Schnellere Bildverarbeitung durch FPGAs

Programmierbare Logik beschleunigt die Bildverarbeitung

FPGAs werden gerne eingesetzt, wenn eine hohe Geschwindigkeit bei der Verarbeitung von Daten gefordert wird – auch in der Bildverarbeitung. Die Programmierung eines FPGA ist jedoch nicht so komfortabel wie bei PC-basierten BV-Systemen, und die Hersteller von FPGA-basierten BV-Komponenten unterstützen es selten. Dabei ermöglichen FPGAs äußerst leistungsfähige Komponenten, wie dieses Beispiel einer intelligenten Kamera zeigt.

Von Matthias Schaffland

Die industrielle Bildverarbeitung ist schon seit geraumer Zeit ein Dauerbrenner in der Automatisierungstechnik. Die Komponenten sollen immer kleiner und leistungsfähiger werden, denn die Anwendungen verlangen immer kürzere Auswerte- und Reaktionszeiten. Wenn die geforderte Geschwindigkeit nicht mit rein softwaregestützten Bildverarbeitungssystemen erreicht werden kann, bietet sich eine immer wichtiger werdende Technologie an: die programmierbaren Hardware-Bausteine, kurz FPGAs.

FPGAs (Field Programmable Gate-Arrays) sind Hardware-Bausteine, die ihre Anfänge in programmierbaren Logikbausteine für einzelne Gatter (GAL, Generic Array Logic) hatten. FPGAs haben in den letzten Jahren eine rasante Entwicklung erlebt und vereinen in einem Chip Ressourcen, die einigen 10 000 bis zu mehreren Millionen Logikgattern entsprechen. Ein FPGA enthält oftmals zusätzlich bereits fertige Komponenten, die für eine Anwendung sehr wichtig sein können. Dazu gehören Speicher, Multiplizierer oder sogar vollständige Prozessoren.

Die Programmierung von FPGAs erfolgt normalerweise mit Software-

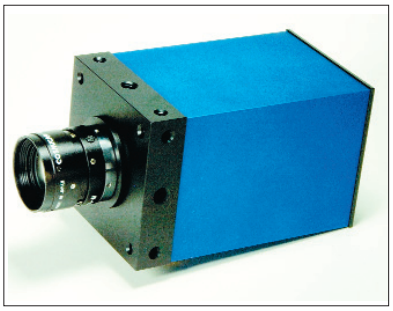

Bild 1. Nur 105 mm lang, 55 mm breit und 45 mm hoch ist die intelligente Kamera CANCam-BF von Feith, ausgestattet mit einem FPGA und einem Mikrocontroller, die beide vom Anwender frei programmiert werden können.

Werkzeugen des jeweiligen Herstellers. Diese übersetzen die in einer Hardware-Beschreibungssprache (z.B. VHDL, Very High Speed Integrated Circuit Hardware Description Language) ausgedrückte Funktion in eine chipspezifische Form und generieren eine Konfigurationsdatei, mit der das FPGA programmiert wird.

FPGAs spielen heute in der Schaltungsentwicklung eine immer wichtiger werdende Rolle. Auch in der industriellen Bildverarbeitung haben sie ihren Platz, als Ergänzung zu Signalprozessoren.

Jede Bildverarbeitungskomponente kann mit FPGAs ausgestattet sein

FPGA-basierte Komponenten lassen sich in unterschiedlichen Bildverarbeitungssystemen einsetzen, die sich in drei Szenarien aufteilen lassen:

▸ *Klassisches BV-System aus Kamera, Framegrabber und PC*

Hier können FPGAs in allen Komponenten vorkommen, in der Kamera, auf dem Framegrabber oder als seperates FPGA-Modul im PC. Allerdings ist zu beachten, ob das FPGA auch vom Anwender genutzt werden kann. So arbeiten zum Beispiel viele Kameras zwar mit FPGAs, diese sind aber nicht vom Anwender für dessen eigene Bildverarbeitungsaufgabe nutzbar. Bei Framegrabbern sind FPGAs ebenfalls recht häufig anzutreffen. Auch hier ist zu unterscheiden, ob und wieweit der Hersteller den Anwender das FPGA individuell nutzen lässt. Für spezielle Anwendungen bieten einige Hersteller auch dedizierte FPGA-PC-Karten an, die dann auch für BV-Aufgaben genutzt werden könnten.

▸ *Kompakte Bildverarbeitungssysteme*

Die Möglichkeiten, hier FPGAs zu nutzen, hängen von der jeweiligen Architektur ab. Zum einen sind diese Systeme PC-basiert, dann besteht z.B. die Möglichkeit, Framegrabber mit FPGAs einzusetzen. Bei anderen Systemen, wie z.B. dem Single-Board-Computer „Cleopatra“ der Firma Feith, sind Framegrabber und FPGA mit integriertem Prozessor auf einer Platine kombiniert. Gerade für platzsparende, aber dennoch leistungsfähige Systeme bietet sich eine solche Kombination an.

▸ *Intelligente Kameras*

Sie stellen eine Weiterentwicklung der kompakten Bildverbeitungssysteme dar: BV-System plus Bildsensor in einem Gehäuse. Intelligente Kameras können auch FPGA-Ressourcen zur

Verfügung stellen, um die Leistungsfähigkeit dieser meist eher begrenzten Systeme zu erweitern (**Bild 1**).

Typische Aufgabenstellungen, die gut in einem FPGA realisiert werden können, sind beispielsweise:

- Realisierung von Look-up-Tabellen z.B. zur Durchführung von Gamma-Korrekturen oder zur Datenreduktion von 10 bit auf 8 bit,
- Datenreduktion auf einen vorgegebenen Bildausschnitt (Windowing, AOI: Area Of Interest),
- Farbinterpolation bei Farbsensoren mit Bayer-Filter,
- Binarisierung, d.h. von einem Schwellwert abhängiges Umwandeln des Bildes zu einem Schwarzweiß-Bild,
- Zusammenfassung benachbarter Pixel zur Helligkeitsteigerung (Binning),
- Histogramm-Erstellung,
- Filter,
- Kanten-Erkennung,
- Bildkompression (z.B. JPEG),
- Nachführung des Bildausschnittes (AOI-Tracking) bei CMOS-Sensoren mit Ansteuerung des Sensors.

Programmierung von Bildverarbeitungsfunktionen im FPGA

Da der FPGA-Baustein sehr mit der Gesamtelektronik einer Bildverarbeitungskomponente verwoben ist – z.B externer Speicher, Prozessorinterface –, ist die Implementierung auch von der gewählten Plattform abhängig. Das heißt, es gibt keine plattformübergreifende Bibliothek, die den Anwender bei der Implementierung von Standardaufgaben unterstützt. Dem Autor ist zur Zeit nur ein Hersteller bekannt, der die Programmierung seiner FPGA-PC-Karten mittels spezieller Software und vordefinierter Funktionsblöcke erleichtert.

Normalerweise erfolgt die Implementierung mit den einschlägigen FPGA-Entwicklungsprogrammen. Dies ist zwar aufwendiger, dafür bleibt aber die volle Flexibilität des FPGA erhalten. Bei dieser „Low-Level"-Implementierung hat sich folgende Vorgehensweise bewährt:

- Entwicklung des Algorithmus auf PC, z.B. unter C, wobei gewisse Richtlinien beachtet werden sollten: Das Abstraktionsniveau von VHDL entspricht etwa dem von Assembler; keine objektorientierten Programmiersprachen; CPUs arbeiten sequenziell, FPGAs parallel.
- Definition der Verarbeitungsreihenfolge beziehungsweise Identifikation der abhängigen Prozesse.
- Test unter entschärften Bedingungen.
- Portierung nach VHDL, evtl. unter Zuhilfenahme entsprechender Software-Tools.
- Simulation und Test unter Laufzeitbedingungen.

```
130  -----------------------
131  --  data processing
132  -----------------------
133  calc: process (vclk)
134  begin
135    if (vclk'event and vclk = '1') then
136      if sys_res= '1' then
137        hw_data <= x"00";
138        hw_wr <= '0';
139      else
140        hw_data <= not(vdata); --invert image
141        hw_wr <= href;
142      end if;
143    end if;
144  end process calc;
```

Dieses Code-Beispiel aus dem Modul „hw_calc“ invertiert die Bilddaten.

Schneller durch Hardware

Entscheidender Vorteil des Einsatzes von FPGAs zur Bildverarbeitung ist die Schnelligkeit. Die Verarbeitung erfolgt direkt in Hardware und wird nicht durch komplexe Software-Architekturen und Betriebssystemfunktionen beeinflusst, sondern nur durch wenige, exakt bestimmbare Parameter. Die wichtigsten sind:

- FPGA-Takt – Ein mit 100 MHz getaktetes FPGA arbeitet normalerweise schneller als eines mit 50 MHz Takt.
- Algorithmus – Wie bei Software steigt mit zunehmender Komplexität auch die Verarbeitungsdauer, und der maximal erreichbare Auswertetakt sinkt. Jedoch lässt sich die Verarbeitungsdauer im FPGA exakt berechnen und messen.
- Speicherzugriffe – Sie kosten immer Zeit, da abhängig von der Speicherbauart gewisse Zugriffssequenzen und Wartezeiten eingehalten werden müssen. Hier lässt sich jedoch eine Worst-Case-Berechnung durchführen, so dass dieser Einfluss auch bestimmbar ist.

Dies sind optimale Voraussetzungen, um ein echtzeitfähiges Bildverarbeitungssystem zu bauen. Was aber bedeutet echtzeitfähig? Echtzeit-Fähigkeit bedeutet eine definierte, reproduzierbare Reaktionszeit. Das heißt nicht automatisch super-schnell, die Reaktionszeit kann durchaus lang sein. Aber: Man kann sich auf die Zeit verlassen. Somit lässt sich beim FPGA eine definierte maximale Reaktionszeit bestimmen, die normalerweise auch noch deutlich kürzer ist als bei software-basierten BV-Systemen.

In FPGAs lassen sich zusätzlich parallele Verarbeitungsstrukturen aufbauen, die im Gegensatz zu software-basierten Funktionen auch wirklich parallel ausgeführt werden. Dies erfordert natürlich vom Entwickler ein genaues Verständnis der Aufgabenstellung und der Technik. Dann lässt sich aber dadurch die Geschwindigkeit noch weiter optimieren.

Maximalgeschwindigkeit durch Online-Verarbeitung

Den größten Geschwindigkeitsvorteil gegenüber einem software-basierten BV-System kann man erzielen, wenn eine Online-Verarbeitung möglich ist. Dies bedeutet, dass die Bilddaten – einzelne Pixel oder begrenzte Anzahl von Pixeln – sofort bei ihrer Verfügbarkeit verarbeitet werden können. Wenn dann das komplette Bild durch das FPGA „gewandert“ ist, steht auch

schon das Ergebnis zur Verfügung. Ein software-basiertes BV-System würde dagegen erst jetzt mit der Verarbeitung anfangen.

Ein Beispiel einer solchen Online-Verarbeitung zeigt das Code-Fragment (**Listing**), bei dem jedes Pixel sofort und ohne Wartezeit manipuliert wird. Selbst wenn für eine Verarbeitungsaufgabe z.B. mehrere Bildzeilen nötig sind, kann schon vor Bildende mit der Verarbeitung begonnen werden. Es lassen sich beispielsweise Pausen im Pixelstrom nutzen – z.B. beim Zeilenwechsel –, so dass das Ergebnis trotzdem praktisch gleichzeitig mit dem Bildende bereitsteht.

Dies verdeutlicht der folgende Zeitvergleich einer realen Applikation. Die Aufgabe ist relativ einfach: Es soll der maximale Pixelwert jeder Spalte – also quer zur Ausleserichtung des Bildsensors – und dessen Position gefunden werden. Zusätzlich soll die Anzahl von Pixeln in einer Spalte ermittelt werden, die einen gegebenen Schwellwert überschreiten, ebenfalls quer zur Ausleserichtung.

Diese Funktion wurde ursprünglich in der CPU einer intelligenten Kamera durchgeführt, was zu einer Zykluszeit von 155 ms führte (Bildaufnahme und Sensorauslesen: 40 ms; Bild-

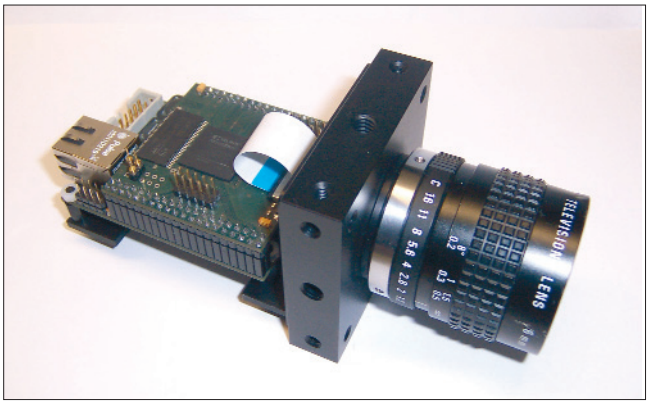

Bild 2. Ein Blick ins Innere der CANCam-BF zeigt die FPGA-Platine, die über der µC-Platine mit den Schnittstellen angeordnet wurde. Der Bildsensor ist direkt mit dem FPGA verbunden und wird von diesem angesteuert.

übertragung vom FPGA zur CPU: 15 ms; Bildauswertung auf CPU (Coldfire, 66 MHz): 100 ms). Da die Applikation eine Gesamtzykluszeit von 60 ms erforderte, war dieser Ansatz deutlich zu langsam. Der CPU stünden statt der erforderlichen 100 ms nur noch max. 5 ms für die Auswertung zur Verfügung.

Nach der Verlagerung der beiden genannten Bildverarbeitungsfunktionen ins FPGA ergab sich eine Zykluszeit von nur noch 42 ms (Bildaufnahme und Sensorauslesen: 40 ms; Bildübertragung vom FPGA zur CPU: 0 ms, da keine Bildübertragung mehr stattfindet; Bildauswertung im FPGA: 0 ms, da diese bereits während der 40 ms der Bildaufnahme bzw. des Auslesens des Sensors erfolgt; Übertragung der Ergebnisdaten zur CPU (Coldfire, 66 MHz): 2 ms). Dies bedeutet eine Beschleunigung um ca. 270 % und eine Unterschreitung des Limits um 18 ms.

Aufwendigere Programmierung, aber schneller

Nachteilig wirkt sich aus, dass der Entwicklungsaufwand höher ist als bei Bildverarbeitungssystemen, deren Operatoren rein in Software gehalten werden. Als groben Anhaltspunkt kann man von einem Faktor 10 ausgehen. Daher lohnt sich der Aufwand

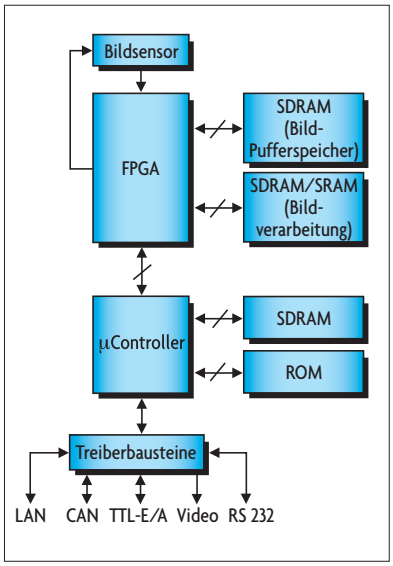

Bild 3. In der CANCam-BF steuert das FPGA den Bildsensor und übernimmt auch die Bilddaten. Die Bildverarbeitungsfunktion kann der Anwender frei zwischen FPGA und Mikrocontroller aufteilen. Der Mikrocontroller übernimmt zusätzlich Steuer- und Kommunikationsaufgaben.

meist nicht für Einzelapplikationen mit nur wenigen Bildverarbeitungskomponenten. Bei großen Anwendungen, Produkten bzw. Maschinen mit Bildverarbeitung, kann der Nachteil aber schnell durch die Vorteile ausgeglichen werden. Zu den Vorteilen gehört – neben der erwähnten hohen Geschwindigkeit – die Möglichkeit, mit intelligenten und abgestimmten Komponenten die Kosten für Hardware in der Anschaffung und im Betrieb zu senken. Zusätzlich bleibt die Flexibilität erhalten, da sich die Programmierdateien der FPGAs leicht austauschen lassen und damit eine neue oder andere Funktion ins System gebracht werden kann.

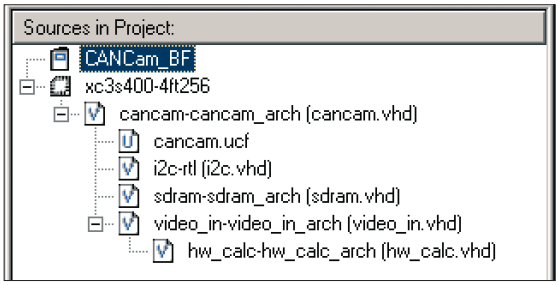

Bild 4. Die FPGA-Programmierumgebung listet alle VHDL-Module auf. Die oberen Module, z.B. „sdram“ und „video_in“, enthalten die Logik zur Infrastruktur und sind dem Anwender nicht zugänglich. Er schreibt seine Bildverarbeitungsprogramme in „hw-calc“.

Glücklicherweise kann man das Thema Logikbedarf etwas entspannter angehen als noch vor wenigen Jahren, als FPGA-Ressourcen richtig teuer waren. Sparsam umgehen sollte man aber weiterhin mit begrenzten Elementen wie z.B. „BlockRAM“, da dieser interne Speicher nach wie vor relativ knapp ist. Relativ, weil die „BlockRAM“-Größe auch deutlich zugenommen hat, aber mit steigender Leistungsfähigkeit der FPGAs auch die Einsatzfelder wachsen.

Ein Teil der FPGA-Resourcen benötigt eine intelligente Kamera für sich selbst. Dennoch stehen dem Anwender genügend Logikelemente zur Verfügung, wie das Beispiel der Kamera CANCam-BF der Firma Feith Sensor to Image mit einem Xilinx-FPGA Spartan3-400 zeigt (**Bild 2**). Die Basislogik des FPGA ohne Bildverarbeitungsmodul „HW_Calc“ benötigt ca. 30 % der Logikelemente (Slices) und drei von 16 „BlockRAM“-Einheiten. Das bedeutet, dass dem Anwender noch fast drei Viertel der FPGA-Ressourcen zur Verfügung stehen. Wird der pingleiche 1000er Baustein eingesetzt, können sogar nahezu 90 % des FPGA vom Anwender genutzt werden.

Intelligente Kamera mit frei programmierbarem FPGA

Um den praktischen Einsatz und die Programmierbarkeit eines FPGA-basierten Bildverarbeitungssystems zu verdeutlichen, folgt ein Beispiel mit der CANCam-BF, deren Blockschaltung (**Bild 3**) die Architektur um das FPGA verdeutlicht. Die Bilddaten laufen vom Sensor in das FPGA, wo sie im VHDL-Modul „video_in“ (**Bild 4**) formatiert werden, d.h., es wird z.B. eine Umsetzung von 10 bit zu 8 bit vorgenommen, ungültige Daten werden ausgefiltert. Die gültigen Daten können dann durch das Modul „hw_calc“ geleitet werden, wo die eigentliche Bildverarbeitung zu platzieren ist. Unbearbeitete oder in „hw_calc“ modifizierte Bilddaten gelangen anschließend in den Bild-Pufferspeicher (über Modul „SDRAM“), von wo aus sie in das SDRAM des Mikrocontrollers (BlackFin-DSP, 600 MHz) übertragen werden. Falls die Bildverarbeitung komplett im FPGA erfolgt, kann dieser letzte Schritt natürlich auch entfallen, und die Ergebnisse können in Registern zur Verfügung gestellt werden. Der zweite Speicher ist für Bildverarbeitungsaufgaben reserviert und kann über das Modul „hw_calc“ angesprochen werden.

Das VHDL-Code-Fragment im Listing zeigt einen Ausschnitt aus dem „hw_calc“-Modul. In diesem Beispiel werden die Originaldaten nur invertiert und wieder an das übergeordnete Modul zurückgeleitet. Das Modul „hw_calc“ steht dem Anwender als Quell-Code zur Verfügung, wohingegen die anderen Module bereits vorübersetzt sind. Als Entwicklungswerkzeug wird Integrated Software Environment (ISE) von Xilinx mit Simulator ModelSim verwendet. Beide Software-Werkzeuge sind auch in einer zwar limitierten, aber kostenfreien Version verfügbar [2], die zumindest für den Einstieg in die FPGA-Programmierung völlig ausreichend ist.

Auch wenn der Einstieg in die Welt der programmierbaren Logik etwas Aufwand bedeutet, eröffnen FPGAs im Bereich der Bildverarbeitung neue Möglichkeiten, um die Leistungsfähigkeit bestehender Systeme zu erweitern. *hs*

Internet

[1] www.feith.de
[2] www.xilinx.com/xlnx/xil_sw_updates_home.jsp

Dipl.-Ing. (FH) Matthias Schaffland

wurde in Coburg geboren und studierte bis 1997 an der FH Nürnberg Elektrotechnik. Bis 2002 war er als Berater und Projektleiter bei einem IT-Dienstleister beschäftigt. Seit 2003 arbeitet er im Vertrieb und in der Entwicklung von Bildverarbeitungskomponenten der Firma Feith Sensor to Image GmbH, Schongau. Schwerpunkt liegt hier bei intelligenten Kamerasystemen mit offener FPGA-Schnittstelle.
mas@feith.de