

# Prototyping und Dynamische Rekonfiguration von anwendungsspezifischen Architekturen

Dipl.-Ing. Heiko Hinkelmann,  
Dr.-Ing. Peter Zipf

Prof. Dr. Dr. h.c. mult. Manfred Glesner  
Technische Universität Darmstadt,  
Fachgebiet Mikroelektronische Systeme  
Karlstrasse 15, 64283 Darmstadt

Kontakt:  
{hinkelmann, zipf, glesner}@mes.tu-darmstadt.de  
[http://www.mes.tu-darmstadt.de/research/dfg\\_rr](http://www.mes.tu-darmstadt.de/research/dfg_rr)

- Zwei Schwerpunkte wurden gesetzt
- Entwicklung einer Prototyp-Plattform für unsere rekonfigurierbare Sensorknotenarchitektur
  - Flexible, FPGA-basierte Prototyp-Plattform
  - Basisboard zur Debugging-Unterstützung
- Untersuchung von Verfahren zur Dynamischen Rekonfiguration
  - Verallgemeinerung unseres in der 2. Phase entwickelten Rekonfigurationsverfahrens
  - Entwicklung mathematischer Modelle
  - Ableitung von Bewertungskriterien und –metriken für dynamische Rekonfiguration

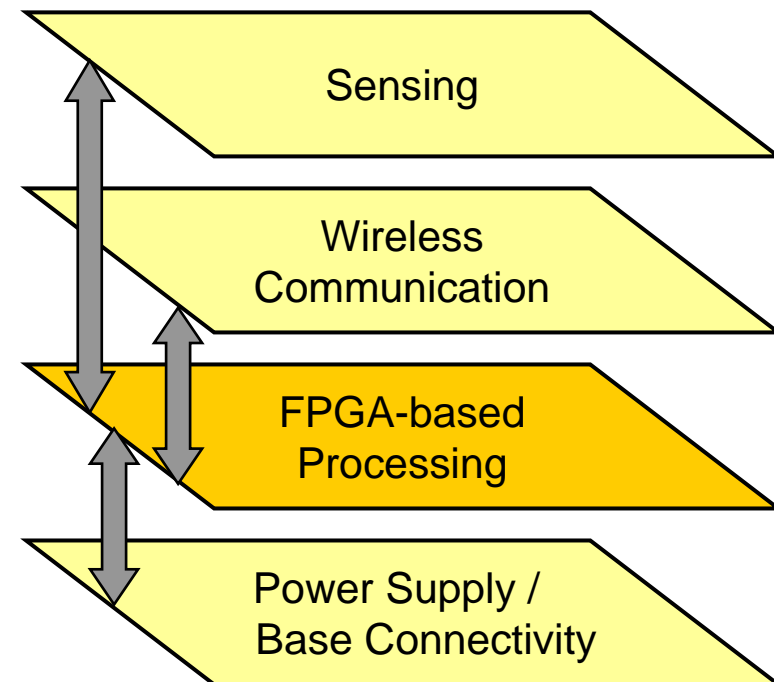
- Rückblick: In der 2. Projektphase wurde eine rekonfigurierbare Architektur für drahtlose Sensorknoten entworfen
  - U.a. konnte eine Steigerung der Energieeffizienz um ein bis zwei Größenordnungen gegenüber rein Prozessor-basierten Architekturen gezeigt werden
- Die Verifikation dieser Architektur erfordert die Entwicklung einer geeigneten Prototyp-Plattform
  - Zur Überwindung der Eingeschränktheit rechnergestützter Simulationen
  - Die Prototypen ermöglichen Tests kompletter Sensornetze unter realen Bedingungen
  - Hauptzweck ist die Überprüfung der Funktionalität unserer Architektur

VHDL-Modell der  
rekonfigurierbaren  
Sensorknotenarchitektur

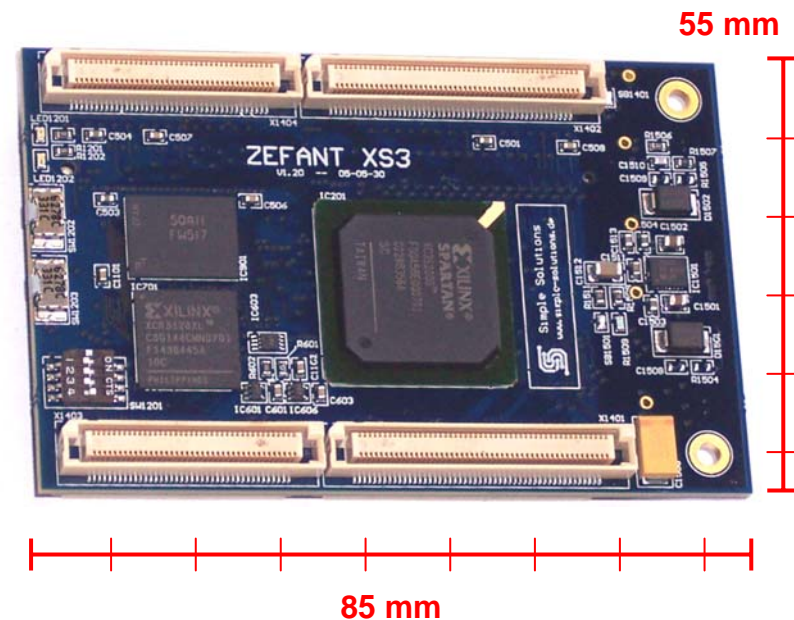


- **Spezielle Herausforderungen:**
  - Die Emulation kompletter Architekturen erfordert ein großes FPGA, gleichzeitig soll die Plattform jedoch auch die Eigenschaften eines kompakten, autonomen eingebetteten Systems besitzen
  - Die Plattform soll flexibel und für verschiedenste Anwendungsbereiche von Sensornetzen, Protokolle und Sensortypen geeignet sein
  - Die Überwachung ganzer Netzwerke erfordert Lösungen zum Fernzugriff auf einzelne Knoten und Information über ihr internes Verhalten
- **Zentraler Lösungsansatz ist die Verwendung eines einzelnen FPGAs mit hoher Kapazität**
  - zur Hardware-Emulation,
  - als generische Schnittstelle zu Sensoren und Funk-Transceivern,
  - sowie zur nahtlosen Einbettung von Debugging-Funktionen
- Dieses Konzept wurde mit der Verwendung eines sekundären Hilfsnetzwerks („**Deployment Support Network**“) gekoppelt, um weitreichende Debugging-Möglichkeiten zu erhalten

- Die Prototyp-Plattform ist schichtweise aufgebaut
- Vorteile des modularen Aufbaus:
  - Kompaktes Design
  - Hauptfunktionen werden getrennt implementiert
    - “Divide & Conquer“
  - Teile der Plattform können leicht gegen Alternativen ausgetauscht werden, ohne andere Schichten zu beeinflussen (z.B. bei Wechsel der Sensoren)
- Die Verbindung der Schichten erfolgt über das FPGA
  - Hohe Flexibilität

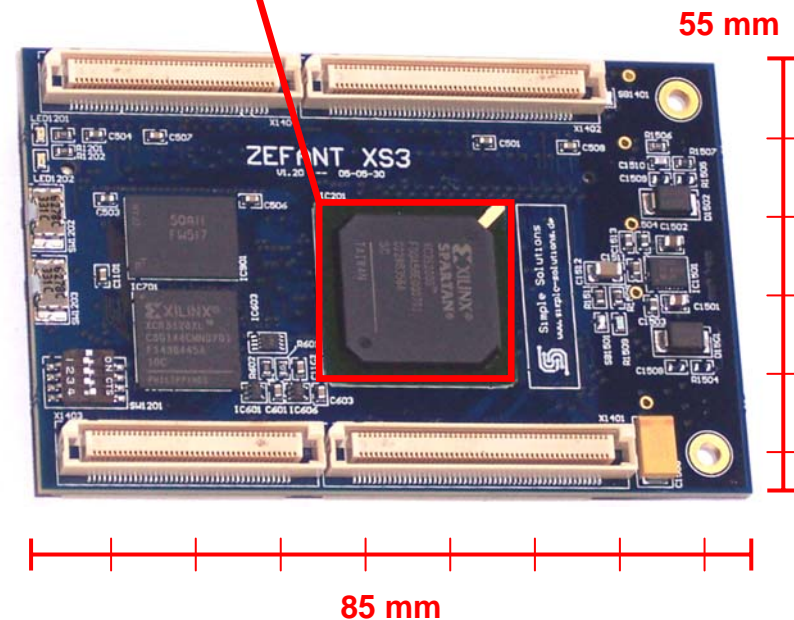


- Technische Details:
  - Zefant XS3-2000 board \*
  - Größe = Kreditkarte (55mm x 85mm)
  - Spartan3-2000 FPGA
    - 2000k system gates
    - (drittgrößtes Spartan3)
  - I/O: mehr als 250 Pins
  - 128 MBit Flash-Speicher
    - für die Konfiguration des FPGA
  - CPLD
  - On-board Spannungswandler
    - von 4 bis 6 V ...
    - ... auf 3.3V, 2.5V, 1.2V
- Alternativen:
  - Andere Zefant boards mit S3-400, S3-1000, or S3-1500 FPGAs



- Technische Details:

- Zefant XS3-2000 board \*
- Größe = Kreditkarte (55mm x 85mm)
- Spartan3-2000 FPGA
  - 2000k system gates
  - (drittgrößtes Spartan3)
- I/O: mehr als 250 Pins
- 128 MBit Flash-Speicher
  - für die Konfiguration des FPGA
- CPLD
- On-board Spannungswandler
  - von 4 bis 6 V ...
  - ... auf 3.3V, 2.5V, 1.2V

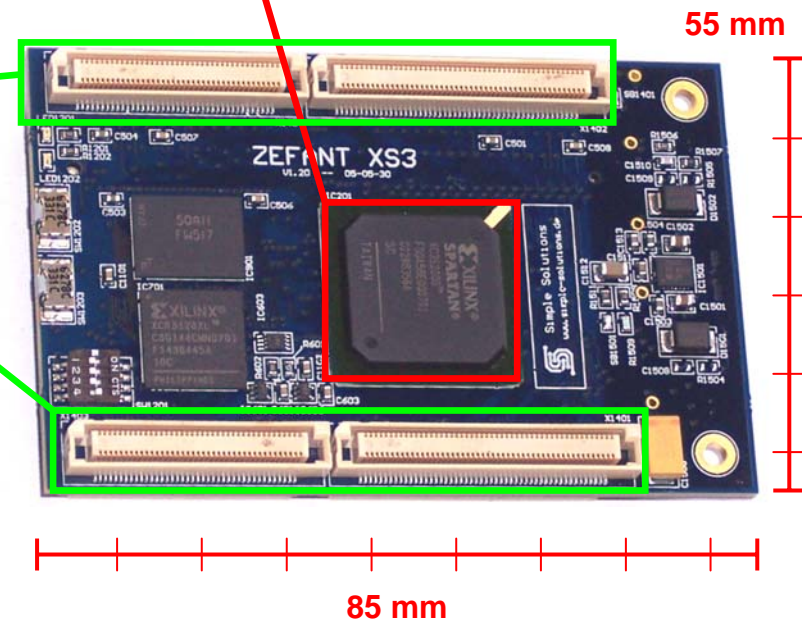


- Alternativen:

- Andere Zefant boards mit S3-400, S3-1000, or S3-1500 FPGAs

- Technische Details:

- Zefant XS3-2000 board \*
- Größe = Kreditkarte (55mm x 85mm)
- Spartan3-2000 FPGA
  - 2000k system gates
  - (drittgrößtes Spartan3)
- I/O: mehr als 250 Pins
- 128 MBit Flash-Speicher
  - für die Konfiguration des FPGA
- CPLD
- On-board Spannungswandler
  - von 4 bis 6 V ...
  - ... auf 3.3V, 2.5V, 1.2V

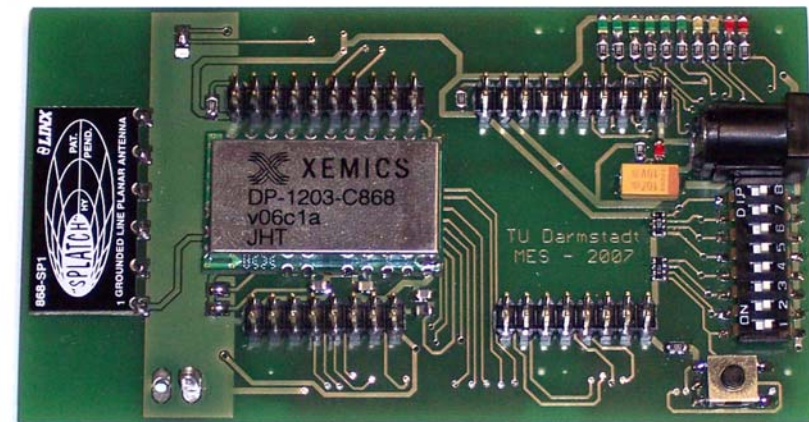


- Alternativen:

- Andere Zefant boards mit S3-400, S3-1000, or S3-1500 FPGAs

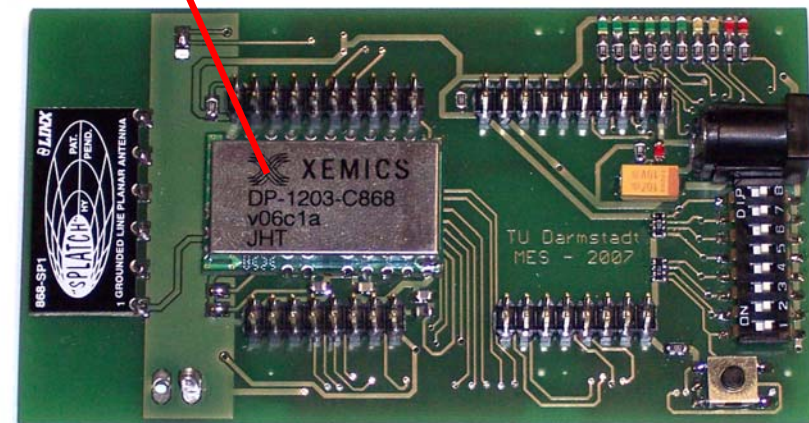
- Anforderungen:
  - Flexible Funkschnittstelle
    - Keine Einschränkung von Kommunikationsstandards
    - Freie Wahl von MAC-Protokollen
    - Möglichkeit, Teile eines Protokolls in Hardware zu implementieren (emuliert durch das FPGA)
      - Unterstützung von MAC-Layer Funktionen
      - Fehlerkorrekturverfahren (forward error correction codes)
      - Hardwarekomponenten zur Verschlüsselung
      - ...
    - Last not least: geringer Leistungsverbrauch
  - Ein einfacher Funk-Transceiver, der lediglich die Bitübertragungsschicht implementiert, wird benötigt!
  - Zusätzliche Komponenten:
    - Antenne
    - Benutzerschnittstelle (LEDs, Tasten, ...)
    - Schnittstelle zur Anbindung von Sensoren

- Implementierungsdetails:
- Eine eigene Platine wurde entwickelt:
  - Xemics DP-1203 Funk-Transceiver
    - 868 MHz ISM-Band
    - Programmierbare Parameter
    - Datenraten: 1.2 to 152.3 kbps
    - RSSI (Received Signal Strength Indication)
  - Planare Antenne
  - DIP-Switch als MAC-Adresse
  - Zwei generische Steckplätze für Sensormodule



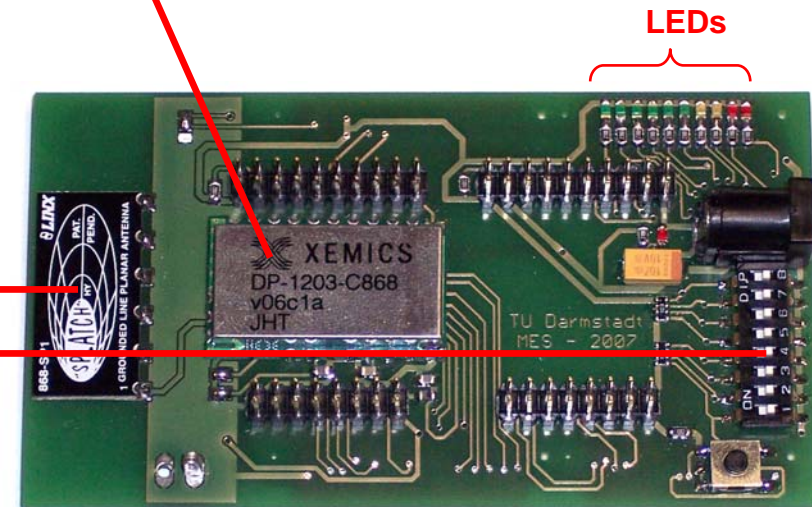
- Alternativen:
  - Spezifische Transceiver/Protokolle können bei Bedarf anstatt der flexiblen Funklösung eingesetzt werden
  - z.B. Zigbee-kompatible Transceiver

- Implementierungsdetails:
- Eine eigene Platine wurde entwickelt:
  - Xemics DP-1203 Funk-Transceiver
    - 868 MHz ISM-Band
    - Programmierbare Parameter
    - Datenraten: 1.2 to 152.3 kbps
    - RSSI (Received Signal Strength Indication)
  - Planare Antenne
  - DIP-Switch als MAC-Adresse
  - Zwei generische Steckplätze für Sensormodule



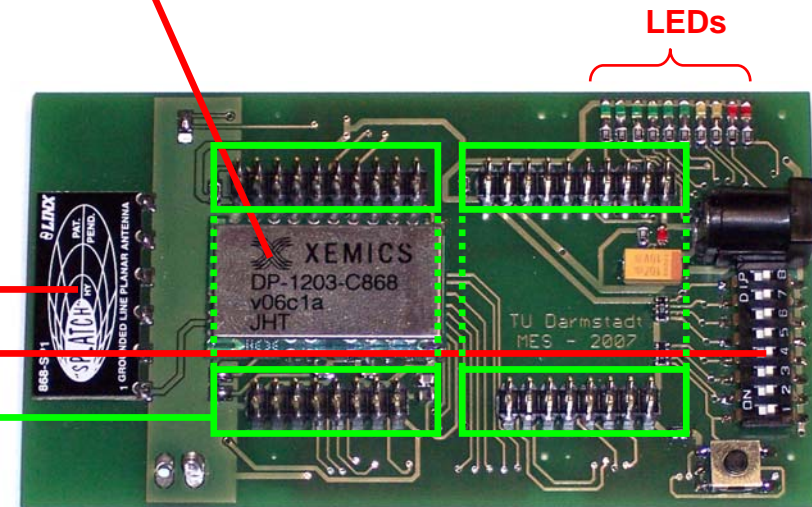
- Alternativen:
  - Spezifische Transceiver/Protokolle können bei Bedarf anstatt der flexiblen Funklösung eingesetzt werden
  - z.B. Zigbee-kompatible Transceiver

- Implementierungsdetails:
- Eine eigene Platine wurde entwickelt:
  - Xemics DP-1203 Funk-Transceiver
    - 868 MHz ISM-Band
    - Programmierbare Parameter
    - Datenraten: 1.2 to 152.3 kbps
    - RSSI (Received Signal Strength Indication)
  - Planare Antenne
  - DIP-Switch als MAC-Adresse
  - Zwei generische Steckplätze für Sensormodule



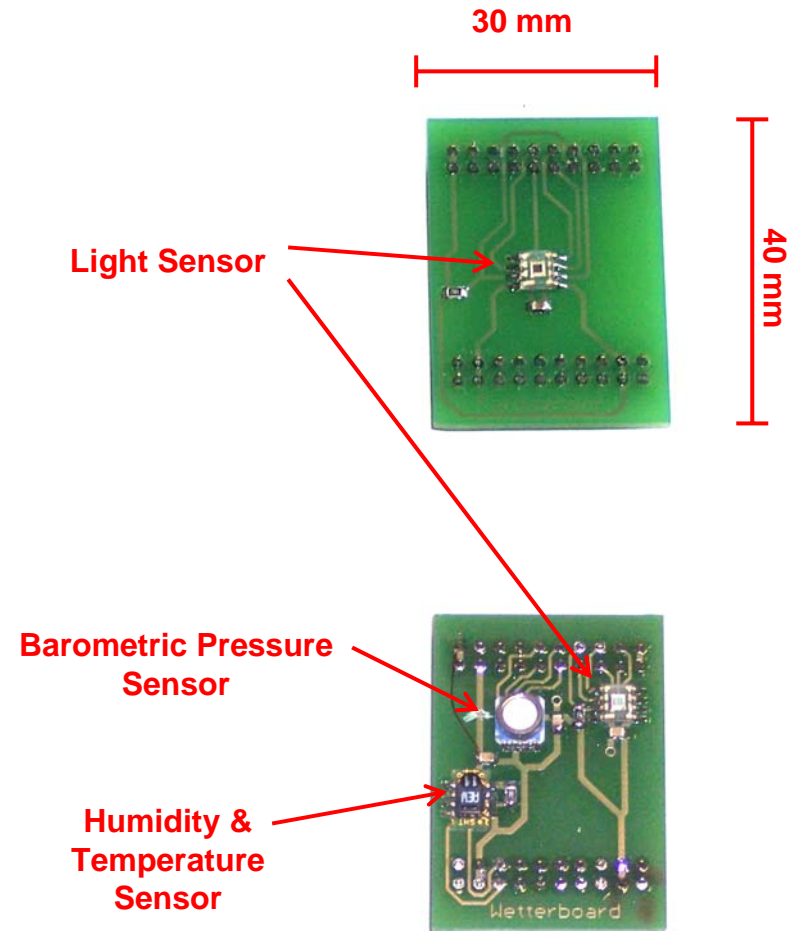
- Alternativen:
  - Spezifische Transceiver/Protokolle können bei Bedarf anstatt der flexiblen Funklösung eingesetzt werden
  - z.B. Zigbee-kompatible Transceiver

- Implementierungsdetails:
- Eine eigene Platine wurde entwickelt:
  - Xemics DP-1203 Funk-Transceiver
    - 868 MHz ISM-Band
    - Programmierbare Parameter
    - Datenraten: 1.2 to 152.3 kbps
    - RSSI (Received Signal Strength Indication)
  - Planare Antenne
  - DIP-Switch als MAC-Adresse
  - Zwei generische Steckplätze für Sensormodule

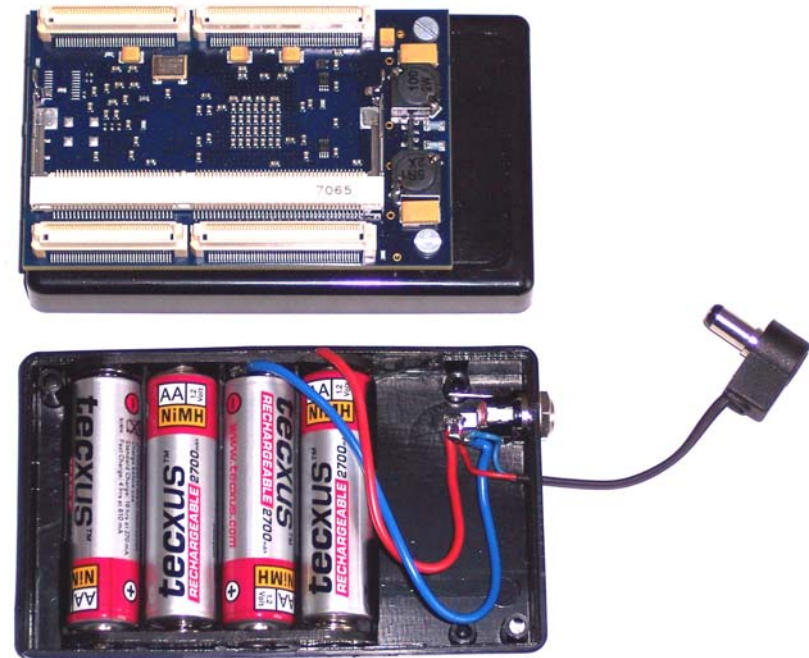


- Alternativen:
  - Spezifische Transceiver/Protokolle können bei Bedarf anstatt der flexiblen Funklösung eingesetzt werden
  - z.B. Zigbee-kompatible Transceiver

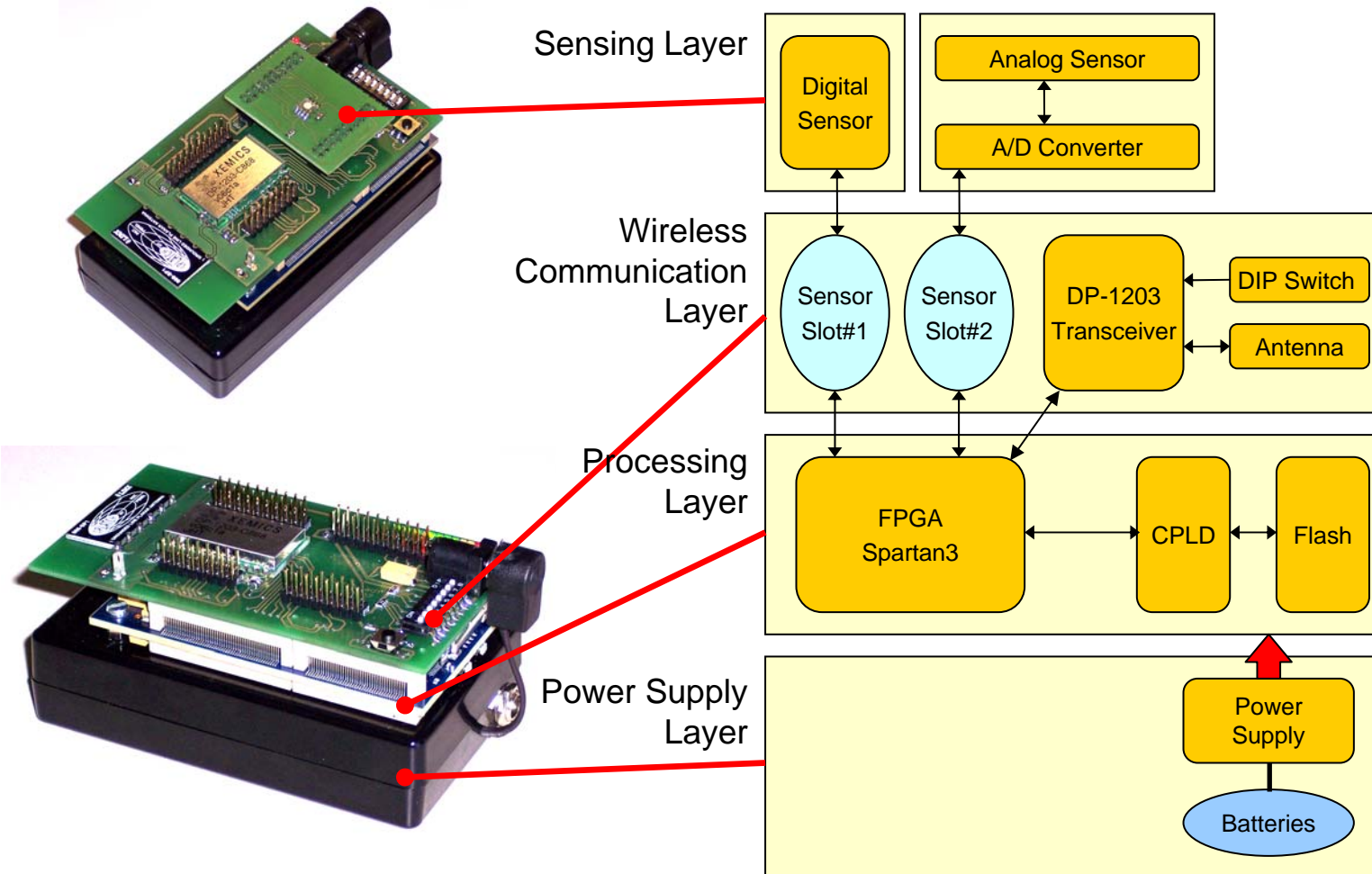
- **Ziele:**
  - Generische Sensor-Kommunikation
  - Einfacher Austausch von Sensoren
- **Technische Details:**
  - Zwei generische Steckplätze
    - je 28 generische FPGA-Pins
    - 3.3V und 5V verfügbar
  - Sensoren werden auf kleinen Modulen montiert
    - Digitale Sensoren
    - Analoge Sensoren + A/D-Wandler
    - Mehrere Sensoren pro Modul möglich
- **Praktische jeder Sensortyp kann dadurch unterstützt werden**
  - Licht, Temperatur, Druck, Beschleunigung, (Ultra-)Schall, ...



- Ziele:
  - Autonome Spannungsversorgung
- Technische Details:
  - Vier AA Akkus
    - Jeweils 2700mAh
    - Verbunden in Serie
  - Montiert in einem Batteriegehäuse unter dem FPGA-Board
  - Batterielevensdauer unter normalen Bedingungen: ca. 10 Std.
  - Schnelle Wiederaufladung innerhalb der Box möglich



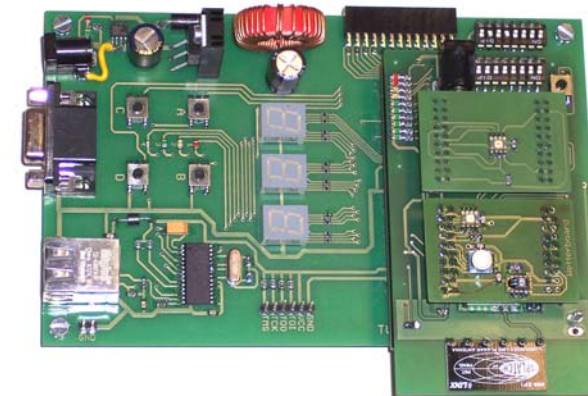
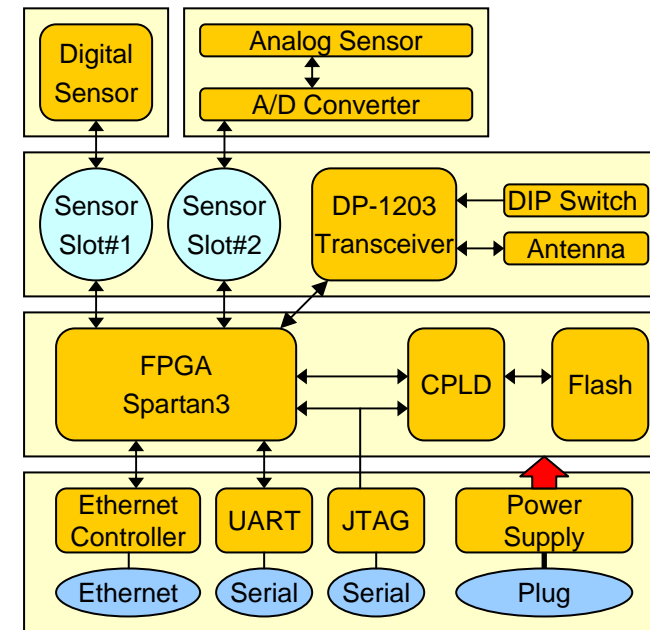
# Die komplette Prototyp-Plattform



- Vorteile dieser Prototyp-Plattform
  - Hohe Flexibilität
    - Die Plattform kann mit praktisch jedem Sensortyp oder Funkchip ausgerüstet werden
    - Dadurch ergeben sich universelle Einsatzmöglichkeiten
  - Sofort funktionsfähig nach dem Einschalten
    - Komfortable Bedienung: keine aufwendigen Set-ups oder Konfiguration nötig
  - Funktionale Anforderungen erfüllt
    - Selbst komplexe Mote-Architekturen können emuliert werden
    - Kombination aus kompakter Größe, hoher Autonomie und großem FPGA
- Beschränkungen:
  - Batterielebensdauer ca. 10 Std. bei konstantem Betrieb (@ 10MHz).
  - Keine externe Verbindung für den Zugriff auf ein oder mehrere Knoten zu Debugging-Zwecken

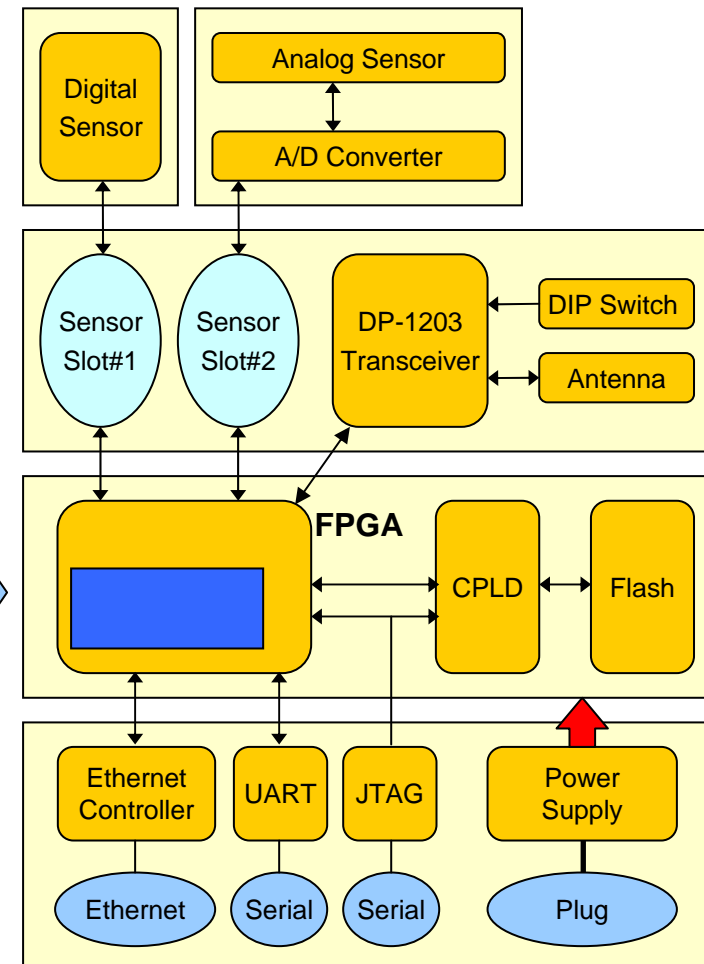
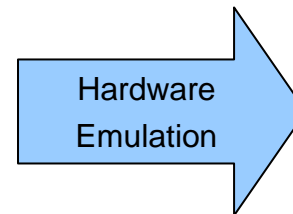
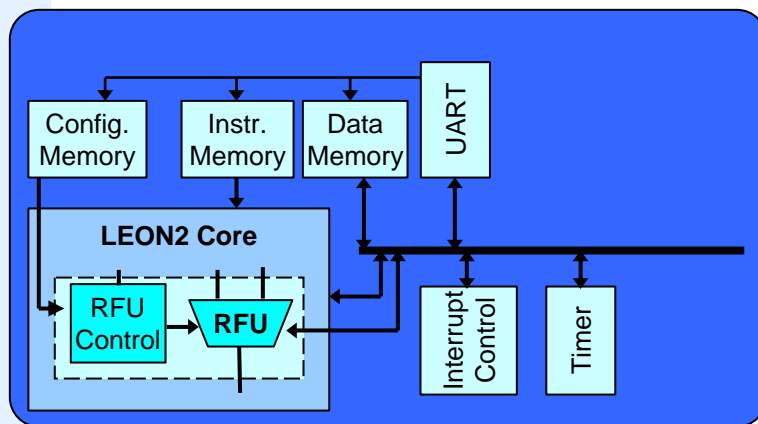
# Erweiterung um ein Basisboard

- Zweck des Basisboards:
  - Netzspannungsversorgung
    - für langfristige Experimente
  - Konnektivität
    - UART (zur Flash-Programmierung)
    - JTAG (zur Konfiguration)
  - Ethernet-Schnittstelle zur Realisierung eines Deployment Support Networks (DSN)
- Ein DSN ist eine gängige Technik zur gleichzeitigen Überwachung vieler Knoten in einem Prototyp-Netzwerk
  - Es handelt sich um ein sekundäres Hilfsnetzwerk nur zu Debugging-Zwecken
  - Ermöglicht zuverlässigen Zugriff auf Knoten
  - Die Kombination aus einem DSN und der Einbettung von Debuggingsschnittstellen im FPGA eröffnet weitreichende Möglichkeiten zur Überwachung der Knoten



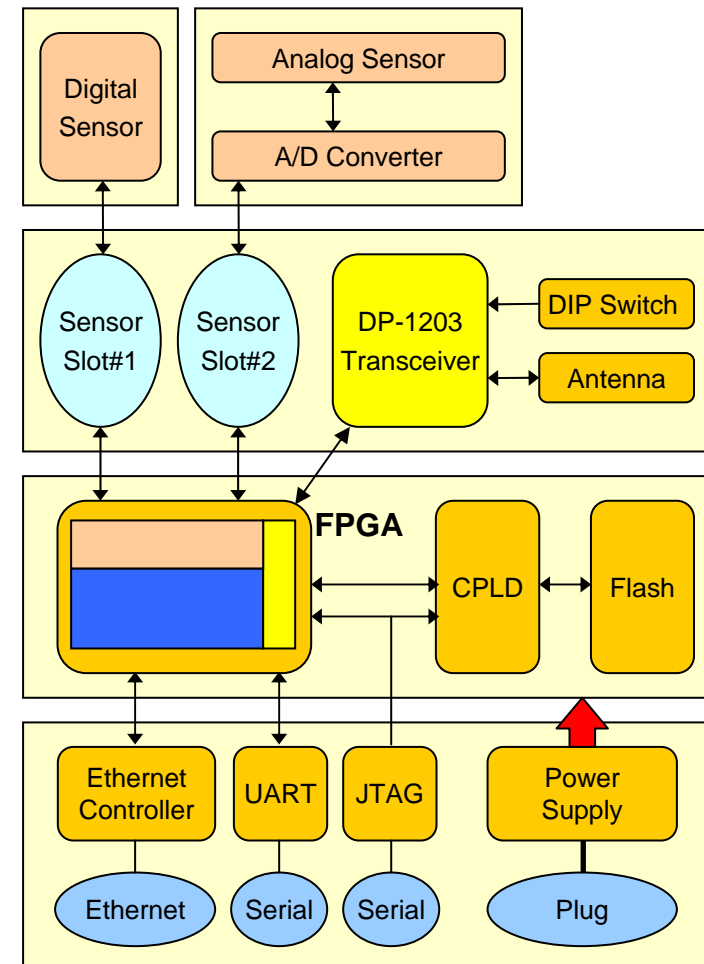
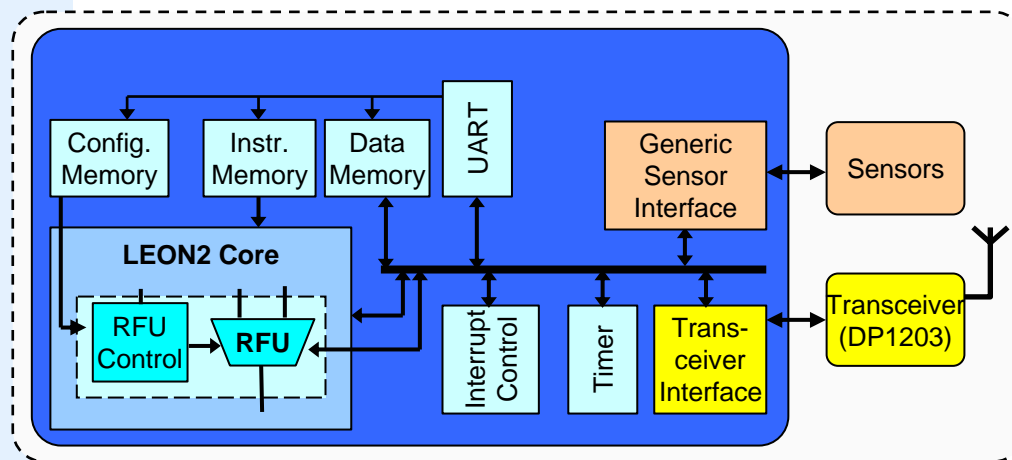
# Anwendungsbeispiel

- 1) Hardware-Emulation
  - Ausgangspunkt ist ein VHDL-Modell unserer rekonfigurierbaren Sensorknotenarchitektur
  - Eigentliche Zieltechnologie: 130nm Standardzellen (ASIC) Technologie



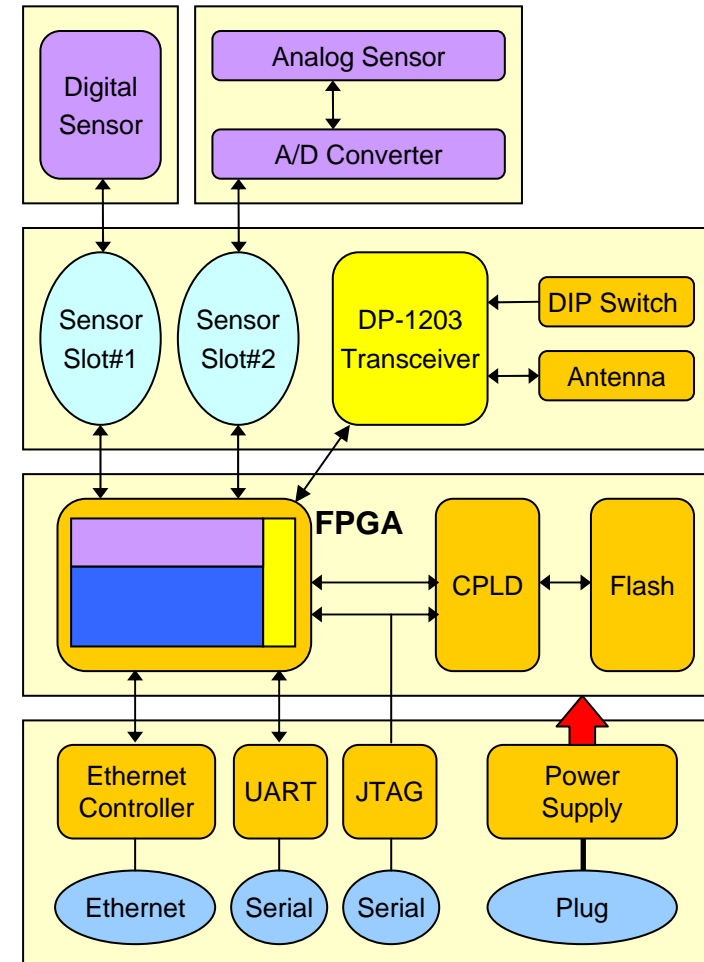
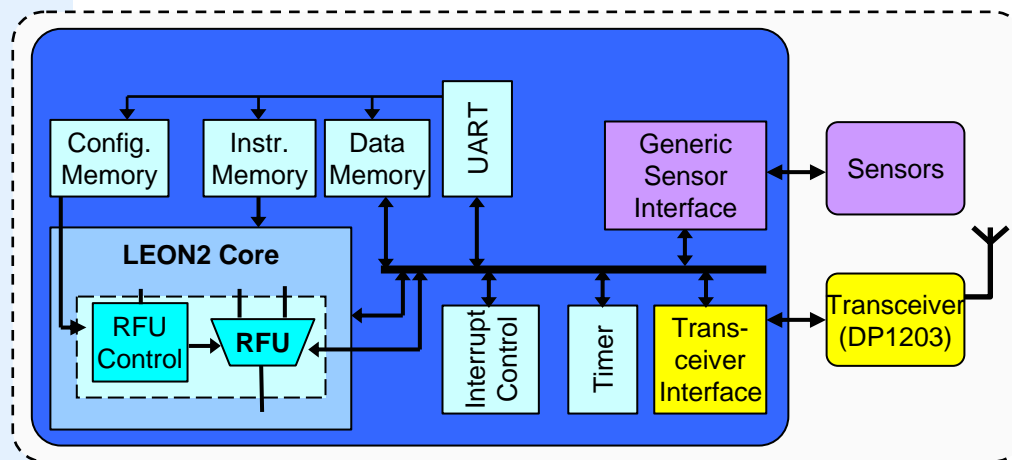
# Anwendungsbeispiel

- 1) Hardware-Emulation
- 2) Sensor- & Funkschnittstellen



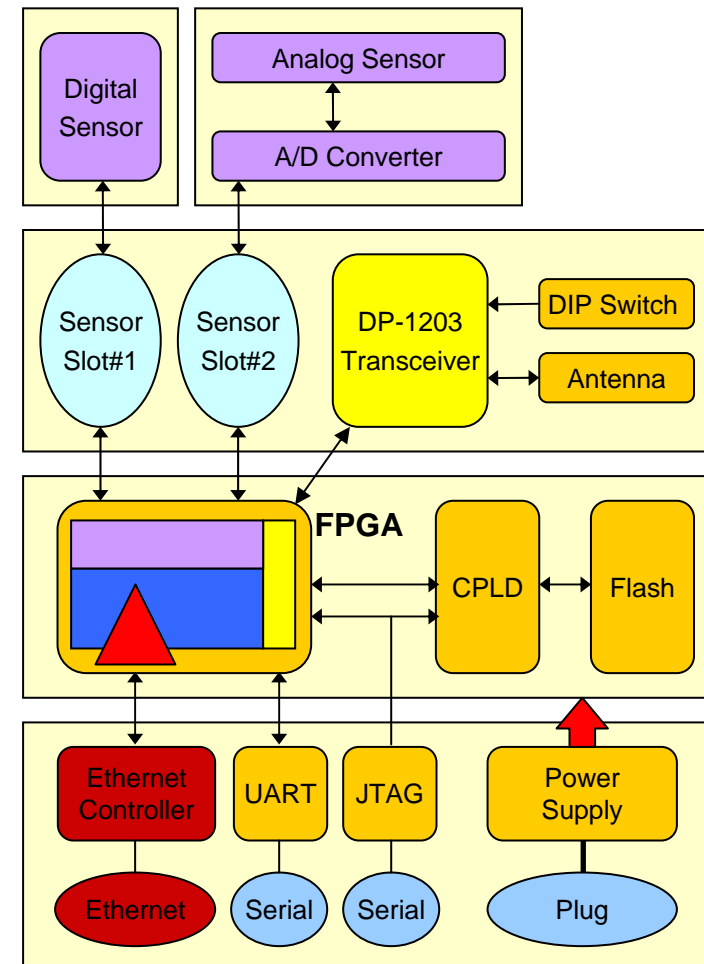
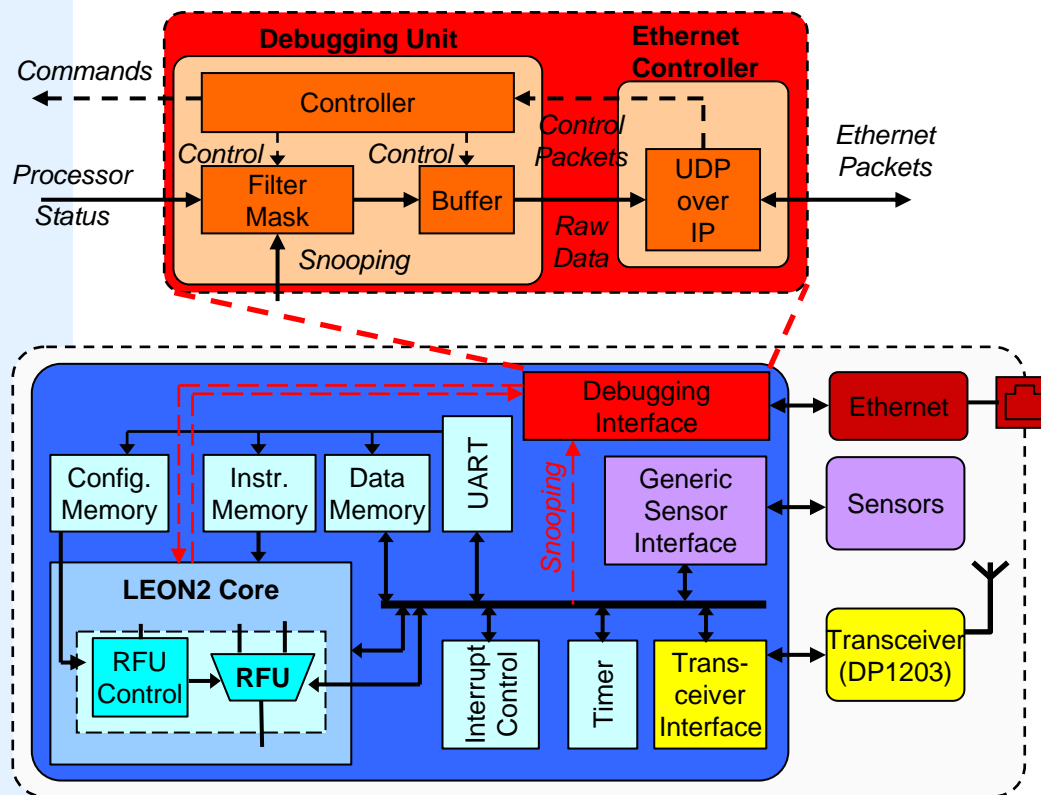
# Anwendungsbeispiel

- 1) Hardware-Emulation
- 2) Sensor- & Funkschnittstellen

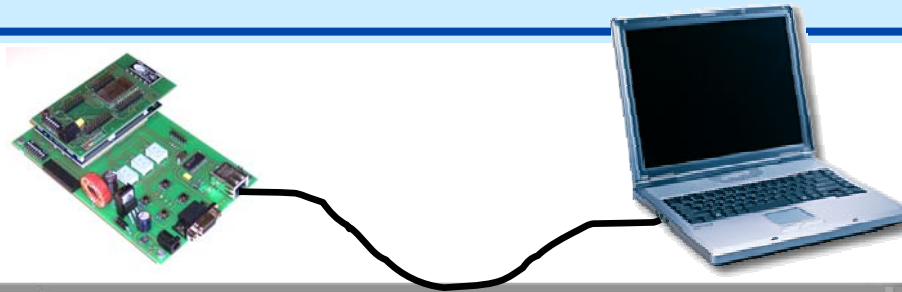


# Anwendungsbeispiel

- 1) Hardware-Emulation
- 2) Sensor- & Funkschnittstellen
- 3) Eingebettete Debuggingschnittstelle



# Grafische Benutzeroberfläche



**Zefant node debugging interface**

Device Selection | Node Control | System Log | Display Options / Tools

Bus transfer selection

<input checked="" type="checkbox"/> Debugging messages (0xFFFFFFFF to 0xFFFFFFFF)	<input checked="" type="checkbox"/> Unknown targets (all other addresses)	Select all	Select none
<input type="checkbox"/> Data memory (0x40000000 to 0x400000FF)	<input checked="" type="checkbox"/> Sensors (0x20000400 to 0x200007FF)	Select debugging	
<input checked="" type="checkbox"/> Radio (0x20000000 to 0x200003FF)	<input checked="" type="checkbox"/> IRQ (0x80000090 to 0x8000009F)	Transmit bus mask	
<input checked="" type="checkbox"/> UART (0x80000200 to 0x800003FF)	<input checked="" type="checkbox"/> Timers (0x80000040 to 0x8000007F)		

CPU status information  
Current CPU status is TT: 00010110 | ICC: 0 | PIL: 0000 | S: true | PS: true | ET: true | CWP: 000

User interaction (right-click for global actions)

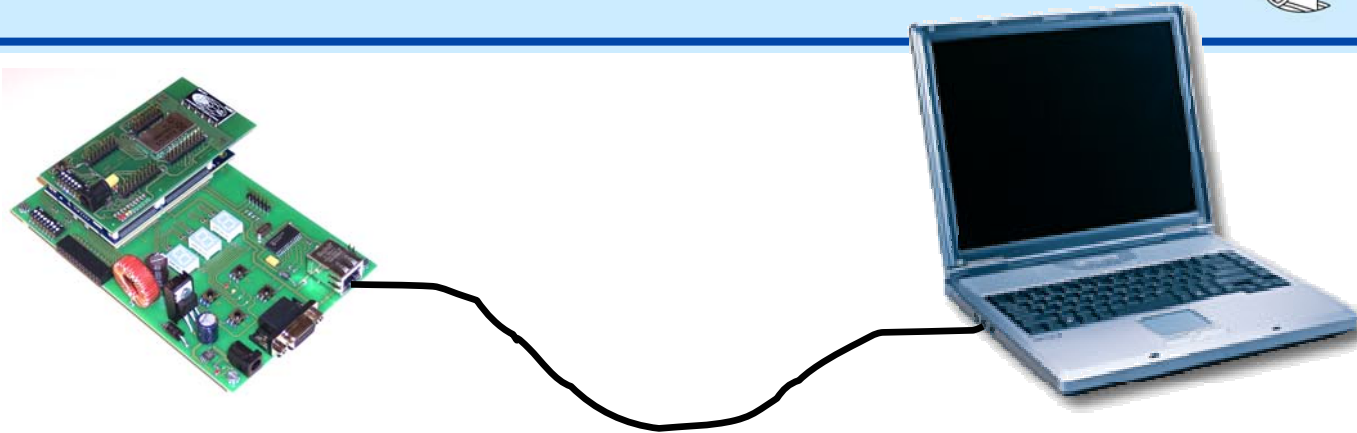
Halt CPU operation | Resume CPU operation | Step through | Toggle PC | Software Reset | Flush log data

Connection status  
Connected to 192.168.0.9 (local port 1061)...

Node messages (double-click to save)

Seq. No.	Pgm Counter (h)	Bus address (h)	Dir	Data (b)	Data (h)	Data (t)
595	00 00 03 64	20 00 04 04	W	00000000 00000000 00000000 00000001	00 00 00 01	....
596	00 00 03 AC	80 00 00 9C	W	00000000 00000000 11111111 11111110	00 00 FF FE	....
597	00 00 03 0C	20 00 02 00	R	00000000 00000000 00000000 00111101	00 00 00 3D	... =
598	00 00 03 10	20 00 02 04	R	01010101 01010101 01010101 10111110	55 55 55 BE	UUU.
599	00 00 03 14	20 00 02 08	R	10101100 00011110 10100110 00110001	AC 1E A6 31	... 1
600	00 00 03 64	20 00 04 04	W	00000000 00000000 00000000 00000000	00 00 00 00	....
601	00 00 03 AC	80 00 00 9C	W	00000000 00000000 11111111 11111110	00 00 FF FE	....

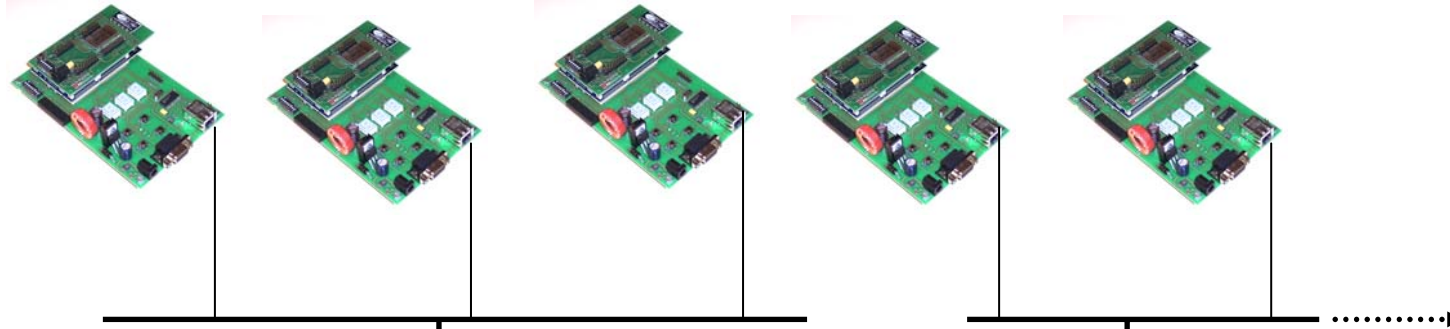
# Grafische Benutzeroberfläche



Node messages (double-click to save)

Seq. No.	Pgm Counter (h)	Bus address (h)	Dir	Data (b)	Data (h)	Data (t)
1	00 00 03 88	FF FF FF FC	W	01010011 01010100 01000001 01010100	53 54 41 54	STAT
2	00 00 03 94	FF FF FF FC	W	01010011 01010001 01001110 01001111	53 51 4E 4F	SQNO
3	00 00 03 9C	FF FF FF FC	W	00000000 00000000 00000001 10000101	00 00 01 85	....
4	00 00 03 A8	FF FF FF FC	W	01001100 01001111 01010011 01010100	4C 4F 53 54	LOST
5	00 00 03 B8	FF FF FF FC	W	00000000 00000000 00000000 00000110	00 00 00 06	....
6	00 00 03 C4	FF FF FF FC	W	01000111 01001111 01001111 01000100	47 4F 4F 44	GOOD
7	00 00 03 CC	FF FF FF FC	W	00000000 00000000 00000000 01011000	00 00 00 58	...X
8	00 00 03 D8	FF FF FF FC	W	01000010 01000001 01000100 00100000	42 41 44 20	BAD
9	00 00 03 E8	FF FF FF FC	W	00000000 00000000 00000000 00000110	00 00 00 06	....
10	00 00 03 88	FF FF FF FC	W	01010011 01010100 01000001 01010100	53 54 41 54	STAT
11	00 00 03 94	FF FF FF FC	W	01010011 01010001 01001110 01001111	53 51 4E 4F	SQNO
12	00 00 03 9C	FF FF FF FC	W	00000000 00000000 00000001 11101101	00 00 01 ED	....
13	00 00 03 A8	FF FF FF FC	W	01001100 01001111 01010011 01010100	4C 4F 53 54	LOST
14	00 00 03 B8	FF FF FF FC	W	00000000 00000000 00000000 00111100	00 00 00 3C	...<
15	00 00 03 C4	FF FF FF FC	W	01000111 01001111 01001111 01000100	47 4F 4F 44	GOOD
16	00 00 03 CC	FF FF FF FC	W	00000000 00000000 00000000 00101000	00 00 00 28	...(
17	00 00 03 D8	FF FF FF FC	W	01000010 01000001 01000100 00100000	42 41 44 20	BAD
18	00 00 03 E8	FF FF FF FC	W	00000000 00000000 00000000 00000000	00 00 00 00	....

# Überwachung kompletter Netzwerke



Deployment Support  
Network  
(LAN)

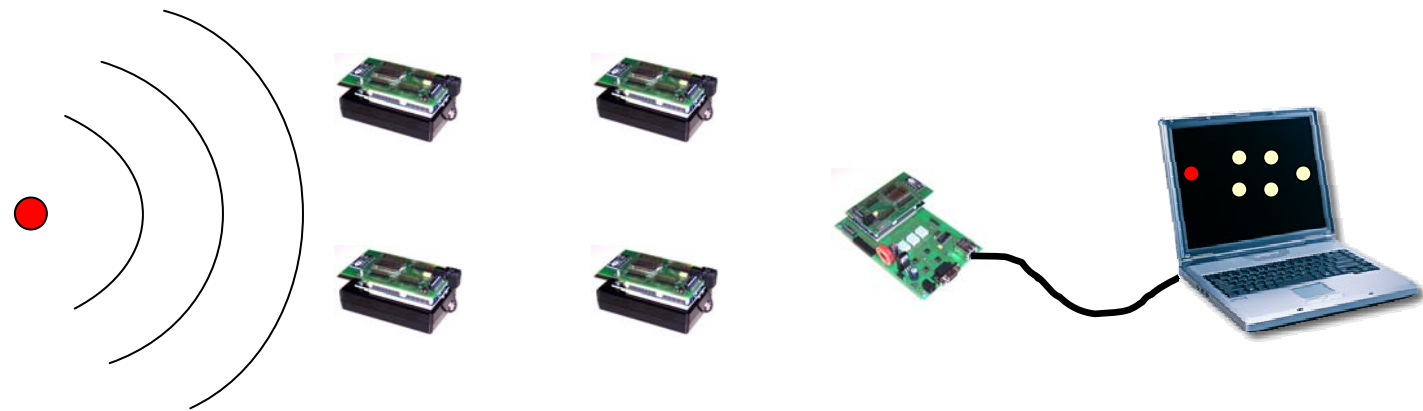


The screenshots show the Zifant node debugging interface. Each window displays the following information:

- Device Selection:** Includes checkboxes for 'Debugging messages (0xFFFFFFFF to 0xFFFFFFFF)', 'Data memory (0x4000000 to 0x40000FF)', 'Radio (0x2000000 to 0x20000FF)', 'UART (0x8000020 to 0x800003F)', 'Unknown targets (all other addresses)', 'Sensors (0x2000400 to 0x20007FF)', 'IRQ (0x8000090 to 0x800009F)', and 'Timers (0x8000040 to 0x800007F)'. There are 'Select all', 'Select none', and 'Transmit bus mask' buttons.
- CPU status information:** Shows 'Current CPU status is TT: 00010110 | ICC: 0 | PSL: 0000 | S: true | PS: true | ET: true | Cwh: 000'. Below this are buttons for 'Halt CPU operation', 'Resume CPU operation', 'Step through', 'Toggle PC', 'Software Reset', and 'Flush log data'.
- Node messages:** A table with columns: Seq. No., Page Counter (h), Bus address (h), Dir, Data (b), Data (h), and Data (i). The table contains several rows of hexadecimal data.

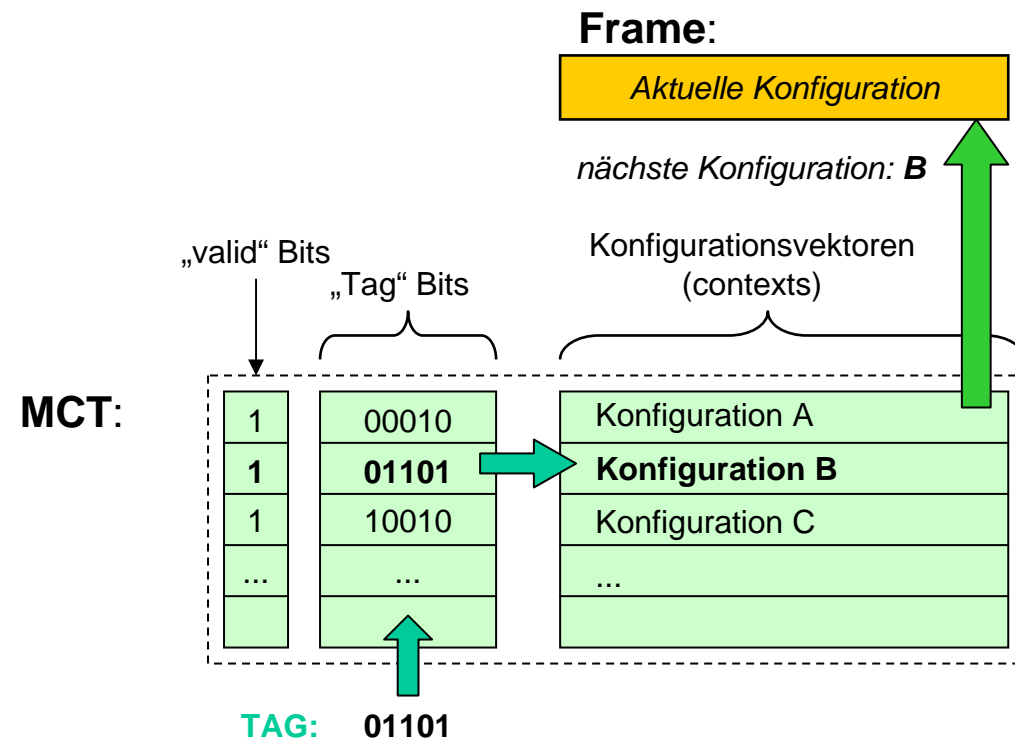
- Der **modulare Aufbau** der Prototyp-Plattform erlaubt den einfachen Einsatz und Austausch von beliebigen Sensoren und Funktransceivern
  - Dadurch ist die Plattform sehr flexibel einsetzbar
- Die Möglichkeit zur Einbettung beliebiger Debuggingfunktionen ist ein wesentlicher Vorteil des „Ein-FPGA“-Ansatzes
- In Kombination mit dem Ethernet-basierten **Deployment Support Network** können dadurch komplette Sensornetzwerke in Echtzeit überwacht und gesteuert werden
  - Eine grafische Oberfläche bietet hierfür bequeme Bedienbarkeit von einem zentralen PC aus
- Die Prototyp-Plattform hat sich daher als wesentliche Hilfe erwiesen für die Verifikation von
  - Hardware-Architekturen,
  - sowie von Software, Protokollen und Applikationen, die auf diesen Architekturen zum Einsatz kommen.
  - Selbst heterogene Sensornetze lassen sich allein mithilfe dieser Plattform einfach prototypisch realisieren.

- Vorführung der Prototypen in der Kaffeepause
- Große Vorführung einer Sensornetzwerkanwendung auf der FPL'08 im Rahmen des Schwerpunktprogramms
  - Geplante Anwendung: Lokalisation von Schallquellen
    - Ausstattung der Prototypen mit Mikrofonen und Ultraschall-Transceivern
    - Positionsbestimmung von Knoten untereinander mittels Ultraschall
    - Lokalisierung externer Schallquellen durch Laufzeitmessungen und anschließende Triangulation.

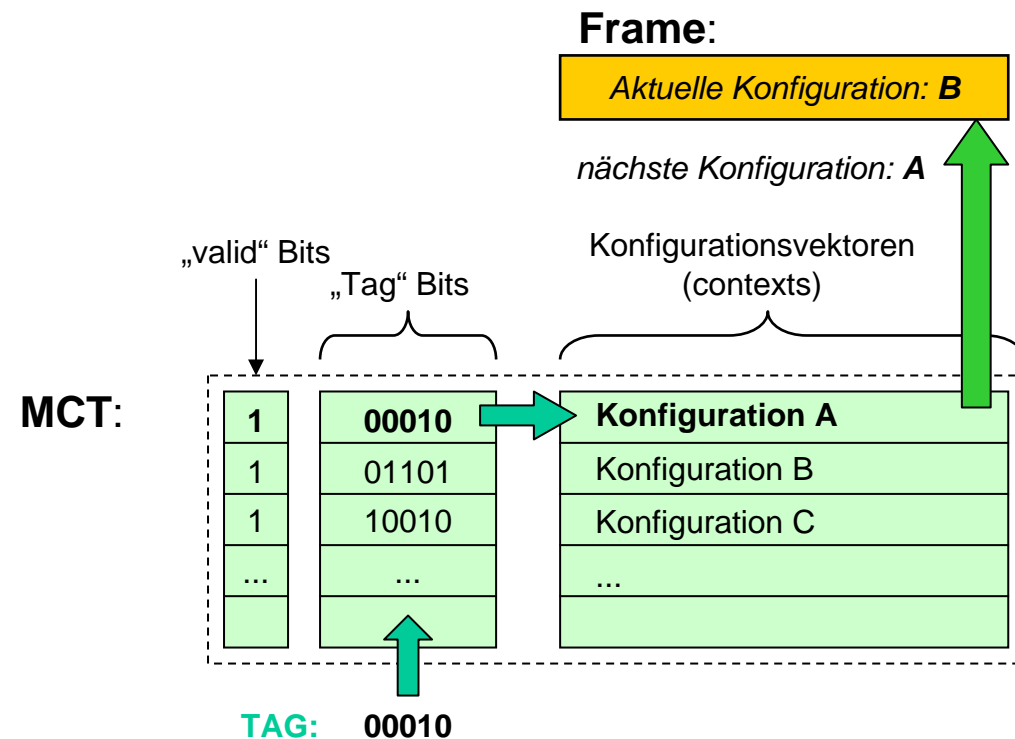


- Zweiter aktueller Forschungsschwerpunkt: **Verfahren zur dynamischen Rekonfiguration anwendungsspezifischer Systeme**
  - Ausgangspunkt: Das von uns in der 2. Projektphase entwickelte Verfahren
    - ist nicht beschränkt auf unsere spezifische RFU-Architektur, sondern praktisch universell einsetzbar
    - besitzt sehr gute Skalierbarkeit
  - Ziele:
    - **Verallgemeinerung des Verfahrens**
      - Wie kann das Verfahren auf andere Architekturen übertragen werden?
    - **Systematische Analyse**
    - **Ableitung eines parametrisierten, skalierbaren Modells**
      - VHDL-Template einerseits
      - Mathematische Modelle zur Abschätzung von Rekonfigurationszeiten, Flächen- und Leistungsverbrauch andererseits
      - als „Design Guide“ oder für die Nutzung in CAD-Werkzeugen

- Grundbaustein des Verfahrens ist die Verwendung sog. **Multi-Kontext Konfigurationstabellen (MCT)**
  - Eine MCT treibt einen einzelnen Frame (atomare Einheit der Konfiguration)
  - Die Auswahl eines Kontexts erfolgt durch Vergleich der „Tags“



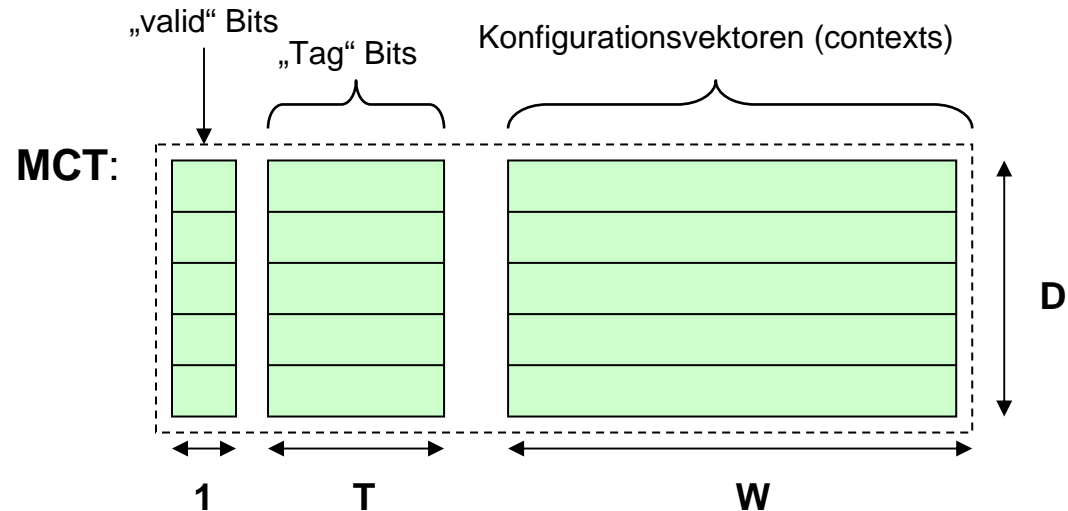
- Grundbaustein des Verfahrens ist die Verwendung sog. **Multi-Kontext Konfigurationstabellen (MCT)**
  - Eine MCT treibt einen einzelnen Frame (atomare Einheit der Konfiguration)
  - Die Auswahl eines Kontexts erfolgt durch Vergleich der „Tags“



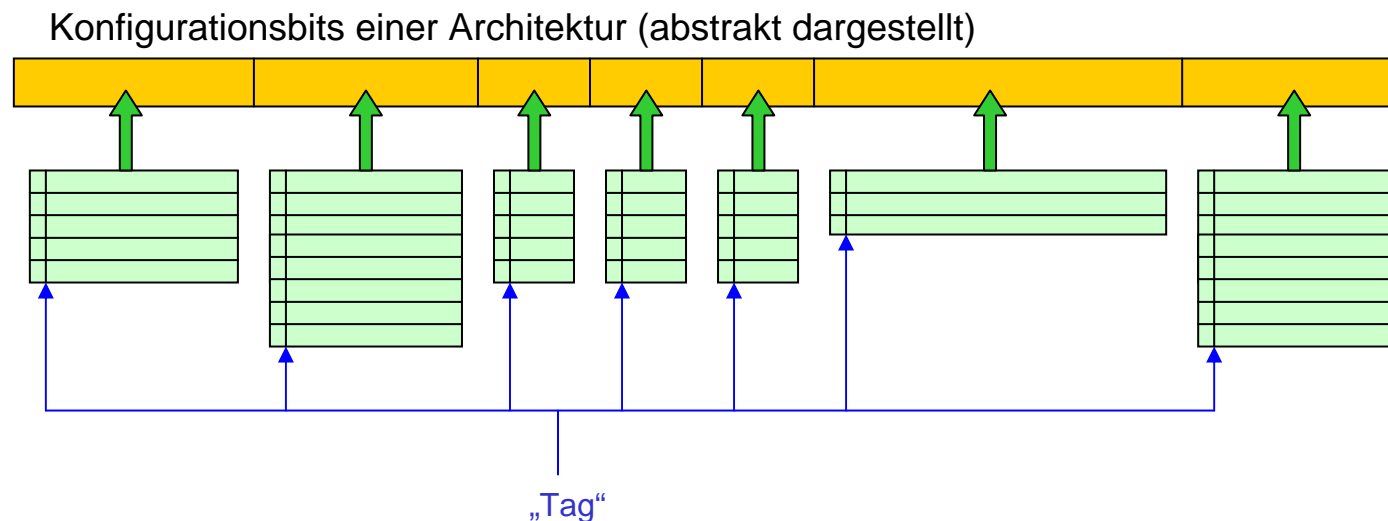
- Die MCT ist leicht parametrisierbar:

- $W$  = Bitbreite des Frames
- $D$  = Tiefe der Tabelle
- $T$  = Bitbreite der Tags

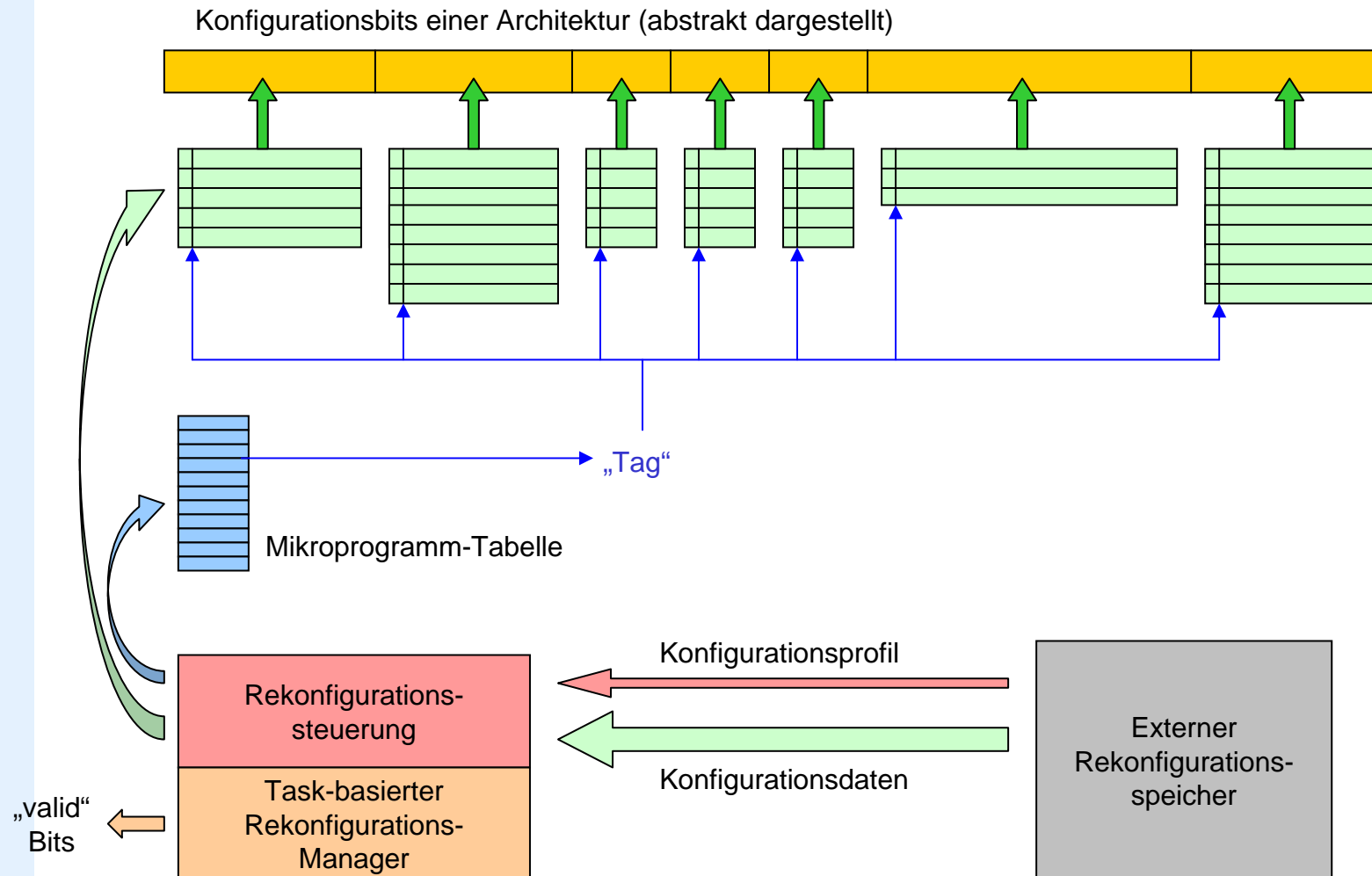
Variable Größen,  
frei wählbar



- Die Gesamtmenge der Konfigurationsbits einer Architektur wird in Frames unterteilt
  - Die Framegröße kann einheitlich oder auch stark verschieden sein
    - Die Parameter jeder MCT können individuell bestimmt werden
    - Anpassung an gegebene Hardwarestruktur möglich
    - Unterstützung auch von heterogenen Architekturen
  - Jeder Frame wird durch eine MCT getrieben

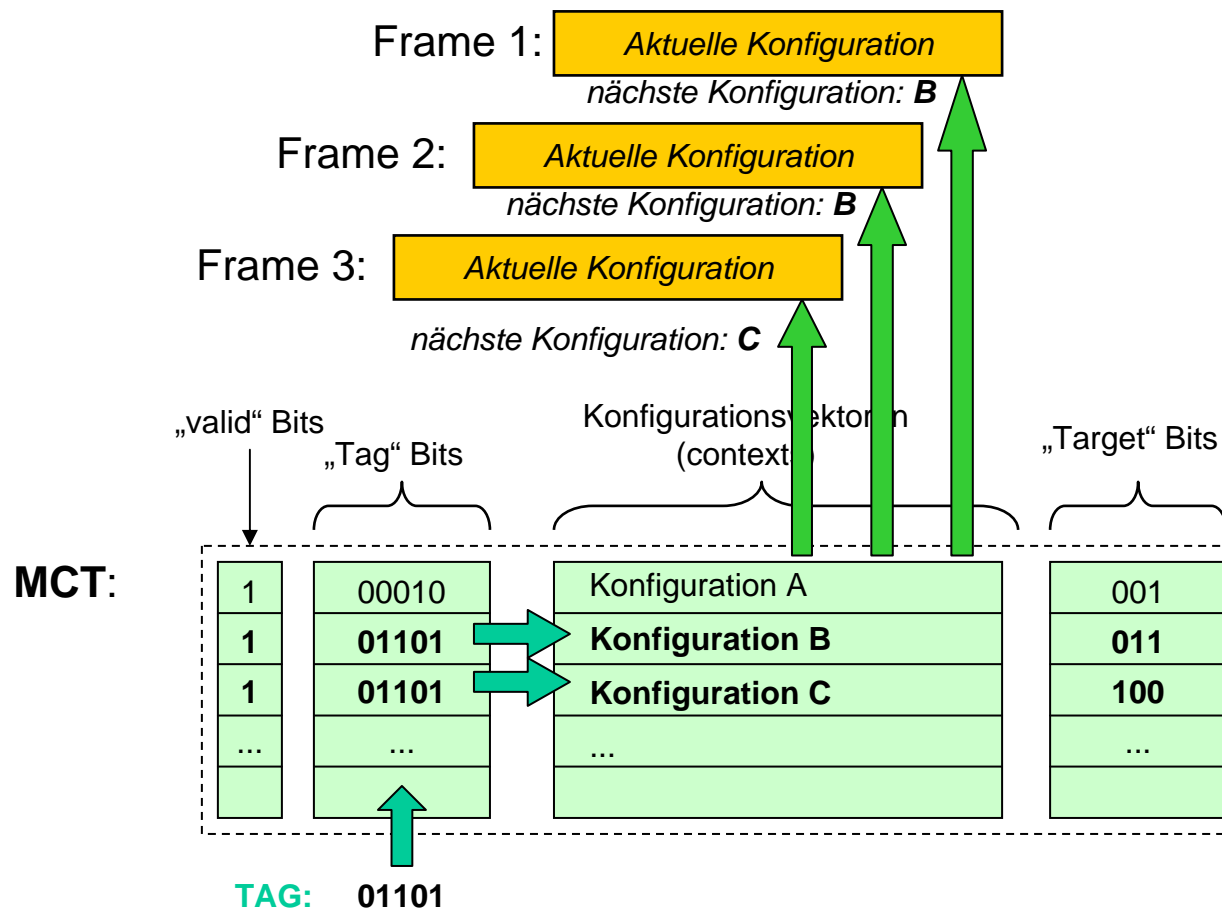


# Das Rekonfigurationsverfahren



# Erweiterung des Verfahrens

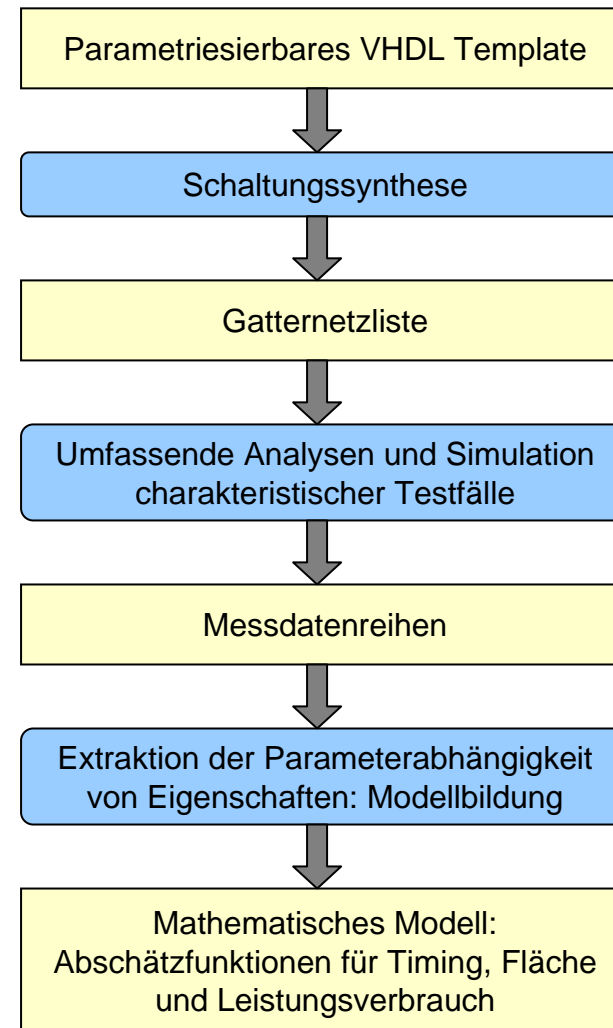
- Eine MCT kann prinzipiell auch mehrere Frames treiben  
→ Multi-Frame MCT



- Die MCT ist leicht parametrisierbar:
  - $W$  = Bitbreite des Frames
  - $D$  = Tiefe der Tabelle
  - $T$  = Bitbreite der Tags

} Variable Größen,  
frei wählbar
- Ableitung entsprechender Modelle:
  - Parametrisierbares VHDL-Template einer MCT
    - Leicht über ‚Generics‘ realisierbar
  - Mathematische Funktionen zur Beschreibung charakteristischer Eigenschaften (Timing, Fläche, Leistungsverbrauch)
    - Oft ist eine Vereinfachung der Funktionen sinnvoll

- **Vorgehensweise:**
  - 1) Erstellung eines VHDL Modells
  - 2) Synthese einer Gatternetzliste
  - 3) Simulation von Messreihen durch Variation einzelner Parameter
  - 4) Analyse der gewonnenen Daten und Ermittlung der Abhängigkeit von Eigenschaften von Parametern
  - 5) Ableitung eines mathematischen Modells in Form von Funktionen zur Abschätzung der Eigenschaften (Timing, Fläche, Power, etc.)



- Beispiel für die MCT

- Abschätzung des Flächenverbrauchs:

- $F_{\text{Fläche}}(D, T, W) = (C_0 + C_1 * W + C_2 * T * W) + (C_3 + C_4 * W) * D$

- $C_0 = 100,0$

- $C_1 = 19,5$

- $C_2 = 57,5$

- $C_3 = 50,0$

- $C_4 = 52,71$

- Die Funktion  $F_{\text{Fläche}}$  liefert eine Abschätzung des Flächenverbrauchs einer MCT, die in 130nm Standardzellen-Technologie realisiert wird

- Die Genauigkeit der Schätzwerte liegt bei unter 1% Abweichung!

- Auf ähnliche Weise werden Funktionen für Rekonfigurationszeiten und Leistungsverbrauch aufgestellt

- $F_{\text{Timing}}$  ist zusätzlich abhängig von der Busdatenbreite des Rekonfigurationsspeichers, aus dem Tabellen geladen werden

- $F_{\text{Power}}$  weist deutliche höhere Variationen auf, die Schätzgenauigkeit liegt hier bei ca. 20% Abweichung

- (Grund: der Leistungsverbrauch ist abhängig von Redundanzen in den Konfigurationsdaten)

- Aus Modellen der einzelnen Grundbausteine lassen sich Modelle gesamter Konfigurationsverfahren zusammenstellen
  - Vergleichbarer Ansatz wie bei der FLPA-Methodik (Aus der Kooperation mit der RWTH Aachen)
- Nutzungsmöglichkeiten:
  - Entwickler können anhand der Modelle bestimmte Entwurfsparameter optimieren
    - z.B. die Wahl geeigneter Frame-Größen
    - Wahl zwischen mehreren MCTs und einer Multi-Frame MCT
  - frühzeitige Kostenabschätzung
  - Entwurfsautomatisierung
    - Durch die enge Verzahnung der mathematischen Funktionen und der VHDL-Modelle über eine gemeinsame Parametermenge lassen sich maßgeschneiderte Hardwareschaltungen automatisiert erzeugen
  - Vergleich verschiedener Verfahren untereinander
    - z.B. Vergleich zwischen „Tag-Matching“ und Adressierung

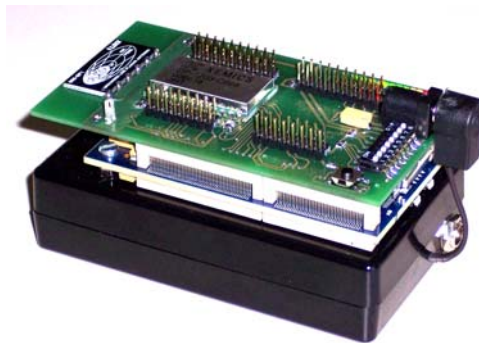
- Für den Vergleich der Verfahren sind geeignete Metriken erforderlich
  - Allerdings unterscheiden sich die Anforderungen an dynamische Rekonfiguration von Fall zu Fall
- Ansatz hierfür ist daher eine **Klassifizierung von typischen Musterfällen**:
  - Gelegentliche Rekonfiguration,  $T_{\text{ex}} \gg T_{\text{conf}}$
  - Rekonfiguration unter harten Echtzeitbedingungen
    - weitere Unterscheidung, ob neue Konfiguration im Voraus bekannt ist
  - Häufige Rekonfiguration zwischen Funktionen,  $T_{\text{ex}} \geq T_{\text{conf}}$ 
    - ebenfalls Unterscheidung, ob neue Konfiguration im Voraus bekannt ist
  - Taktbasierte Rekonfiguration zwischen Rechenschritten
  - ...

- **Ausweitung der Modellierung unseres Rekonfigurationsverfahrens**
  - Ziel ist die Erstellung eines umfassenden skalierbaren Modells und Hardware-Templates
  - Gegenüberstellung mit alternativen Verfahren
    - durch detaillierte Vergleiche der modellierten Kostenfunktionen
    - für verschiedene Anwendungsfälle
- **Einbeziehung weiterer Verfahren zur dynamischen Rekonfiguration**
  - Erstellung einer Modellbibliothek
- **Ableitung geeigneter (anwendungsspezifischer?) Metriken zur Bewertung und insbesondere zur Optimierung von Rekonfigurationsverfahren**

- H. Hinkelmann, A. Reinhardt, M. Glesner :  
**A Methodology for Wireless Sensor Network Prototyping with Sophisticated Debugging Support.** To appear in:  
Proc. 19th IEEE/IFIP Int. Symp. on Rapid System Prototyping (RSP), Juni 2008, Monterey CA, USA.
- A. Reinhardt, H. Hinkelmann, M. Glesner :  
**Developing a Debugging Interface for Reconfigurable Wireless Sensor Nodes.** Accepted at: 7th European  
Workshop on Microelectronics Education (EWME), Mai 2008, Budapest, Ungarn.
- H. Hinkelmann, A. Reinhardt, S. Varyani, M. Glesner :  
**A Reconfigurable Prototyping Platform for Smart Sensor Networks.** In Proc. IV Southern Conference on  
Programmable Logic (SPL 2008), März 2008, Bariloche, Argentinien, pp. 125-130.
- H. Hinkelmann, P. Zipf, M. Glesner :  
**Dynamically Reconfigurable Computing for Wireless Sensor Networks: Why, and How Can This Save Energy.**  
Submitted to: International Journal of Reconfigurable Computing.
- H. Hinkelmann, P. Zipf, M. Glesner :  
**A Domain-Specific Dynamically Reconfigurable Hardware Platform for Wireless Sensor Networks.** In Proc. Int.  
Conf. on Field-Programmable Technology (ICFPT), Dezember 2007, Kitakyushu, Japan, pp. 313-316.
- P. Zipf, H. Hinkelmann, L. Deng, M. Glesner, H. Blume, T. Noll :  
**A Power Estimation Model for an FPGA-Based Softcore Processor.** In Proc. 17th Int. Conf. on Field Programmable  
Logic and Applications (FPL'07), August 2007, Amsterdam, Netherlands, pp. 171-176. *aus der Kooperation TU  
Darmstadt / RWTH Aachen*

# Fragen

?



Demonstration der Prototypen  
in der Kaffeepause

---