Implementierung der Inversen Diskreten Modifizierten Cosinustransformation für MP3 auf FPGA-Zielarchitekturen

Markus Vogt

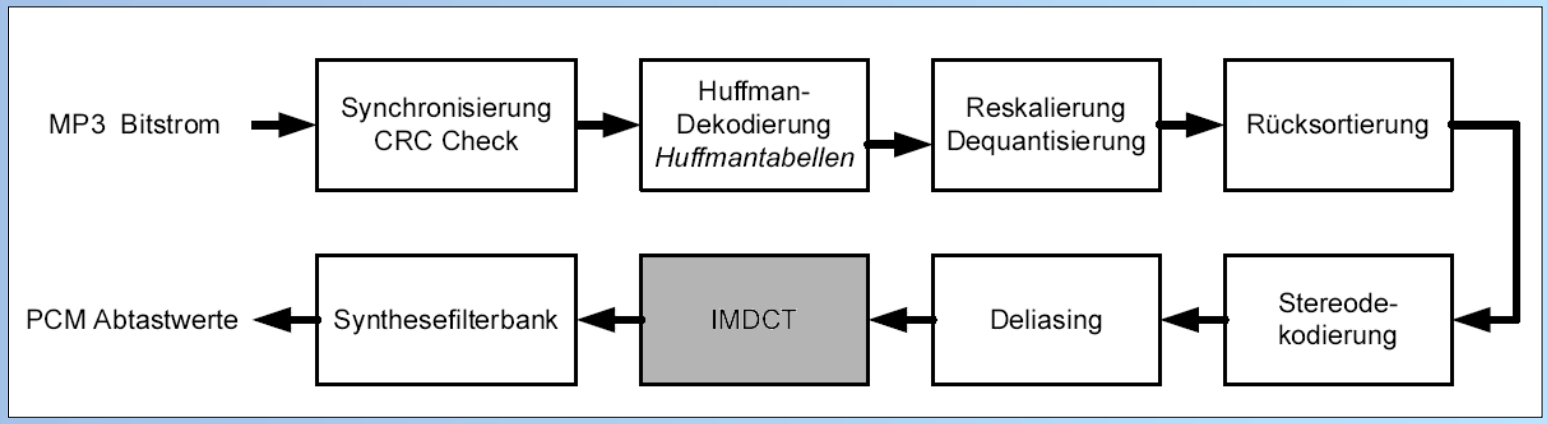
s4402901@inf.tu-dresden.de

Gliederung

- Motivation
- Algorithmus
  - Kurze IMDCT Einführung
  - Die IMDCT bei der Mp3-Dekodierung
- Hardwareimplementierung
  - Randbedingungen
  - Zielarchitektur Virtex II FPGA
  - Einzelheiten des Hardwareentwurfs
- Simulation+Synthese
- Zusammenfassung
- Quellen

Zielstellung

- Zielstellung:
  - Implementierung eines IMDCT-Algorithmus zur Mp3-Dekodierung
  - Zielplattform: FPGA
- Entwurfsumgebung
  - Sprache: VHDL, Tools: NcSim, SimVision, XST
  - Referenzdesign des Mp3-Dekoders in SystemC
    - Gibt Schnittstellen und Funktionsspezifikation vor
    - Ermöglicht Verifikation als Gesamtdesign



Kurze IMDCT Einführung

- IMDCT = Inverse Modifizierte Diskrete Cosinustransformation
- FFT ähnliche Transformation
  - Basierend auf DCT
  - Überlappung aufeinander folgender Transformationen
- Abbildung: N/2 reelle Werte X_k Frequenzbereich → N reelle Werte x_i Zeitbereich

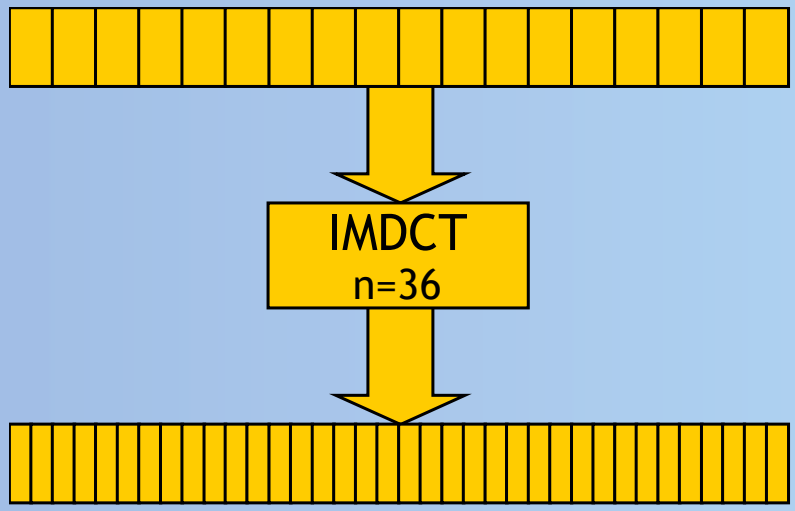
$$x_i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cdot c_{k,i}$$

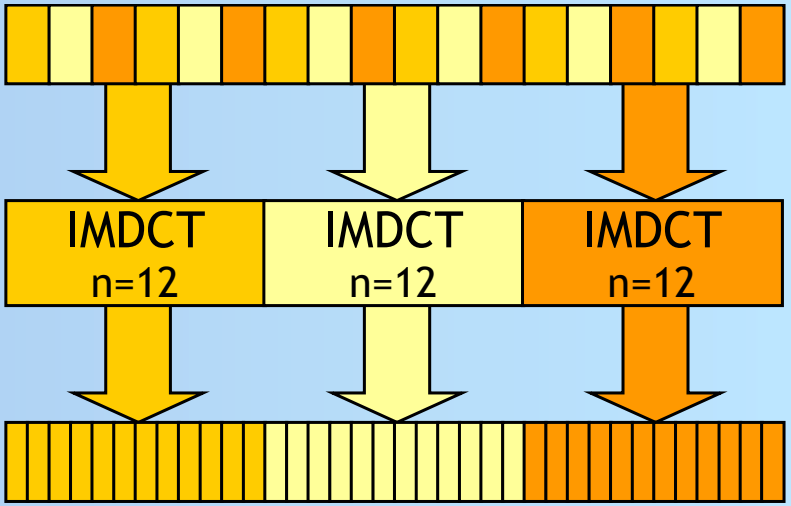
$$c_{k,i} = \cos\left(\frac{\pi}{2n} \cdot \left(2i + 1 + \frac{n}{2}\right) \cdot (2k+1)\right)$$

$$i = 0 \ldots n-1$$

IMDCT bei der Mp3-Dekodierung (1)

- Blockweise
- 18 Eingabewerte, 36 Ausgabewerte
- Long Blocks: Eine 18-Werte-IMDCT (n=36)
- Short Blocks: Drei 6-Werte-IMDCTs (n=12)





IMDCT bei der Mp3-Dekodierung (2)

- Umsetzung als Matrixmultiplikation:

$$(x_0, x_1, \ldots, x_{35}) = (X_0, X_1, \ldots, X_{17}) \bullet \begin{pmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,35} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,35} \\ \vdots & \vdots & \ddots & \vdots \\ c_{17,0} & c_{17,1} & \cdots & c_{17,35} \end{pmatrix}$$

- Weitere Optimierungen nicht sinnvoll
  - Hardwareaufwand steigt
  - Speedup-zu-Overhead-Verhältnis ungünstig wegen geringer Problemgröße
- Overlapping hier nicht realisiert, ist Bestandteil eines separaten Moduls

Randbedingungen

- Zielplattform: Virtex II FPGA von Xilinx
- Angestrebte Taktfrequenz: 800kHz – 1MHz
- Bei 44.1kHz Audiosamplerate ca. 18-22 Takte verfügbare Rechenzeit pro IMDCT-Eingabesample
- Verarbeitungsbreite 36 Bit

Rechenzeitabschätzung

- Ausgabevektor: 36 Werte
- Pro Eingabewert X_k 36 Rechenoperationen nötig
  - X_k mit k-ter Zeile der Konstantenmatrix multiplizieren
  - Jeweils 1xMUL, 1xADD/SUB
  - = 1xMAC = 1 Operation
- Zeitquantum: 18-22 Takte
- 2 Berechnungen pro Takt parallel nötig:

$$x_{2k-1} = x_{2k-1} + X_i \cdot c_{i,2k-1}$$

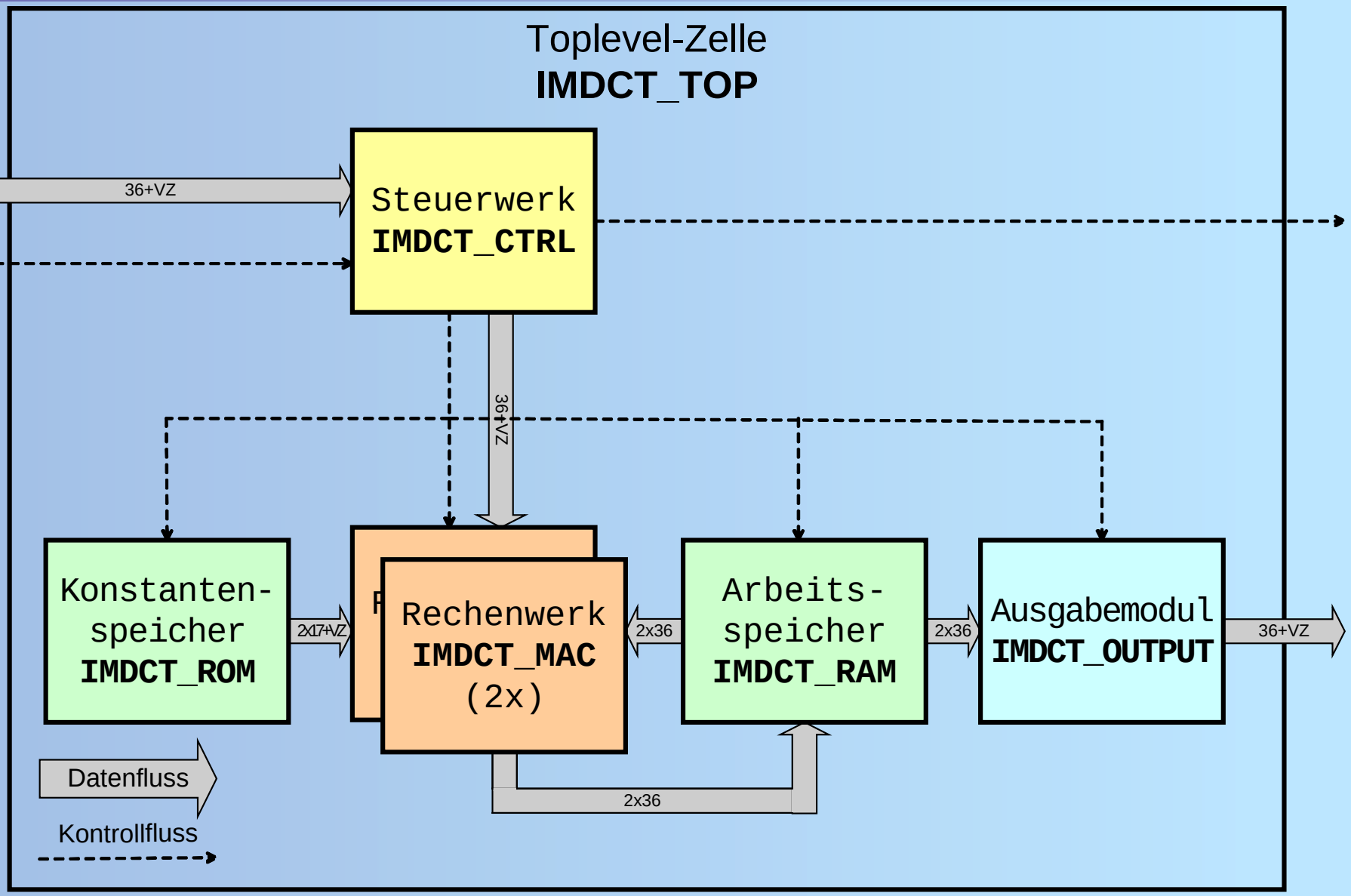
$$x_{2k} = x_{2k} + X_i \cdot c_{i,2k}$$

$$(i,k = 0..17)$$

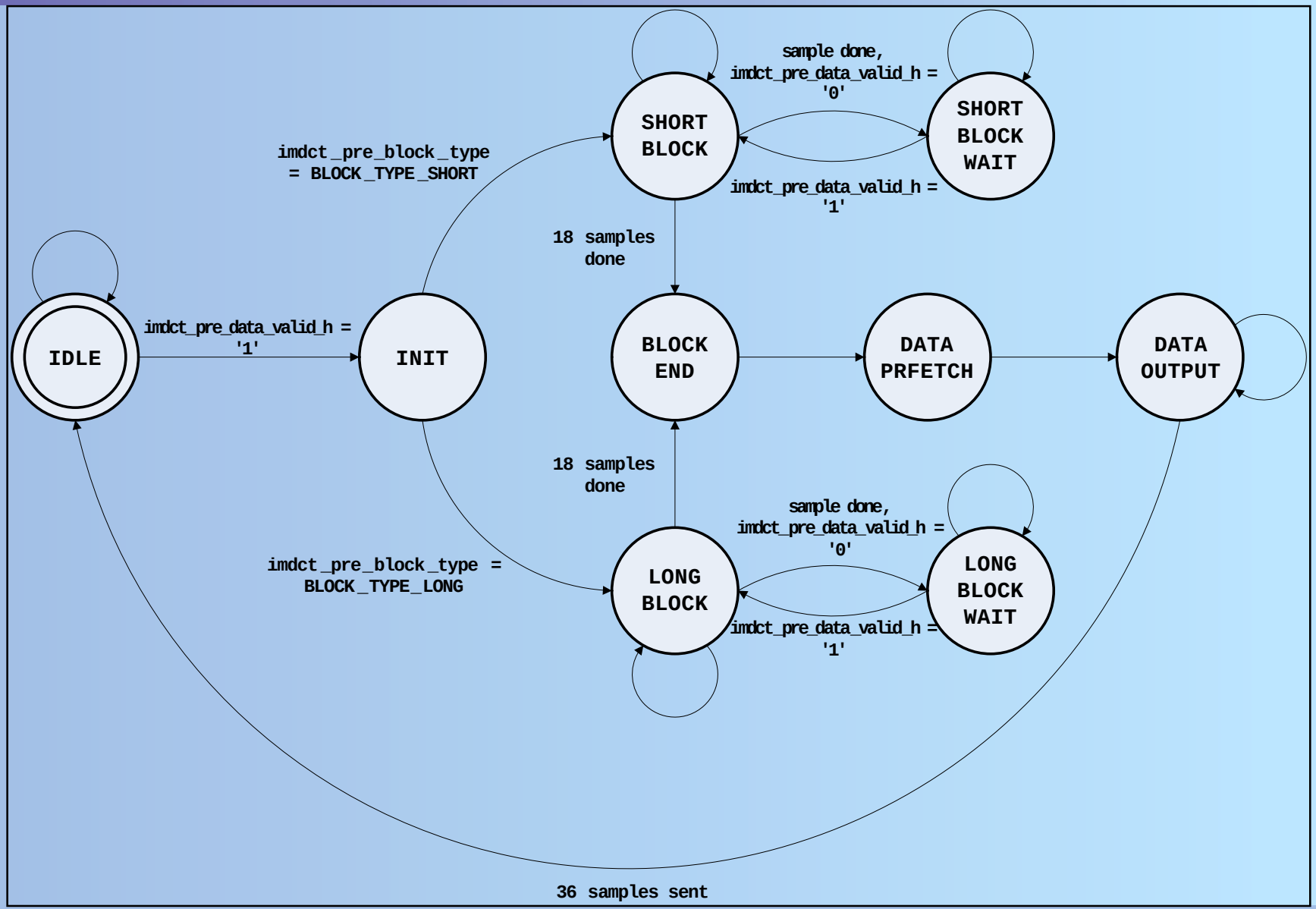
Zielarchitektur Virtex II FPGA

- Wichtige Features:
  - 40 18x18Bit Hardware-Multiplizierer
  - 40 DualPort BlockRAM, je 512*36 Bit = 18432 Bit
- Multiplizierer
  - parallel, Berechnung mit Sicherheit in 1 Takt fertig
- BlockRAM
  - 36 Bit Wortbreite (32 Data, 4 Parität)
  - DualPort: Lesen **und** Schreiben von **2x36 Bit pro Takt**
  - Schafft nötigen Datendurchsatz, um 2 Operationen/Takt durchführen zu können

Überblick


Toplevel-Zelle
IMDCT_TOP
36+VZ
Steuerwerk
IMDCT_CTRL
36+VZ
Konstanten-
speicher
IMDCT_ROM
2x17+VZ
Rechenwerk
IMDCT_MAC
(2x)
2x36
Arbeits-
speicher
IMDCT_RAM
2x36
Ausgabemodul
IMDCT_OUTPUT
36+VZ
2x36
Datenfluss
Kontrollfluss

Steuerwerk: FSM



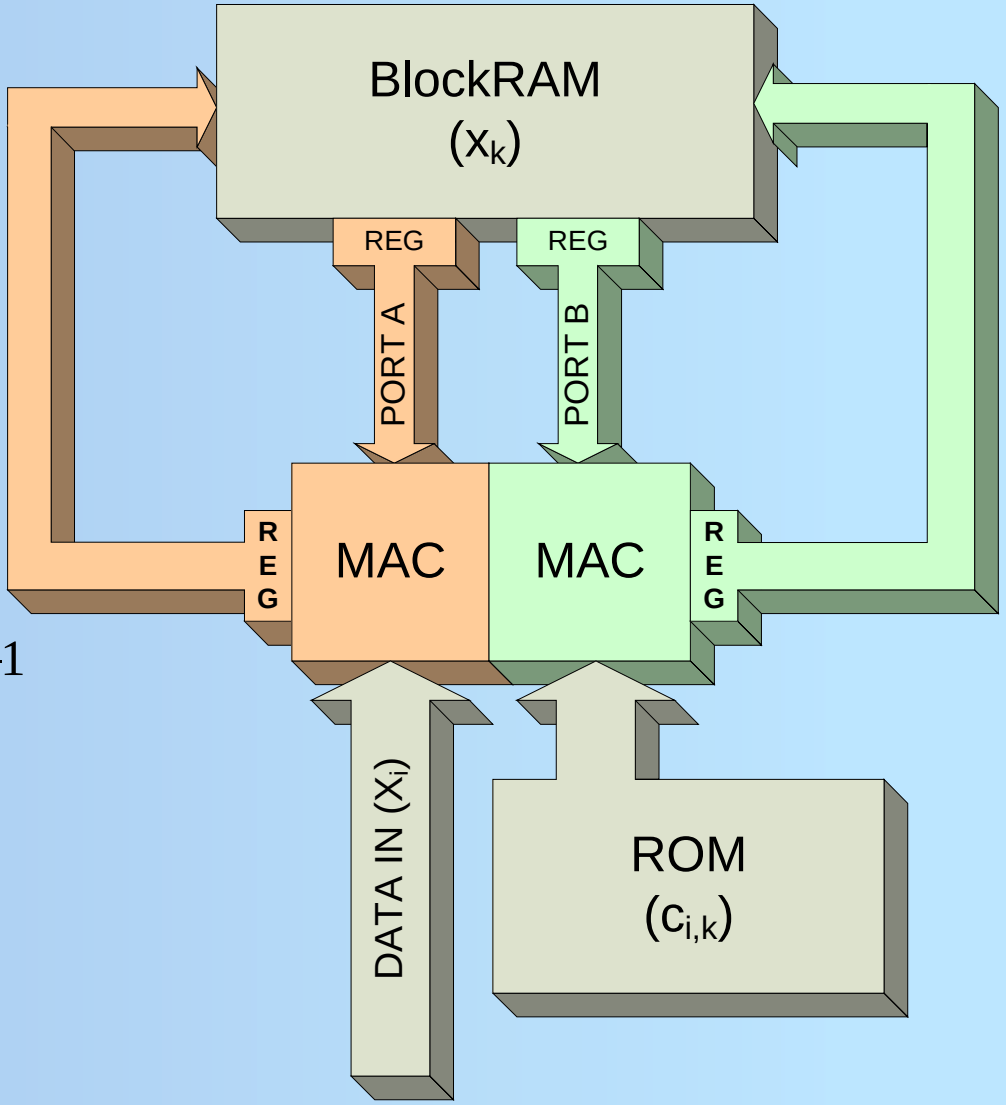
Datenfluss

- Datenformate:
  - RAM 2er-Komplement, 36 Bit
  - ROM Festkomma 0.17 mit VZ, 2x18 Bit je Datenwort
- MAC:

$$x_{2k-1} = x_{2k-1} + X_i \cdot c_{i,2k-1}$$

$$x_{2k} = x_{2k} + X_i \cdot c_{i,2k}$$

  - keine Nachkommastellen



VHDL Codebeispiel

- RAM und ROM werden als Primitive aus Library instantiiert (BlockRAM)
- Codeausschnitt aus IMDCT_MAC.VHD

```
process (
     cos_value, mac_sample_in, mac_sample_sign, mac_accu_in, prod, sign,
     mac_load_accu_h, accu, pipeline_hold_h, data_out_reg
)
begin
     prod <= std_logic_vector( unsigned( cos_value(16 downto 0) ) * unsigned( mac_sample_in ));
     sign <= cos_value(17) xor mac_sample_sign;
     if ( mac_load_accu_h = '1' ) then
          accu <= mac_accu_in;
     else
          accu <= ( others => '0' );
     end if;
     if ( pipeline_hold_h = '0' ) then
          if ( sign = '1' ) then
               next_data_out_reg   <= std_logic_vector( unsigned( accu ) - unsigned( prod( 52 downto 17 )));
          else
               next_data_out_reg   <= std_logic_vector( unsigned( accu ) + unsigned( prod( 52 downto 17 )));
          end if;
     else
          next_data_out_reg   <= data_out_reg;
     end if;
end process;
```

Vergleich Long Blocks, Short Blocks (1)

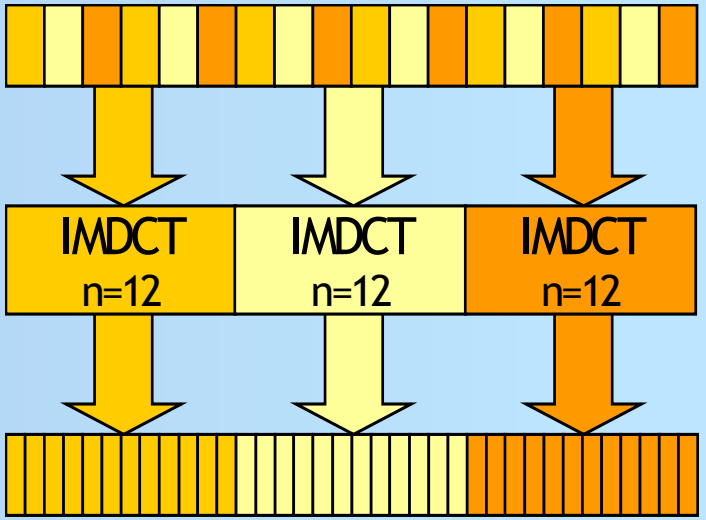
- Long Blocks
  - 18 Eingabewerte → 36 Ausgabewerten, eine IMDCT, Implementierung einfach, eine sequenzielle Matrixmultiplikation
  - einfaches Auslesen des ROMs mit Aufwärtszähler, alle Werte werden nur 1x benötigt
- Short Blocks
  - 3 IMDCTs, Eingabewerte ineinander Verschränkt
  - 3 Matrixmultiplikationen mit kleinerer Konstantenmatrix nötig

$$(x_0, x_1, \ldots, x_{11}) = (X_0, X_1, \ldots, X_5) \bullet \begin{pmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,11} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,11} \\ \vdots & \vdots & \ddots & \vdots \\ c_{5,0} & c_{5,1} & \cdots & c_{5,11} \end{pmatrix}$$

Vergleich Long Blocks, Short Blocks (2)

- Lösung: Konstruktion einer speziellen Konstantenmatrix

$$\underline{x} = \underline{X} \bullet \begin{pmatrix} c_{0,0}, c_{0,1}, \cdots c_{0,11} & 0 & 0 \\ 0 & c_{0,0}, c_{0,1}, \cdots c_{0,11} & 0 \\ 0 & 0 & c_{0,0}, c_{0,1}, \cdots c_{0,11} \\ c_{1,0}, c_{1,1}, \cdots c_{1,11} & 0 & 0 \\ 0 & c_{1,0}, c_{1,1}, \cdots c_{1,11} & 0 \\ 0 & 0 & c_{1,0}, c_{1,1}, \cdots c_{1,11} \\ \vdots & \vdots & \vdots \\ c_{5,0}, c_{5,1}, \cdots c_{5,11} & 0 & 0 \\ 0 & c_{5,0}, c_{5,1}, \cdots c_{5,11} & 0 \\ 0 & 0 & c_{5,0}, c_{5,1}, \cdots c_{5,11} \end{pmatrix}$$



$mit:$

$$\underline{x} = (x_1, x_2, \cdots, x_{11,} y_1, y_2, \cdots, y_{11,} z_1, z_2, \cdots, z_{11})$$

$$\underline{X} = (X_0, Y_0, Z_0, X_1, Y_1, Z_1, \cdots, X_0, Y_0, Z_0)$$

Vergleich Long Blocks, Short Blocks (3)

- Vorteile
  - Short Blocks lassen sich rein algorithmisch wie Long Blocks behandeln
  - Ineinander verschränkte Eingabewerte werden sortiert
  - Berechnung trotz „gleicher Matrixgröße“ schneller als Long Block (Nullelemente der Matrix werden ausgelassen)
- Spezielle Adressierung des Konstantenspeichers nötig
  - Separate 6x12 Matrix im ROM
  - Jede Matrixzeile wird mehrmals benötigt
  - Layout: Zeile 1 → 180h, Zeile 2 → 190h, ...
  - Vereinfacht Adressierung

Abschätzung der Taktfrequenz

- Long Block
  - 18 Werte → 36 Werte
  - 366 Takte, davon 324 reine Rechenzeit
- Short Block
  - 3x6 Werte → 3x12 Werte
  - 150 Takte, davon 108 reine Rechenzeit
- Minimale Taktfrequenz zur Dekodierung eines 44.1kHz Mp3-Audiostreams: 900kHz

Syntheseergebnis (1)

- Macroanalyse

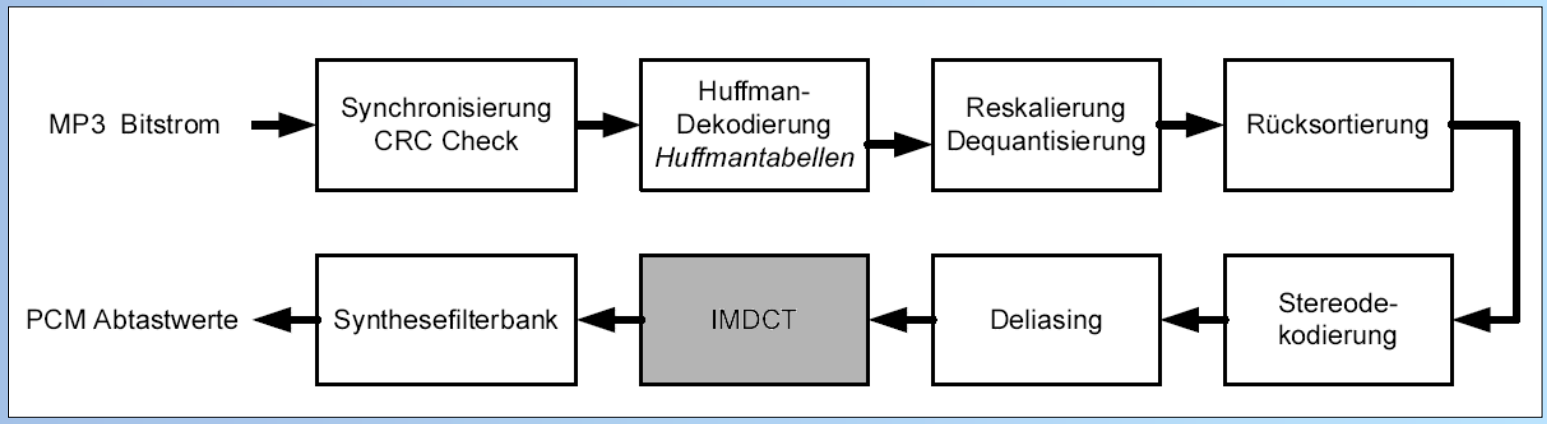
```
Macro Statistics
# FSMs                          : 1
# Multipliers                   : 2
 36x17-bit multiplier           : 2
# Adders/Subtractors            : 8
 3-bit adder                    : 1
 36-bit adder                   : 1
 36-bit addsub                  : 2
 5-bit adder                    : 2
 6-bit adder                    : 1
 9-bit adder                    : 1
# Registers                     : 26
 1-bit register                 : 17
 3-bit register                 : 1
 36-bit register                : 3
 37-bit register                : 1
 5-bit register                 : 2
 6-bit register                 : 1
 9-bit register                 : 1
# Multiplexers                  : 1
 36-bit 4-to-1 multiplexer      : 1
# Xors                          : 2
 1-bit xor2                     : 2
```

Syntheseergebnis (2)

- Hardware-Auslastung der FPGA

Number of Slices:	288	out of	5120	5%
Number of Slice Flip Flops:	190	out of	10240	1%
Number of 4 input LUTs:	530	out of	10240	5%
Number of bonded IOBs:	83	out of	328	25%
Number of BRAMs:	2	out of	40	5%
Number of MULT18X18s:	4	out of	40	10%

- Gesamtauslastung: 5% (Multiplizierer 10%)



Zusammenfassung

- IMDCT für Mp3-Dekoder
  - 18 Werte → 36 Werte (Overlapping!)
  - Sequenzielle Matrixmultiplikation
  - 2 Elemente parallel
  - Short Block, Long Blocks
- Hardware
  - 4 Hardwaremultiplizierer 18x18 Bit
  - 2 BlockRAM á 512*36 Bit = 18432 Bit
  - 1 FSM (71 Flipflops, LUTs für Kombinatorik)
  - ca. weitere 100 FF als Pufferregister im Datenpfad
  - ca. 5% Auslastung des FGPA
- Timing
  - 366 Takte / Long Block = 18 Samples effektiv
  - 44.1kHz Audio benötigt mind. 900kHz Systemtakt

Quellen

Quellen

- [1] Wikipedia, MDCT
  (http://en.wikipedia.org/wiki/Modified_discrete_cosine_transform)
- [2] Wikipedia, DCT
  (http://en.wikipedia.org/wiki/Discrete_cosine_transform)
- [3] Mp3 Format nach ISO
  (http://www.mp3-tech.org/programmer/docs/iso11172-3.zip)
- [4] Xilinx XST Manual
- [5] Xilinx Virtex II Users Guide
  (http://www.xilinx.com)
- [6] Einführungsvorlesung Mp3-Hardwaredekoder
  (http://www.iee.et.tu-dresden.de/~ads-root/MP3_Intro.pdf)

Vielen Dank für die Aufmerksamkeit!

Fragen?