsprechend 160 2-Eingangs-NAND -Gatteräquivalenten.

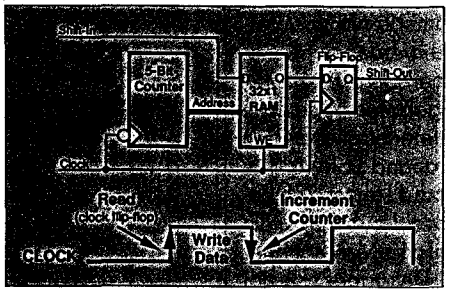


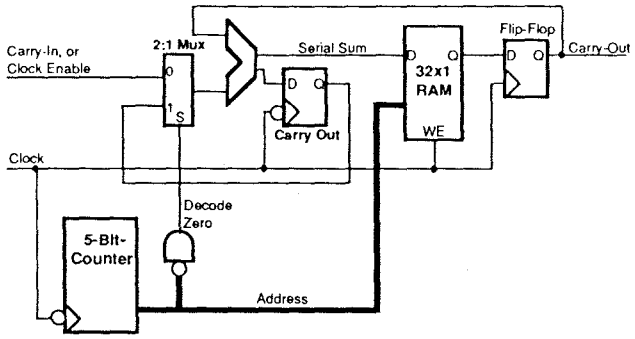
Bild 2 Eine neuartige Realisierung eines 32-Bit-Schieberegisters mit Onchip-RAM anstelle einzelner Flipflops.

Realisiert man dieses Schieberegister mit Xilinx-LCA der XC3000-Serie, so belegt es 16 Logikblöcke (CLB), denn jeder Logikblock enthält zwei Flipflops. Eine Alternative zu diesem Flipflop-intensiven Aufbau bietet die bereits angesprochene RAM-Realisierung mit den Look-up Tables der XC4000-Serie. Diese neuartige zweite Methode ist in **(Bild 2)** dargestellt.

Das 32-Bit-RAM ist der Speicher für das 32-Bit-Schieberegister. Ein 5-Bit-Zähler dient zum Adressieren des RAM, wobei die Zählersequenz – ob ein einfacher Binärzähler oder ein lineares Feedback-Schieberegister – nicht weiter wichtig ist. Ein einfaches D-Flipflop speichert den Ausgang des RAM. Die RAM-Implementation beruht auf einer Read-Modify-Write-Sequenz. Mit der ansteigenden Taktflanke werden Daten vom RAM in das D-Flipflop geschrieben. Das logische High des Taktsignals steuert das Write Enable des RAM und speichert die Daten bei Shift-in in den gewählten RAM-Bereich. Der Inhalt eines RAM-Bereichs wird also erst aus dem Flipflop beschrieben, und danach werden neue Daten in diesen Bereich abgespeichert. Die fallende Flanke des Clock Inputs erhöht den Zähler zum nächsten RAM-Bereich. Dieses Beispiel beschreibt ein 32-Bit-Schieberegister; es können natürlich Schieberegister jeder beliebigen Grösse nach dieser Methode aufgebaut werden. Wegen der festen Logikstruktur von FPGA ist die RAM-Lösung ideal für programmierbare Bausteine. Für ein maskenprogrammiertes Gate Array bietet eine RAM-Lösung keinen Vorteil; die Grundzelle eines Gate Arrays – ein 2-Input-NAND-Gatter – eignet sich eher zum Aufbau einfacher Schieberegister.

Beispiel 2: 32-Bit-Binärzähler

Für den Aufbau breiter Zähler kann das gleiche Konzept verwendet werden, das schon für den Aufbau von Schieberegistern beschrieben wurde. Zähler sind weitaus komplexer als Schieberegister. Jedes Bit eines Binärzählers erfordert mehr als zehn Gatteräquivalente. Eine RAM-Lösung bietet sowohl für anwenderprogrammierbare als auch für maskenprogrammierte Gate Arrays viele Vorteile. Der Schaltungsauf-



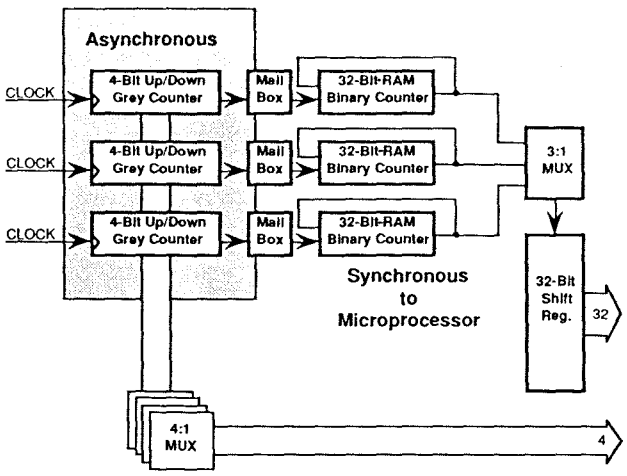


Bild 3 Ein relativ langsamer, aber platzsparender Binärzähler, der das Schieberegister aus Schaltbeispiel 1 verwendet (oben).

Bild 4 Eine schnelle 3-Achsen-Steuerlogik für Roboteranwendungen, äusserst platzsparend und zuverlässig aufgebaut mit Hilfe von Onchip-RAM (rechts).

bau ist angelehnt an das vorangegangene Design. Der Binärzähler ist wie ein Schieberegister aufgebaut, das seriell aufaddiert. Es besteht aus einem Halbaddierer mit Carry-Flipflop **(Bild 3).**

Wie im Schaltbeispiel des Schieberegisters wird der Inhalt des aktuellen RAM-Bereichs bei steigender Clock-Flanke in ein Flipflop geschrieben. Wenn der Adresszähler auf Null steht, selektiert der 2-zu-1-Multiplexer entweder den Carry-in- oder den Clock-Enable-Eingang. Der Flipflop-Wert wird dann dem Wert des Carry-in- oder Clock-Enable-Signals aufaddiert und in den aktuellen RAM-Bereich zurückgeschrieben. Bei fallender Taktflanke wird das Carry-Signal des seriellen Addierers in ein Flipflop gespeichert, und der Adresszähler wird auf den nächsten Bereich erhöht. Bei >0 sorgt ein Null-Detect-Decoder dafür, dass der 2-zu-1-Multiplexer das Carry-Flipflop selektiert. Der Inhalt des nächsten RAM-Bereichs wird dem Carry-Wert des vorangegangenen Zyklus aufaddiert. Dieser Read-Modify-Write-Vorgang läuft so lange weiter, bis alle Bereiche im RAM aufaddiert sind. Bei Beginn des nächsten vollen Zyklus selektiert der Multiplexer wieder den Carry-in-Eingang. In diesem Beispiel muss der Zähler mindestens 32mal so schnell wie die ankommenden Zählerdaten getaktet werden. Dies begrenzt die hier vorgestellte Implementation auf etwa 1 MHz. Man kann diese Einschränkung allerdings durch das Einfügen eines kleinen Vorteilers am Eingang des RAM-basierenden Zählers umgehen und damit die Zählerfrequenz auf 32 MHz und mehr erhöhen. Bemerkenswert an dieser RAM-Methode ist die enorme Einsparung an Logik, verglichen zu der eines konventionellen Aufbaus. In einem Baustein der XC4000-Serie werden für diese Applikation nur fünf Logikblöcke belegt: einer für das 32 x 1-RAM und das Ausgangs-Flipflop, einer für Addierer, Multiplexer und Carry-Flipflop und drei weitere für den 5-Bit-Adresszähler und die Nullerkennung. Das bedeutet, dass in jedem Logikblock 73 Zwei-Input-NAND-Gatteräquivalente realisiert werden!

Die RAM-Lösung kommt in diesem Falle auch der hochgranularen Gate-Array-Architektur entgegen. Für einen einfachen Binärzähler benötigt man 368 Gatter. In dem Design nach Bild 3 werden nur 333 Gatter benötigt, d. h. 10% weniger. So ungewöhnlich diese RAM-Methode auch erscheint, es finden sich viele Anwendungen dafür, z. B. in der Roboter- und Steuerungstechnik. Die Position eines Roboterarms wird normalerweise mit drei Drehgebern bestimmt, die aus Auf/Ab-Impulsgeneratoren und Zählern bestehen. Für eine Geschwindigkeit von max. 5 m/s und einer Auflösung von 1 µm benötigten diese Zähler eine Auflösung von 0,2 µs-Pulsen und eine Kapazität von mind. 2 Mio. Schritten. Die Zähler müssen ein einfaches Interface zum Systemmikroprozessor haben, so dass der Zählerwert jederzeit und eindeutig ausgelesen werden kann. Die bisher gebräuchlichen Zähler haben einige Einschränkungen. Sie sind relativ klein, haben keine Auf/Ab-Steuerung und sind in einer asynchronen Umgebung sehr schwer auszulesen. Das Problem bei einem herkömmlichen Auf/Ab-Zähler besteht darin, dass er zwischen zwei Werten springen kann, wenn alle (oder die meisten) Zählerbit bei der ankommenden Taktrate wechseln. Ein zuverlässiges Interface zum Mikroprozessor ist damit beinahe unmöglich zu realisieren. Die gesamte Architektur besteht aus drei Zählern, die jeweils in zwei Bereiche aufgeteilt sind **(Bild 4).** Die unteren 4 Bit des Zählers sind ein 4-Bit-Auf/Ab-Grey-Zähler. Dieser läuft asynchron zum Rest des Systems. Zählt man die untern Bit im Grey-Code, so ändert sich nur ein Bit bei jedem Taktübergang. Deshalb kann zu jedem Zeitpunkt sicher und zuverlässig ausgelesen werden. Die oberen 32 Bit des Zählers werden so implementiert, wie in Bild 3 gezeigt. Die oberen Bit sind mit dem Systemprozessor synchronisiert. Bei einer Taktrate von 32 MHz läuft dieser Zählerteil einmal pro Mikrosekunde durch und kann dabei in seinem Wert erhöht, vermindert oder gelesen werden bzw. kann auch ein Preset erfolgen. Die beiden Teile des Zählers kommunizieren über ein Mailbox-Register, das am Carry-out des Grey-Zählers angeschlossen ist. Immer wenn der Grey-Zähler den Wert plus oder minus 8 erreicht, setzt er das Carry/Borrow-Flipflop. Wenn der Mikroprozessor den Zähler lesen möchte, legt er zuerst fest, welcher der drei Zähler zum Ausgangsschieberegister geschickt werden soll. Dieses 32-Bit-Ausgangsschieberegister ist in herkömmlicher Schaltungstechnik implementiert. In der vorliegenden Schaltung hat der Mikroprozessor eine zusätzliche Aufgabe zu bewältigen: die untern Bit des Zählers müssen eventuell vom Grey-Code in eine binäre Form umgewandelt werden. Dieses Design benötigt 39 Logikblöcke eines Bausteins aus der XC4000-Serie. Der kleinste Baustein dieser Produktfamilie hat 64 Logikblöcke. Mit einer konventionellen Lösung würde dieses Design über 78 Logikblöcke belegen und damit doppelt so viele als mit der hier gezeigten RAM-Lösung.

Metronic