AW1 - Vortrag

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Automatisierte Hardware-Software Partitionierung

Andre Jestel

Betreuer: Prof. Dr.-Ing. Bernd Schwarz

Inhaltsverzeichnis

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Motivation

- Problem-/ Fragestellung für System-Designer → Welche Funktionalitäten in Hard- und Software ?

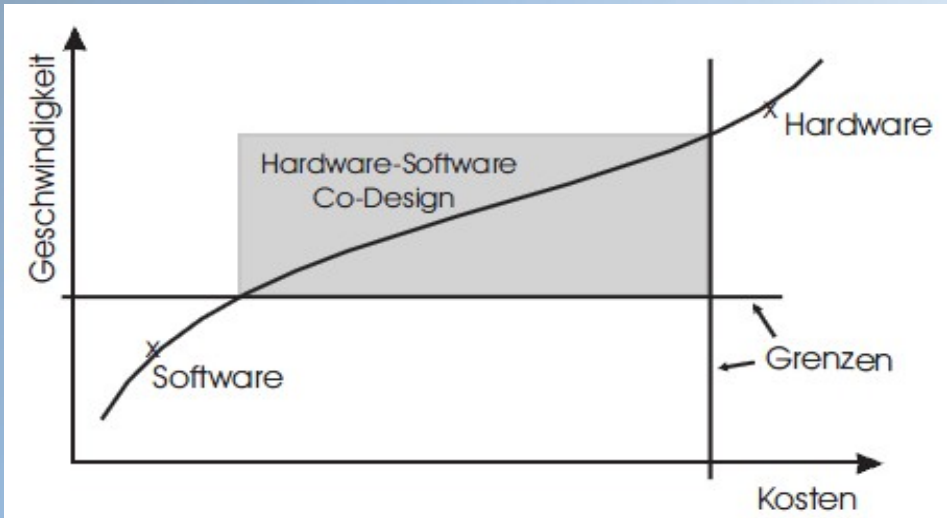

Bild 1: Kosten-Geschwindigkeisgraph mit Anforderungsgrenzen für die Implementierung in HW und SW

[2]

- Herausforderung dabei → Kosten minimieren und Verarbeitungsgeschwindigkeit maximieren

Lösungsansatz

- Bisherige Praxis: Entscheidung durch Erfahrungswerte der Entwickler → Datenabhängige Verarbeitung in SW und datenunabhängige Verarbeitung in HW
- Besser → Nutzung von objektiven Entscheidungskriterien basierend auf Metriken
- Wünschenswert → Plattformunabhängige und automatisierte Partitionierungsverfahren

Zielsetzung

- *„Symmetrische Multiprozessor-systeme“* → [1] Parallelisierung von SW-Threads und Hardware maximieren
- FAUST-Projekte → Anwendungen zur Technologie-erprobung nutzen

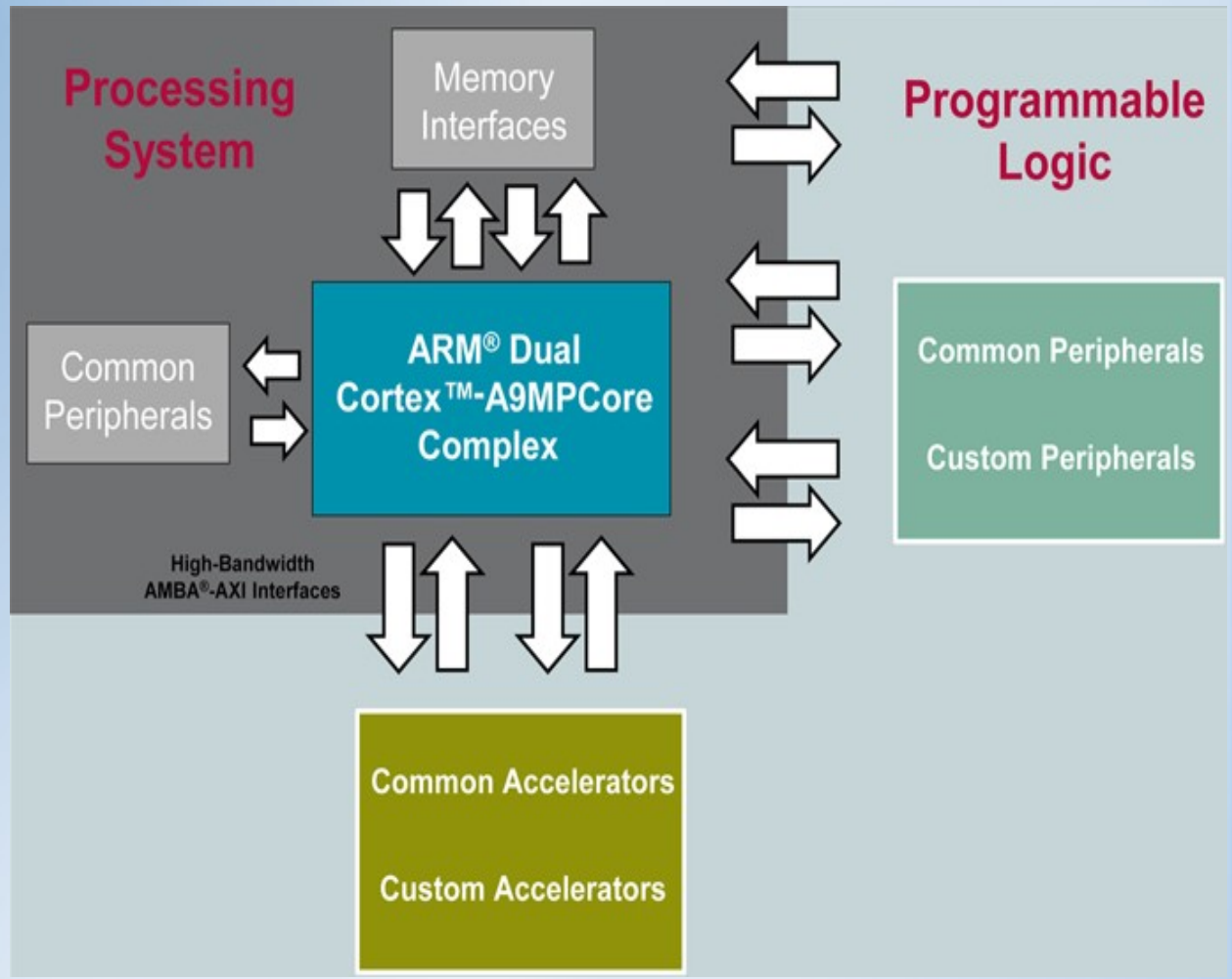

Bild 2: Xilinx Extensible Processing Platform [7]

Inhaltsverzeichnis

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Vergleichsbeispiel Hard-/Software

- Klassisches Teilproblem → Lösen von Differentialgleichungen (DGLs)
- Auswahl einer Beispiel-DGL → $y'(x) = y(x)$ (e-Funktion)
- Wahl eines überschaubaren, numerischen Lösungsverfahren → Explizites Eulerverfahren [4]
- Rekursive Approximationsvorschrift → $y(n + 1) = y(n) + \Delta x * y(n) = y(n) * (1 + \Delta x)$

SW-Realisierung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

```
int a_in, x0_in, dx_in, y0_in;
volatile int y_out;

int main( void )
{
  int a = a_in;
  int x = x0_in;
  int dx = dx_in;
  y_out = y0_in;

  while (x <= a) {
    x = x + dx;
    y_out = y_out * ( 1 + dx );
  }
  return 0;
}
```

```
main:
    ldr r3, a_in
    ldr r0, [r3, #0]
    ldr r3, x0_in
    ldr r2, [r3, #0]
    ldr r3, dx_in
    ldr r1, [r3, #0]
    ldr r3, y0_in
    ldr r3, [r3, #0]
    b   .while
.loop:
    add ip, r1, #1
    mul r3, ip, r3
    ldr ip, y_out
    add r2, r2, r1
    str r3, [ip, #0]
.while:
    cmp r2, r0
    ble .loop
    bx  lr
```

Bild 3: SW-Implementierung des Lösungsverfahrens zur linearen DGL in C und Assembler

- Zielplattform: RISC-µP → ARM7
- Kostenaufwand durch Quellcodekomplexität: ca. 15 Zeilen
- Leistung durch Laufzeit mit 100 Iterationen → Anzahl der Instruktionen / Taktfrequenz = (9 + 7*100) / (100 MHz) = 7,09 µs

HW-Realisierung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

- Zielplattform: FPGA → Spartan-3E
- Quellcode-komplexität: ca. 45 Zeilen
- Taktzyklen / Taktfrequenz = (4 + 100) / (60 MHz) = 1,73 µs

```
LIBRARY ieee;
  USE ieee.std_logic_1164.ALL;
  USE ieee.std_logic_signed.ALL;

ENTITY dgl_solver IS
    PORT ( bus_inout : INOUT
           std_logic_vector(31 DOWNTO 0);
           clk,reset : IN std_logic)
END dgl_solver;

ARCHITECTURE behavior OF dgl_solver IS
    SIGNAL a, x, dx, y :
    std_logic_vector(31 DOWNTO 0);
    SIGNAL state :
    std_logic_vector(2 DOWNTO 0);
BEGIN
  PROCESS (clk, reset) BEGIN


  END PROCESS;
  bus inout <= y WHEN state(2) = '1'
                ELSE (OTHERS => 'Z');
END ARCHITECTURE behavior;
```

```
IF (reset = '1') THEN
  a <= (OTHERS => '0');
  x <= (OTHERS => '0');
 dx <= (OTHERS => '0');
  y <= (OTHERS => '0');
  state <= (OTHERS => '0');
ELSIF(clk'EVENT AND clk = '1')THEN
  CASE state IS
    WHEN "000" => a <= bus_inout;
                  state <= "001";
    WHEN "001" => x <= bus_inout;
                  state <= "010";
    WHEN "010" => dx <= bus_inout;
                  state <= "011";
    WHEN "011" => y <= bus_inout;
                  state <= "1--";
    WHEN OTHERS =>
    IF ( x < a ) THEN
      x <= x + dx;
      y <= y * ( 1 + dx );
    END IF;
  END CASE;
END IF;
```

Bild 4: HW-Implementierung des Lösungsverfahrens in VHDL

Weitere Metriken

- **Leistungsaufnahme** → Strom einsparen und die Betriebsdauer (vor allem bei Batteriebetrieb) maximieren
- **Ressourcenbedarf** → Minimierung von Gewicht, Fläche, Anzahl der Pins zur Kostenreduzierung (insbesondere bei eingebetteten Systemen)
- **Wartbarkeit** → System soll erweiterbar sein sowie Testbarkeit und Fehlertoleranz garantieren

[3] [6]

Inhaltsverzeichnis

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Hierarchisches Clustering

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

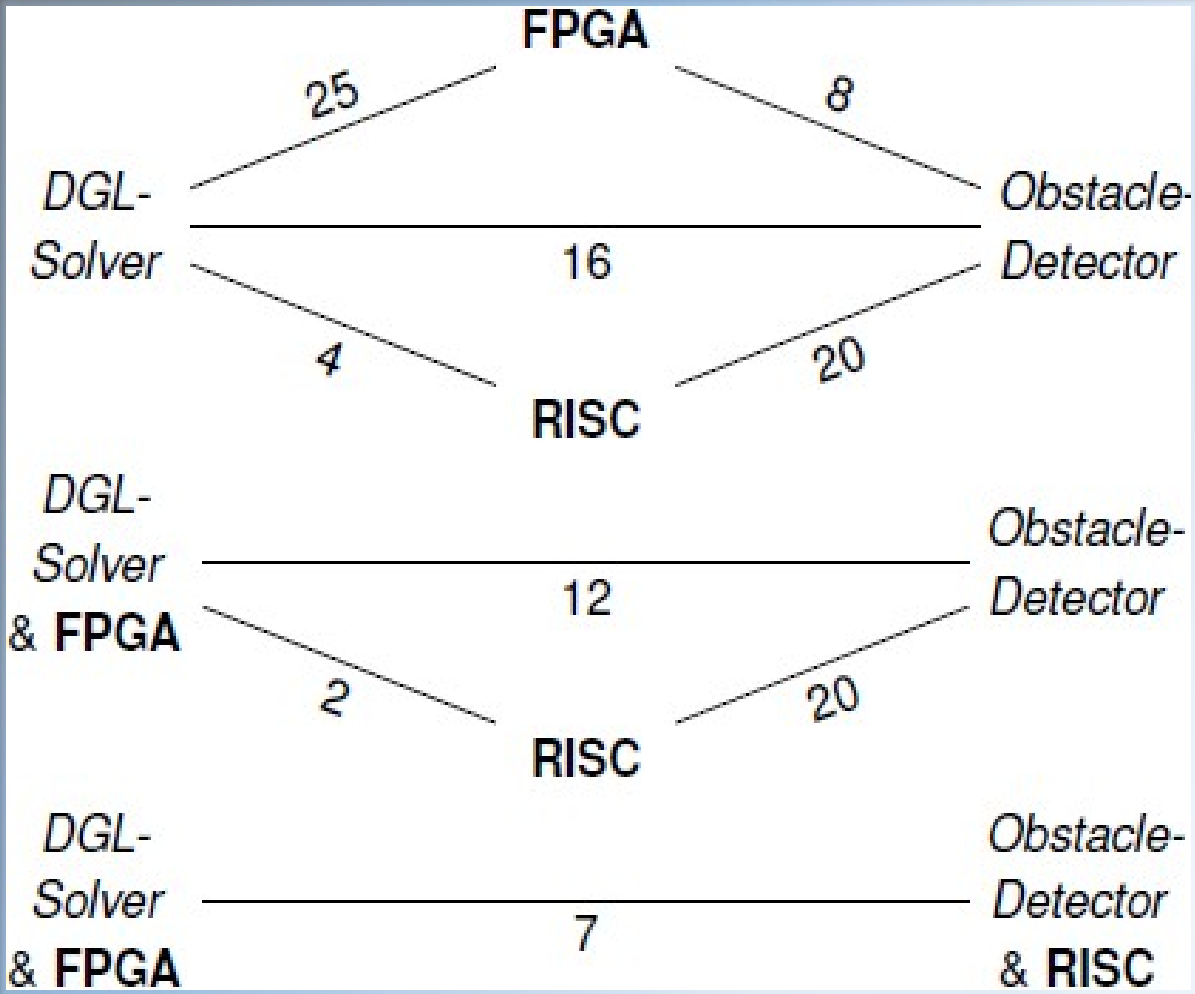

Bild 5: Beispielablauf in 2 Schritten

- *n* Funktionsblöcke und **m** Plattformen bilden einen ungerichteten und gewichteten Graphen
- Kantengewichte sagen aus, wie wünschenswert eine Zusammen-legung ist

Hierarchisches Clustering

- Es existieren bereits Algorithmen mit quadratischer Laufzeit → $O((n+m)^2)$ [5]
- Nachteil: Schwierige Erfassung von geeigneten Anfangskantengewichten
- Erweiterungsmöglichkeit durch Ergänzung von Kommunikationskanälen: Bussysteme oder gemeinsame Speicher → Ausgangsbasis für die Integration

Simulated Annealing

- Inspiriert von den Abkühlungsprozessen der Festkörperphysik: Atome streben nach der energetisch günstigsten Struktur
- Festigkeit eines Stoffs gegeben durch die momentane, strukturelle Ausrichtung der Atome
- Temperatur entspricht einem Maß dafür, wie wahrscheinlich energetisch ungünstige Atombewegungen stattfinden

Simulated Annealing

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

- Aufenthaltsräume der Atome ≈ Plattformen
- Bewegende Atome ≈ Funktionsblöcke
- Energetik ≈ Metriken
- Temperaturverlauf ≈ zeitlicher Verlauf des Verfahrens

```
PROCEDURE SIMULATED-ANNEALING(P)
  temp = Anfangstemperatur;
  cost = c(P);
  WHILE (Frozen == FALSE)
     WHILE (Equilibrum == FALSE)
          P' = RandomMove(P);
          cost' = c(P');
          deltacost = cost' − cost;
          IF ( min(1, e^(−deltacost/temp)) > Random[0,1))
                P = P';
                cost = cost';
          ENDIF
     ENDWHILE
  temp = DecreaseTemp(temp);
  ENDWHILE
  RETURN(P);
END PROCEDURE
```

Bild 6: Pseudocode für SA [6]

Kerninghan-Lin

- Modifikation von Simulated Annealing → Ersetzt Zufallsanteil durch festgelegte Metriken und kein mehrfaches Verschieben einer Komponente
- Alle Komponenten zunächst *versuchsweise* umgruppieren → Tabelle speichert wie hoch die Kosten *wären*
- Iteratives Auswählen der günstigsten Partitionen bis keine günstigere mehr gefunden werden kann
- Zeitkomplexität → $O(m * n^3)$ [2]

Inhaltsverzeichnis

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Zusammenfassung

- Einhalten der Laufzeit- und Kostenanforderungen → Partitionierungsentscheidungen sind unumgänglich
- Bisherige Praxis bei der Entscheidung → Subjektiv nach Erfahrungswerten
- Analytische Lösung nicht effizient berechenbar → Heuristiken nötig
- Ziel: Technologie-Kompetenz im Hinblick auf SMP aufbauen → Metriken identifizieren und (automatisierte) Verfahren erproben

Literatur

[1] BORDASCH, Heiko: Multiprozessor System on Chip. Hochschule für Angewandte Wissenschaften Hamburg: Seminar Anwendungen 1 Wintersemester 2009/10. 2009. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/bordasch/bericht.pdf. – Zugriffsdatum: 2.10.2010

[2] KAZUBIAK, Jens: Automatisierte Hardware-Software Partitionierung am Beispiel eines eingebetteten, echtzeitfähigen Stereobildanalysesystems in Kraftfahrzeugen, Otto-von-Guericke-Universität Magdeburg - Fakultät für Elektrotechnik und Informationstechnik, Dissertation, 2008. – URL http://diglib.uni magdeburg.de/Dissertationen/2008/jenkaszubiak.pdf. – Zugriffsdatum: 28.09.2010

[3] MARWEDEL, Peter: Embedded System Design. Springer, 2005. – ISBN 978-0387292373

[4] MEISEL, Andreas: Differentialgleichungen. Hochschule für Angewandte Wissenschaften Hamburg: Vorlesungsfolien - Modellierung dynamischer Systeme im Wintersemester 2010/11. 2010. – URL http://www.informatik.haw-hamburg.de/fileadmin/Homepages/ProfMeisel/Vorlesungen/MO/V/MOV02.pdf. – Zugriffsdatum: 4.11.2010

[5] MÖLLER, Dietmar P. F.: Vorlesungsfolien: Hardware/Software Co-Design (HSCD). Universität Hamburg - Fachbereich Informatik. 2007. – URL http://www.informatik.uni-hamburg.de/TIS/files/VL_HSCD _Modul_MV4.8_2_D.pdf. – Zugriffsdatum: 14.10.2010

[6] PLATZNER, Marco: Skriptum zur Vorlesung: Hardware/Software Codesign. Universität Paderborn - Fachgebiet Technische Informatik. 2007. – URL http://www.paderborn.de/fileadmin/Informatik/AGPlatzner/Teaching/SS07/Hardware_Software_Codesign/HwSwCd_Skriptum.pdf. – Zugriffsdatum: 9.10.2010

[7] SANTARINI, Mike: Xilinx Architects ARM-Based Processor-First, Processor-Centric Device. In: Xcell Journal (2010), März, S. 6–11