Implementierung der 1D-DFT in VHDL und Erweiterung zur 2D-DFT

Aktueller Stand der Bachelorarbeit

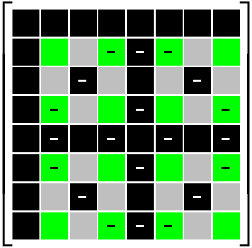
Thomas Lattmann

HAW-Hamburg, ISAR-Projektmeeting

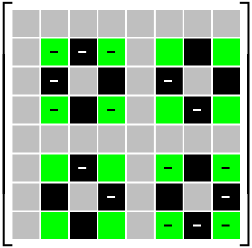
8.1.2018

Veranschaulichung

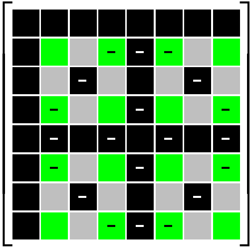
$\Re\{W\}$



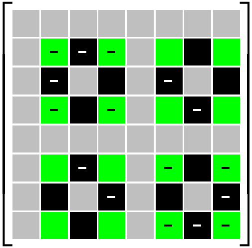
$\Im\{W\}$

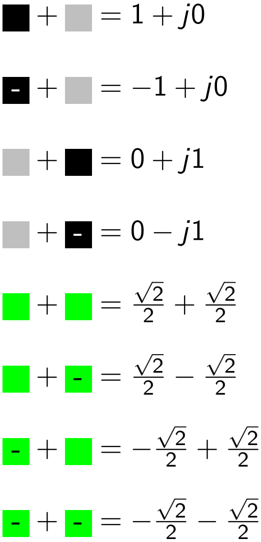


$\Re\{W\}$



$\Im\{W\}$

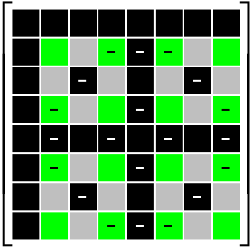




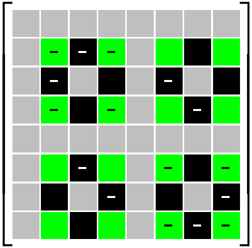
$= 1 + j0$
$= -1 + j0$
$= 0 + j1$
$= 0 - j1$
$= \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}$
$= \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}$
$= -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}$
$= -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}$

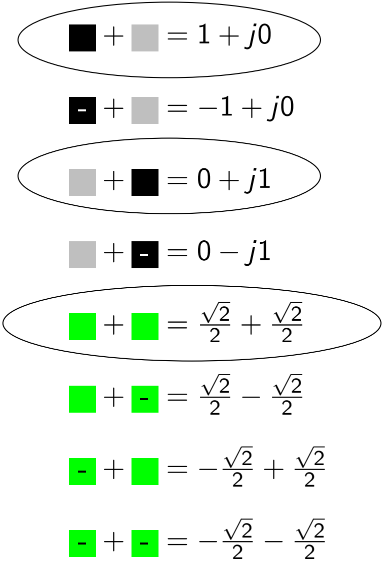
Veranschaulichung

$\Re\{W\}$



$\Im\{W\}$





Komplexe Eingangsmatrix

$$Input = \begin{bmatrix} a_{00} + jb_{00} & a_{01} + jb_{01} & \dots & a_{07} + jb_{07} \\ a_{10} + jb_{10} & a_{11} + jb_{11} & & \\ \vdots & & \ddots & \vdots \\ a_{70} + jb_{70} & & \dots & a_{77} + jb_{77} \end{bmatrix}$$

$$\Re\{Input\} = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{07} \\ a_{10} & a_{11} & & \\ \vdots & & \ddots & \vdots \\ a_{07} & & \dots & a_{77} \end{bmatrix} \qquad \Im\{Input\} = \begin{bmatrix} b_{00} & b_{01} & \dots & b_{07} \\ b_{10} & b_{11} & & \\ \vdots & & \ddots & \vdots \\ b_{70} & & \dots & b_{77} \end{bmatrix}$$

In der VHDL-Implementierung sind Real- und Imaginärteil von einander getrennt (`input_real` und `input_imag`).

Komplexe Multiplikation

$x + jy$: komplexer Twiddlefaktor
$a + jb$: komplexer Eingangswert, cos() + j sin()

$$(x + jy) \cdot (a + jb) = xa + jya + jxb + j^2 yb$$

$$= \overset{\Re}{xa - yb} + j\overset{\Im}{(ya + xb)}$$

Komplexe Multiplikation

$x + jy$: komplexer Twiddlefaktor
$a + jb$: komplexer Eingangswert, cos() + j sin()

$$(x + jy) \cdot (a + jb) = xa + jya + jxb + j^2 yb$$

$$= \overset{\Re}{xa - yb} + j\overset{\Im}{(ya + xb)}$$

$$\overset{\Re}{\blacksquare} + \overset{\Im}{\square} \Rightarrow 1 + j0 \cdot a_{kl} + jb_{kl} \quad = \quad a_{kl} + jb_{kl}$$

Komplexe Multiplikation

$x + jy$: komplexer Twiddlefaktor
$a + jb$: komplexer Eingangswert, cos() + j sin()

$$(x+jy)\cdot(a+jb) = xa + jya + jxb + j^2 yb$$

$$= \underbrace{xa - yb}^{\Re} + j\underbrace{(ya + xb)}^{\Im}$$

$\Re$		$\Im$			
$\blacksquare$	$+$	$\square$	$\Rightarrow 1 + j0 \cdot a_{kl} + jb_{kl}$	$=$	$a_{kl} + jb_{kl}$
$\square$	$+$	$\blacksquare$	$\Rightarrow 0 + j1 \cdot a_{kl} + jb_{kl}$	$=$	$-b_{kl} + ja_{kl}$

Komplexe Multiplikation

$x + jy$: komplexer Twiddlefaktor
$a + jb$: komplexer Eingangswert, cos() + j sin()

$$(x + jy) \cdot (a + jb) = xa + jya + jxb + j^2 yb$$

$$= \overbrace{xa - yb}^{\Re} + j\overbrace{(ya + xb)}^{\Im}$$

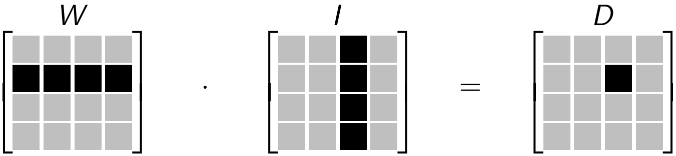
$\Re$ $\Im$

$+ \quad \Rightarrow 1 + j0 \cdot a_{kl} + jb_{kl} = a_{kl} + jb_{kl}$

$+ \quad \Rightarrow 0 + j1 \cdot a_{kl} + jb_{kl} = -b_{kl} + ja_{kl}$

$+ \quad \Rightarrow \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2} \cdot a_{kl} + b_{kl} = \frac{\sqrt{2}}{2} \cdot a_{kl} - \frac{\sqrt{2}}{2} \cdot jb_{kl} + j\frac{\sqrt{2}}{2} \cdot a_{kl} + j\frac{\sqrt{2}}{2} \cdot b_{kl}$

Matritzenmultiplikation



Zeile W_m · Spalte I_n = Element D_{mn}

1. Spalte der komplexen Eingangsmatrix

$$Input(:,0) = \begin{bmatrix} a_{00} + jb_{00} \\ a_{10} + jb_{10} \\ a_{20} + jb_{20} \\ a_{30} + jb_{30} \\ a_{40} + jb_{40} \\ a_{50} + jb_{50} \\ a_{60} + jb_{60} \\ a_{70} + jb_{70} \end{bmatrix}$$

$$\Re\{Input(:,0)\} = \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{40} \\ a_{50} \\ a_{60} \\ a_{70} \end{bmatrix}$$

$$\Im\{Input(:,0)\} = \begin{bmatrix} b_{00} \\ b_{10} \\ b_{20} \\ b_{30} \\ b_{40} \\ b_{50} \\ b_{60} \\ b_{70} \end{bmatrix}$$

Vorüberlegungen zur Konstantenmultiplikation

Von jeder Spalte der Input-Matrix müssen vor der Addition, sowohl für den Real- als auch den Imaginärteil, das 2., 4., 6. sowie 8. Element mit $+\frac{\sqrt{2}}{2}$ multipliziert werden.
Das ergibt $4 \cdot 8 \cdot 2 = 64$ Berechnungen. Diese werden im 1. Takt durchgeführt und in einem Array abgespeichert.
Die Elemente werden später so arrangiert, dass nur die positive Konstantenmultiplikation von Nöten ist.

Array erstellen

1. Takt:

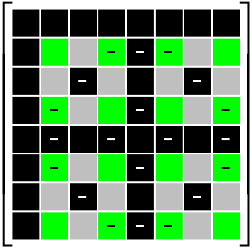
```
for i in 0 to 7 loop
    constMult_long(i*8)   := my_const*input_real(1)(i);
    constMult_long(i*8+1) := my_const*input_imag(1)(i);
    constMult_long(i*8+2) := my_const*input_real(3)(i);
    constMult_long(i*8+3) := my_const*input_imag(3)(i);
    constMult_long(i*8+4) := my_const*input_real(5)(i);
    constMult_long(i*8+5) := my_const*input_imag(5)(i);
    constMult_long(i*8+6) := my_const*input_real(7)(i);
    constMult_long(i*8+7) := my_const*input_imag(7)(i);
end loop;

for j in 0 to 63 loop
    constMult(j) <= constMult_long(j)(bit_width_extern -1 downto 0)
    ;
end loop;
```

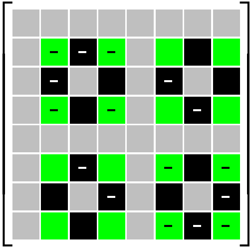
Die Elemente werden im weiteren Verlauf mit x_k bezeichnet.

Veranschaulichung

$\Re\{W\}$



$\Im\{W\}$



1. Zeile: 8 Additionen

3., 5. und 7. Zeile: immer 4 Additionen, 4 Subtraktionen
$\Rightarrow$ gleichviele Takte wie 1.

2., 4., 6. und 8. Zeile immer 6 Additionen, 6 Subtraktionen

Schrittweise Berechnung der 1. Zeile

Berechnung:

$$a_{k0} + a_{k1} + a_{k2} + a_{k3} + a_{k4} + a_{k5} + a_{k6} + a_{k7}$$

$$
\begin{array}{c c c}
\text{Takt} & & \text{Bit} \\
 & \underbrace{a_{k0} + a_{k1}} \qquad \underbrace{a_{k2} + a_{k3}} \qquad \underbrace{a_{k4} + a_{k5}} \qquad \underbrace{a_{k6} + a_{k7}} & 12 \\
2 & \Downarrow \qquad\qquad \Downarrow \qquad\qquad \Downarrow \qquad\qquad \Downarrow & \\
 & \underbrace{sum\_s1\_1 \quad + \quad sum\_s1\_2} \qquad \underbrace{sum\_s1\_3 \quad + \quad sum\_s1\_4} & 13 \\
3 & \Downarrow \qquad\qquad\qquad\qquad \Downarrow & \\
 & \underbrace{sum\_s2\_1 \qquad + \qquad sum\_s2\_2} & 14 \\
4 & \Downarrow & \\
 & sum\_s3\_1 & 15
\end{array}
$$

$\Rightarrow$ 3 Takte, 1. und 5. Leerlauf

Schrittweise Berechnung der 3., 5. und 7. Zeile

Beispiel: 3. Zeile der Twiddlefaktor-Matrix.

Berechnung:

$$a_{0k} - b_{1k} - a_{2k} + b_{3k} + a_{4k} - b_{5k} - a_{6k} + b_{7k}$$

Takt								Bit
	$a_{k0} - b_{k1}$		$b_{3k} - a_{2k}$		$a_{4k} - b_{5k}$		$b_{7k} - a_{6k}$	12
2	$\Downarrow$		$\Downarrow$		$\Downarrow$		$\Downarrow$	
	sum_s1_1	$+$	sum_s1_2		sum_s1_3	$+$	sum_s1_4	13
3		$\Downarrow$				$\Downarrow$		
		sum_s2_1		$+$		sum_s2_2		14
4				$\Downarrow$				
				sum_s3_1				15

$\Rightarrow$ 3 Takte, 1. und 5. Leerlauf

In der 2., 4., 6. und 8. Zeile müssen 12 Elemente addiert werden:

Beispiel: 2. Zeile der Twiddlefaktor-Matrix

$$a_{k0} + \frac{\sqrt{2}}{2} \cdot a_{k1} - \frac{\sqrt{2}}{2} \cdot b_{k1} + a_{k2} + \frac{\sqrt{2}}{2} \cdot a_{k3} - \frac{\sqrt{2}}{2} \cdot b_{k3} + a_{k4} + \frac{\sqrt{2}}{2} \cdot a_{k5} - \frac{\sqrt{2}}{2} \cdot b_{k5} + a_{k6} + \frac{\sqrt{2}}{2} \cdot a_{k7} - \frac{\sqrt{2}}{2} \cdot b_{k7}$$

$$= a_0 + x_0 - x_1 - b_2 + x_2 - x_3 + a_4 + x_4 - x_5 - b_6 + x_6 - x_7$$

Dadurch, dass bei dem Twiddlefaktor $\pm\frac{\sqrt{2}}{2} \pm \frac{\sqrt{2}}{2}$ sowohl Real- als auch Imaginärteil existieren, erhalten wir jeweils 2 Werte und somit insgesamt 4 mehr als bei den anderen Zeilen.

Schrittweise Berechnung der 2., 4., 6. und 8. Zeile

Berechnung (umsortiert):

$$a_0 - x_1 + x_0 - b_2 + x_2 - x_3 + a_4 - x_5 + x_4 - b_6 + x_6 - x_7$$

Takt		Bit
	$a_0 - x_1$ $x_0 - b_2$ $x_2 - x_3$ $a_4 - x_5$ $x_4 - b_6$ $x_6 - x_7$	12
2	$\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$	
	$s1\_1 + s1\_2$ $s1\_3 + s1\_4$ $s1\_5 + s1\_6$	13
3	$\Downarrow$ $\Downarrow$ $\Downarrow$	
	$s2\_1 + s2\_2$ $s2\_3$	14
4	$\Downarrow$ $\Downarrow$	
	$s3\_1$ $\Downarrow$	15
5	$\Downarrow$	
	$s4\_1$	16

5 Takte, 1. Multiplikationen, 2.-5. Additionen

Symmetrien der rein reellen 2D-DFT

```
A =

    1    5    3
    2    3    7
    4    2    9



>> real(fftshift(fft2(A)))
ans =

    -3.0000    -4.5000     1.5000
    -7.5000    36.0000    -7.5000
     1.5000    -4.5000    -3.0000

>> imag(fftshift(fft2(A)))
ans =

    5.19615   -2.59808   -7.79423
   -7.79423    0.00000    7.79423
    7.79423    2.59808   -5.19615
```

Symmetrien der rein reellen 2D-DFT

```
A =

   1   5   3
   2   3   7
   4   2   9
```

```
>> real(fftshift(fft2(A)))
ans =

   -3.0000   -4.5000    1.5000
   -7.5000   36.0000   -7.5000
    1.5000   -4.5000   -3.0000

>> imag(fftshift(fft2(A)))
ans =

    5.19615   -2.59808   -7.79423
   -7.79423    0.00000    7.79423
    7.79423    2.59808   -5.19615
```

```
B =

   2   2   2
   3   3   3
   4   4   4
```

```
>> real(fftshift(fft2(B)))
ans =

    0.50000    2.00000   -2.50000
    3.50000   38.00000    3.50000
   -2.50000    2.00000    0.50000

>> imag(fftshift(fft2(B)))
ans =

    0.86603    5.19615    9.52628
   -4.33013    0.00000    4.33013
   -9.52628   -5.19615   -0.86603
```

Asymmetrien der komplexen 2D-DFT

```
>> real(fftshift(fft2(A+i*B)))
ans =

   -3.86603   -9.69615   -8.02628
   -3.16987   36.00000  -11.83013
   11.02628    0.69615   -2.13397

>> imag(fftshift(fft2(A+i*B)))
ans =

    5.69615   -0.59808  -10.29423
   -4.29423   38.00000   11.29423
    5.29423    4.59808   -4.69615
```

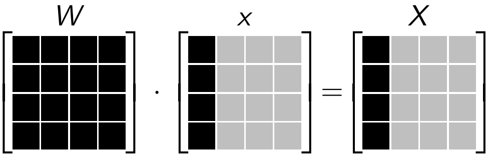
$\Rightarrow$ keine Symmetrien vorhanden!

Ansätze & weiteres Vorgehen

- Alle Spalten parallel berechnen
  - Alle Zeilen stehen (jeweils) gleichzeitig zur Berechnung der 2D-DFT bereit
  - Bei der jetzigen Implementierung werden auch alle Zeilen parallel berechnet
  - sehr viel Parallelität!
  - wenig Takte nötig, evtl. stünden mehr zur Verfügung
  - ungleiche Ausnutzung der Hardware je Takt

1D-DFT

Bisherige Implementierung:



5 Takte

Ideen, Gedanken & Fragen

- Unterschiedliche Bitbreiten bei den Zeilen 1, 3, 5, 7 und 2, 4, 6, 8!
  - 16tes Bit abschneiden?
  - mit wieviel Bit in die 2D-DFT hinein??
  - andere Breiten als 12 bedeuten in jeder Hinsicht mehr Aufwandt bezüglich der Wiederverwendbarkeit der 1D-DFT-Einheit
- Wieviele Bit sollen am Ende Übrig bleiben?
- Der Faktor $\frac{\sqrt{2}}{2}$ könnte rausgezogen werden
  - nur 1 statt 6 Multiplikationen je Zeile
  - erst alle Zahlen aufsummieren, dann die Multiplikation
-

Speicher

Wie wird die Speicherung sein?
Welche Art von Speicher verwenden wir?