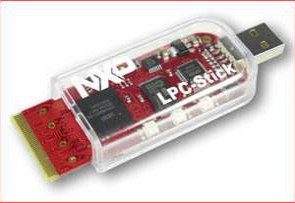


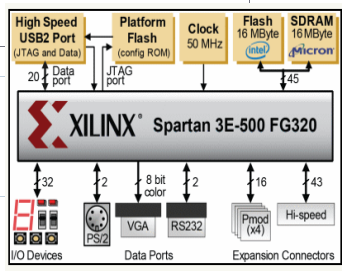
Informatik
HAW Hamburg



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Computer Engineering WS 2012

Digitale Systeme
PE



Prof. Dr. B. Schwarz

Informatik
HAW Hamburg

Digitale Systeme

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

3. Architektursynthese

ASM-Diagramme

Prozessorelement für eine S-Kurvenapproximation

Gemeinsame Nutzung von Arithmetik-Funktionseinheiten

(Ressource Sharing)

Gemeinsame Register- / Speicher-Nutzung

(Register Sharing)

VHDL-Modell des PE mit Fixed-Point Arithmetik im Q-Format

Datenpfad in Pipelinestruktur

Dynamische Systeme mit Rückführung

Multizyklusdatenpfad für ein Filter 1. Ordnung

Ergebnisübergabe an Folgezyklen



3.2 Entwurf eines Prozessorelementes

- In diesem Abschnitt werden die Entwurfsschritte der Architektursynthese am Beispiel eines Prozessorelementes vorgestellt:
Spezialprozessor zur S-Kurvenapproximation für eine Motoransteuerung.
- Am Beispiel dieses Spezialprozessors werden die folgenden Techniken dargestellt:
Multizyklus-Datenpfad für Datenraten \ll Systemfrequenz
Entwurf von ASM-Charts
Gemeinsame Nutzung von Hardware Funktionseinheiten (Ressource-Sharing)
Gemeinsame Register- / Speicher-Nutzung (Register-Sharing)

3

CE - DS 2



Entwurf eines Prozessorelementes

- Spezialprozessor
Operanden- und Ergebnispfad-Steering mit Multiplexer
Zustandsdiagramm für den Steuerpfad
- VHDL-Modellierung mit Integer-Arithmetik im Q-Format

4

CE - DS 2



Anwendungshintergrund (1)

- Zur Begrenzung der Anlaufströme für Antriebsmotore wird die Motorspannung nicht sprunghaft, sondern mit einer S-Funktion eingeschaltet.
- Diese S-Funktion kann mit einer cos-Funktion gebildet werden:

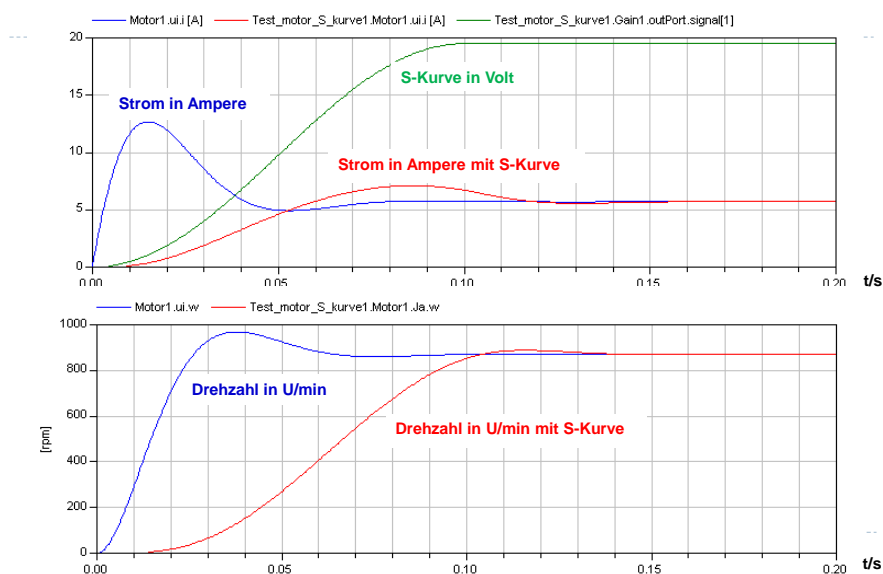
$$S(x) = (1 - \cos(x))/2$$
, wobei das Argument x mit einer Rampenfunktion zu erzeugen ist.
- Die in Antriebseinheiten zum Einsatz kommenden μ Controller können für zeitkritische Realtime-Anwendungen ggf. keine rechenintensiven Floatingpoint-Bibliotheken verwenden, sodass nur Integer-Arithmetik in Frage kommt.
- Eine S-Funktion ist also durch ein Polynom höherer Ordnung in x zu approximieren.

5

CE - DS 2



Anwendungshintergrund (2)





Entwurfsvorgaben (1)

- Die gleiche Vorgehensweise ist auch bei einer Realisierung von Antriebsteuerungen mit rekonfigurierbaren FPGA-ICs erforderlich.
- Da die Abtastrate für solche Antriebssteuerungen um Größenordnungen geringer ist als die verfügbare Taktfrequenz der ICs, kann mit einem **Multizyklus-Datenpfad** und **Resource-Sharing** gearbeitet werden.
- Für die HW-Realisierung der Arithmetik sind deshalb nur **ein Multiplizierer und ein Addierer** einzusetzen.
- Die Berechnung der S-Funktion erfolgt pro Wert $S(x)$ in einer Sequenz von Rechenschritten, wobei Zwischenwerte in Registern gespeichert werden (vgl. D.D. Gajski S. 336 – 353).

7

CE - DS 2



Entwurfsvorgaben (2)

- **Taylor-Reihenentwicklung** zur Approximation der Funktion $S(x) = (1 - \cos(x))/2$ um den Punkt $x = 0$.
- Nur die ersten beiden Terme mit geradem Exponenten werden berücksichtigt.
- Die approximierte Funktion kann für $x > 0$ nur im Bereich bis zum ersten positiven Maximum verwendet werden.
- Bestimmung der Koordinaten dieses Maximums: $S(x_0) \leq 1$. Mit Kenntnis dieses Maximums wird die Funktion normiert, damit das neue Maximum 1 wird.
- Die unabhängige Variable x der S-Funktion wird im Q7-Format mit einem Vorzeichenbit und 7 Nachkommabits repräsentiert :
 $-1 \leq x_Q < 1$.
- Substitution von **$x = x_0 x_Q$** in der S-Funktion, sodass sich eine Darstellung $S(x_Q)$ mit neuen Koeffizienten ergibt.

8

CE - DS 2



S-Kurven-Approximation

➤ Taylor-Reihenentwicklung

$$S(x) = 0.25 x^2 - x^4 / 48$$

➤ Maximum bei x_0 :

$$S(x_0 = \sqrt{6}) = \frac{3}{4} = 1/k$$

➤ Maximalwertnormierung $S(x_0 = \sqrt{6}) \Rightarrow 1!$

$$S(x) * k = \frac{1}{3} x^2 - \frac{1}{36} x^4$$

➤ Skalierung der x-Achse $x = x_0 x_Q$ mit $-1 \leq x_Q < 1$

$$S(x_Q) = 2 * (x_Q)^2 - (x_Q)^4$$

9

CE - DS 2



Struktur- und Freigabekonzept

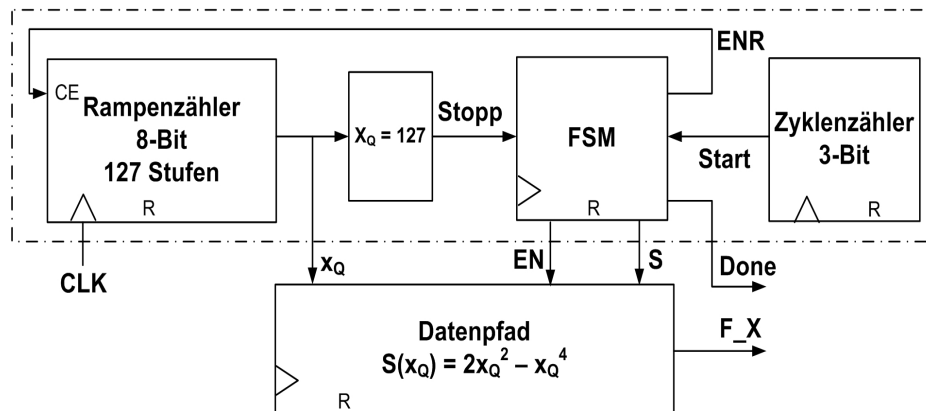
- Ein integrierte **Rampenzähler** ist ein 8 Bit-Zähler der die x_Q Eingangsgröße im Q7-Format bereitstellt.
- 128 Zählstufen sollen in 100 ms den Bereich 0 bis 1 liefern und danach angehalten werden.
- Ein **3 Bit-Zyklenzähler** wird hier vorgeschlagen, der zusammen mit der **FSM (Control Unit)** als eine Art **Frequenzteiler** für den Rampenzähler wirkt.
- Die FSM startet zusammen mit diesem Zyklenzähler und wartet nach Rückkehr in den Idle-Zustand solange, bis der Zyklenzähler seinen wrap-around ausgeführt hat.
- Mit jedem wrap-around des Zyklenzählers erhält der Rampenzähler die Freigabe für das Inkrement einer Rampenstufe.
- Dieses Freigabekonzept für die Komponenten, die alle mit der gleichen Taktfrequenzbetrieben werden, ist ein typischer Ansatz in so genannten **Multiraten-Systemen**.

10

CE - DS 2



Systemstruktur mit unterschiedlichen Taktbereichen



11

CE - DS 2



Zeitskalierung

- **8-Bit Rampenzähler mit 128 Stufen: $x_{Qmax} = 127$**
- **Anlaufintervall 100ms**
- **Effektive Taktfrequenz des Rampenzählers:**
 $f_{Rclk} = 1/(100ms/128) = 1280 \text{ Hz}$
- **3-Bit Zyklenzähler:**
8 Zählzustände müssen das Intervall $1/f_{Rclk}$ abdecken
 $f_{clk} = 8 * 1280 \text{ Hz} = 10,24 \text{ kHz}$
Taktfrequenz der Systemkomponenten

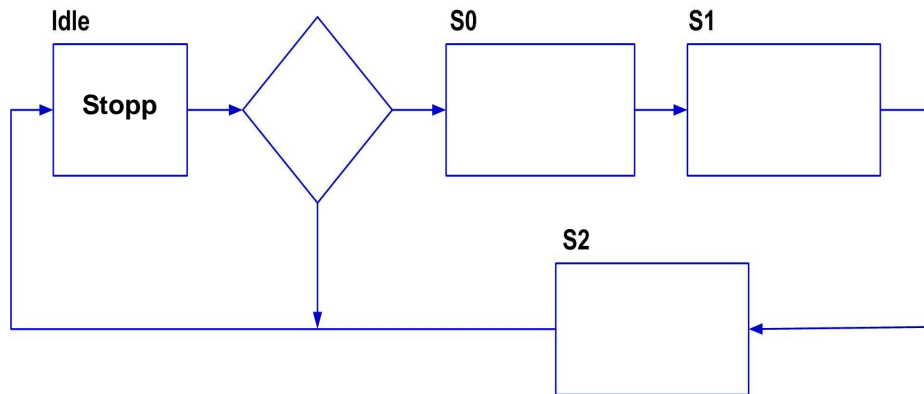
12

CE - DS 2



ASM des PE zur Approximation der S-Kurve

➤ Zu realisierende Funktion: $S(x_Q) = 2 \cdot (x_Q)^2 - (x_Q)^4$

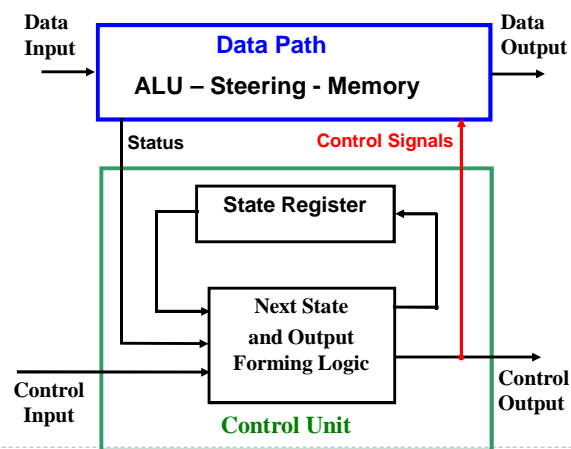


13

CE - DS 2



Strukturierung eines digitalen Systems als Prozessorelement in einen Datenpfad und einen Steuerpfad



14

CE - DS 2



Gemeinsame Register-Nutzung (Register Sharing)

- Nicht alle Zwischensignale müssen in einem eigenen Register abgelegt werden. Abhängig von der Nutzungsdauer können verschiedene Zwischensignale in einem Register abgelegt werden.
- Vom Zustandsautomaten wird festgelegt, welche Operationen in den verschiedenen Takten mit den Registerwerten durchgeführt werden sollen.
- In einer Tabelle (signal lifetime table) wird die Lebensdauer der Zwischensignalwerte dargestellt. Dabei werden alle die Zustände gekennzeichnet, in denen das Zwischensignal noch benötigt wird. Begonnen wird mit dem Zustand, der auf die Abspeicherung im Zwischenregister folgt.

15

CE - DS 2



Signal-Lebensdauertabelle

	Idle	S0	S ₁	S ₂	Reg
t1					
t2					
y					

Es ist zu berücksichtigen, dass das Ergebnis $y = S(xq)$ während eines kompletten Zyklus (8 Takte) am Ausgang zur Verfügung steht.

16

CE - DS 2



➤ Der Datenpfad ist in mehrere Funktionsebenen gegliedert:

1. Die **parallelen Register** nehmen die Zwischenergebnisse auf, sodass alte Werte ohne weitere Verwendung überschrieben werden können. Ggf. ist ein Multiplexer vorzuschalten, falls zu Beginn eines Zyklus die Eingangsgrößen eingespeichert werden müssen.
2. Die **Operanden** werden den Arithmetikfunktionen über **Multiplexer** zugeführt. Diese werden erforderliche, wenn eine Operation mehrmals verwendet wird und die Operanden aus unterschiedlichen Registern bzw. von extern zugeführt werden.
3. Die **AU-Elemente** stehen parallel zur Verfügung und verarbeiten die Operanden in einer Sequenz. Nur wenn der zu implementierende Algorithmus es zulässt, können die AU-Elemente auch parallel genutzt werden.
4. Eine weitere Multiplexer-Ebene verteilt die Rechenergebnisse auf die Register. Wenn bei der Signal-Register-Zuordnung auch die Operationen berücksichtigt werden, lassen sich hier Multiplexer einsparen.

17

CE - DS 2



Datenpfad der S-Kurvenapproximation

18

CE - DS 2



Integerarithmetik im Q-Format mit Signaltyp signed

- Signale und Variablen in digitalen Systemen auf Basis von FPGAs werden mit VHDL auf Bitvektorebene modelliert.
- Für die unsigned and signed Vektordatentypen sind die arithmetischen Operatoren +, - und * sowie Vergleichsoperatoren in den Bibliotheken ieee.numeric_std und ieee.std_logic_arith definiert.
- Damit stehen Integerzahlen und rein gebrochene Zahlendarstellungen, d.h. die Fractional-Zahlen, sowie deren Kombination für die 2er-Komplementarithmetik zur Verfügung.
- Für Fractional-Zahlen im Q-Format gilt ein Bitstring mit der Form:

$$x_{\text{Bin}} = b_B \cdot b_{B-1} \dots b_1 b_0 \quad \text{mit } b \in (0,1) \text{ und}$$

einem Vorzeichenbit $b_B = 1$ für $x_{\text{Dez}} < 0$.

Dieses QB-Format repräsentiert Zahlen im Intervall $-1 \leq x_{\text{Dez}} < 1$

19

CE - DS 2



Q-Format

- Bewährt hat sich die **Fractional-Darstellung** im Q-Format für die Eingangs- und Ausgangssignale, Koeffizienten sowie Rechengrößen in digitalen Systemen, da Multiplikationsergebnisse im Bereich $-1 \leq x_{\text{Dez}} < 1$ **begrenzt** bleiben und betragsmäßig in Richtung des LSBs 2^{-B} für $b_0 = 1$ streben.
- Die QB-Notation angewandt auf Vektordatentypen sagt aus, dass die Binärdarstellung B Bits rechts des impliziten Binärpunktes enthält.
- Dies ist hier in einer beispielhaften Schreibweise dargestellt:

QB: sign bit · B significant bits = sign · msb ... lsb

z.B. für das Signal $x[\text{sign. } B-1 : 0]$

- Ein B+1=5-Bit Koeffizient im Q4-Format hat z.B. folgende Deklaration:
`constant C0: signed(4 downto 0) := 01011; -- 0.6875 = 11/16`

20

CE - DS 2



Addition mit vorzeichenrichtiger Erweiterung der Operanden

- Eine Addition zweier QB-Größen A und C, die jeweils den positiven Maximalwert $1-2^{-B}$ annehmen können, liefert maximal den Wert $2-2^{-B+1}$, der aufgrund des Integeranteils nicht im QB-Format allein darstellbar ist.
- Der Ergebnisvektor SUM ist also um eine Bitstelle (Guard-Bit) breiter zu wählen, die den Integeranteil (Übertrag) aufnimmt.
- Da Signal- und Variablenzuweisungen auf beiden Seiten der Zuweisung den gleichen Datentyp und die gleiche Vektorbreite aufweisen müssen, sind die Summanden A und C zusätzlich mit einer vorzeichenrichtigen Erweiterung an die Breite des Summenvektors SUM anzupassen.

```
signal A, C: signed(7 downto 0); -- inputs
signal SUM: signed(8 downto 0); -- result
begin
SUM <= (A(A'left) & A) + (C(C'left) & C) after 5 ns;
```

21

CE - DS 2



Binäre Multiplikation (1)

- Auch bei der Multiplikation ist die Vektorbreite des Ergebnisses auf die der Faktoren anzupassen.
- Eine Multiplikation mit einem QB₁-Multiplikanden und einem QB₂-Multiplikator ergibt ein QB₃-Ergebnis. Darin stehen $B_3=B_2+B_1$ Bits rechts vom impliziten Binärpunkt und der gesamte Vektor enthält B_3+2 Bits.
- Eine **Q-Format-Multiplikation** liefert nämlich **zwei Vorzeichenbits**, von denen das linke Bit ein „echtes“ Vorzeichen ist und das rechte Bit vor dem Binärpunkt als **Guard-Bit** genutzt werden kann.

22

CE - DS 2



Binäre Multiplikation (2)

```
constant COEFF: signed(7 downto 0) := x"7f";
                                -- 127/128 multiplier Q7
signal STAGE:   signed(11 downto 0); -- multiplicand Q11
signal MUL:     signed(19 downto 0); -- result Q18
begin
MUL <= STAGE * COEFF after 6 ns;
```

Symbolische Schreibweise:

$MUL[sign, sign_{17:0}] = MUL[sign, guard_{17:0}] =$

$STAGE[sign_{10:0}] * COEF[sign_{6:0}]$

23

CE - DS 2



Binäre Multiplikation als Summe von Teilprodukten

Multiplikand * Multiplizierer

-0.875 * (-0.6875)
 04375
61250
 065625
700000
 0765625
5250000
 0.6015625

1.0010 * 1.0101

24

CE - DS 2



Entwurf von Integer-Arithmetik mit dem Q-Format (1)

➤ Strategieübersicht

- **1. Ansatz:** Auslegung des Datenpfades auf **z.B. 10 Bit** breite Signalpfade. Als Ausgangspunkt der Dimensionierung werden die Register als Quellen der Operanden mit 10 Bit Vektoren deklariert.
- Die für die Arithmetik-Operationen erforderlichen Modifikationen der Operanden und der Ergebnisse sind für die Vektorauslegung zu planen.
- Die Simulationsergebnisse des VHDL-Modells zeigen auf, ob die Vektorbreiten zur Genauigkeitserhöhung (kleineres LSB 2^{-B} , $B \uparrow$) anzupassen sind.
- Zyklus jeweils mit Wiederholung der Simulationsauswertung.

25

CE - DS 2



Entwurf von Integer-Arithmetik mit dem Q-Format (2)

- **2. Ansatz:** Die Eingangssignale bzw. Operanden eines **Matlab Floating-Point-Modells** werden mit den **Funktionen zur Fixed-Point-Arithmetik** auf einen begrenzten Darstellungsbereich reduziert.
- Aus einer Ergebnisbewertung heraus wird auf die geeignete Vektordimensionierung des Q-Formats mit Nachkommastellen und Guard-Bits geschlossen.
- Ein Beispiel hierzu liefert die Vorabanalyse der Wurzel-Approximation mit dem Matlab-Code in der Laborbeschreibung CEP 3.

26

CE - DS 2



Entwurf von Integer-Arithmetik mit dem Q-Format (3)

- **3. Ansatz: MDA** (Modell getriebene Architektur) mit dem VHDL-Codegenerator **Systemgenerator**. RTL-Modellierung von PEs mit einer Matlab-Simulink basierten grafischen Oberfläche.
- Dimensionierung der Datenpfad-Funktionselemente mit Full-Precision (32 Bit Q-Format-Vektoren).
- Sukzessive Verfeinerung durch Vektorbreitenreduzierung und Überprüfung der Ergebnisgenauigkeit abgestimmt mit dem FPGA-Ressourcen-Bedarf.
- Ausgehend vom Simulationsendergebnis Übergang auf die VHDL-Simulation und die FPGA-Implementierung.
- Beispiele:
 - D. Mellert : Modellierung einer Video-basierten Fahrspurerkennung mit dem Xilinx Systemgenerator für eine SOC-Plattform. BA HAW Hamburg, Department I; 04.2010
 - D. Brandmeier: FPGA-Implementierung einer feldorientierten Regelung zur Leistungsfaktorkorrektur. Diplomarbeit HAW Hamburg, Department I/E 2006; ESW Wedel

27

CE - DS 2



Entwurf von Integer-Arithmetik mit dem Q-Format (4)

- **4. Ansatz: Mathematische Analyse** von linearen Systemen mit Rückkopplungen (IIR-Filter, Differentialgleichungen) durch komplexe **Teilübertragungsfunktionen $G(j\omega)$** für innere Summationspunkte.
- Ausgangsskalierung und zusätzliche Dimensionierung des Q-Formats für Rückkopplungsaddierer mit Guard-Bit Erweiterung, die die durch Verstärkungsüberhöhungen verursachten Überschwinger überlauffrei aufnehmen.
- [5] Kapitel 9.2 IIR-Filter

28

CE - DS 2



Entwurf von Integer-Arithmetik (5)

5. Ansatz: Mobile-Kommunikation

Modellierung der Sprachcodierung in drei Schritten:

Floating-Point Realsierung

Reduzierung des C/C++ Codes für Festkomma-DSPs

Akustischer Vergleich der Sprachqualität und Entscheidung für Fixed-Point Genauigkeit.

D. Bonkougou: Bewertungskonzept und Berechnung von Festkomma-codebüchern für eine Festkommaimplementierung eines Sprachcodieralgorithmus. Diplomarbeit FH-Hamburg FB E/I 2002; Ericson Nürnberg

29

CE - DS 2



Auslegung des Q-Formats für die S-Kurve

➤ Verfahren nach Ansatz 1.:

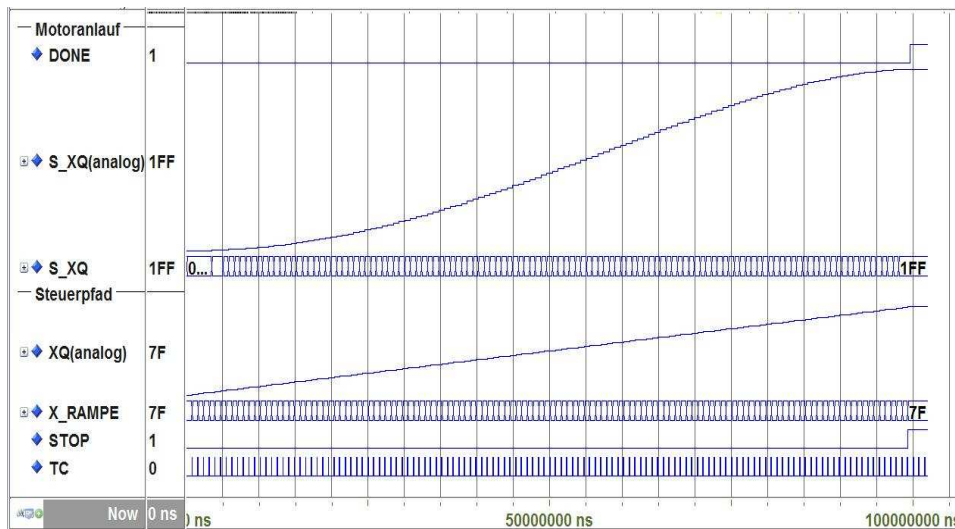
- Rampenvektor x_Q : **Erweiterung** auf der LSB-Seite mit 2 Bit "00" und Kompensation des SHL durch Übergang von Q7 auf Q9.
- **Abgriff des Multiplikationsergebnisses** Q18 mit 2 VZ: RESULT(18 : 9); abschneiden der unteren 9 Bit ohne Rundung.
- **Erweiterung der Subtraktionsoperanden**, sodass Faktor 2 für x^2 durch SHL erreicht wird und Sign-Extension bei x^4 , damit die Vektorlängen auf WIDTH+1 angeglichen werden.
- Hier **Reduzierung des Subtraktionsergebnisses** auf 10 Bit gezeigt, da $S(x_Q) < 1$ angenommen wird.
Mit einem 11 Bit Register R_3 für $S(x_Q)$ wäre **mehr Sicherheit** gegen einen Überlauf im Fall $S(x_Q) > 1$ gegeben.

30

CE - DS 2



VHDL-Simulation der S-Kurve



31

CE - DS 2



Modell des Datenpfades (1)

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed.all;

entity DATENPFAD is -- Multizyklus-DP
generic ( WIDTH : natural := 13); -- Vektorbreite im Datenpfad
port (CLK, RESET_N : in bit; -- LOW ACTIVE für XC3S400-Board 8. Stock
      SEL : in bit; -- Multiplexer Select S aus FSM
      R1_EN, R2_EN, R3_EN : in bit; -- Registerfreigabe EN aus FSM
      X : in bit_vector(7 downto 0); -- Rampengenerator Xq
      F_X : out std_logic_vector(9 downto 0) -- S-Kurve
    );
end DATENPFAD;

-----INTERNE SIGNALE-----
architecture VERHALTEN of DATENPFAD is
    -- Register Power-Up
    signal REG1, REG2, REG3 : std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal MUL : std_logic_vector(WIDTH-1 downto 0); -- Arithmetik-Prozess
    signal SUB : std_logic_vector(WIDTH-1 downto 0);
    signal X_CALC : std_logic_vector(WIDTH-1 downto 0); -- shift left u. Q7 -> Q(WIDTH-1)
    signal ZEROS : std_logic_vector(WIDTH-1 - 7 - 1 downto 0); -- XQ Ergänzung

```

32

CE - DS 2



Modell des Datenpfades (2)

```

begin
    -----REGISTER-----
    _REGISTER1:process(CLK)
    begin
        if (CLK = '1' and CLK'event) then
            if (RESET_N = '0') then
                REG1 <= (others => '0') after 5 ns;
            elsif (R1_EN = '1') then
                REG1 <= MUL after 5 ns; -- t1 = xx
            end if;
        end if;
    end process _REGISTER1;
    _REGISTER2:process(CLK)
    begin
        if (CLK = '1' and CLK'event) then
            if (RESET_N = '0') then
                REG2 <= (others => '0') after 5 ns;
            elsif (R2_EN = '1') then
                REG2 <= MUL after 5 ns; -- t2 = xx*xx
            end if;
        end if;
    end process _REGISTER2;
    _REGISTER3:process(CLK)
    begin
        if (CLK = '1' and CLK'event) then
            if (RESET_N = '0') then
                REG3 <= (others => '0') after 5 ns;
            elsif (R3_EN = '1') then
                REG3 <= SUB after 5 ns; -- t3 = y = 2xx - xxxx
            end if;
        end if;
    end process _REGISTER3;
end
33

```

CE - DS 2



Modell des Datenpfades (3)

```

ZEROS <= (others => '0');
X_CALC <= TO_stdlogicvector(X) & ZEROS; -- Rampe shift left u. Q7 -> Q(width-1)
-----ALU's-----
MULTIPLIER : process (SEL, X_CALC, REG1)
    variable TMP : std_logic_vector(WIDTH-1 downto 0); -- selektierte Operanden
    variable RESULT : std_logic_vector(2*WIDTH - 1 downto 0); -- doppelte Vektorbreite
    begin
        if (SEL = '0') then
            TMP := X_CALC;
        else
            TMP := REG1; -- (SEL = '1')
        end if;
        RESULT := TMP * TMP; -- Q(width-1)*Q(width-1) = sign,sign.msb ... lsb; 2*width bit
        MUL <= RESULT ((2*WIDTH - 2) downto WIDTH-1); -- t1 ; t2 width bit: sign.msb ... lsb
    end process MULTIPLIER;

SUBTRACTOR : process (REG1, REG2)
    variable RESULT : std_logic_vector(WIDTH downto 0); -- Width +1 bit
    begin
        -- shift left sign extension ; bleibt Q(WIDTH-1)
        RESULT := (REG1 & '0') - (REG2(REG2'left) & REG2); -- sign,guard.msb .. lsb
        SUB <= RESULT (WIDTH -1 downto 0); -- y
    end process SUBTRACTOR;
F_X <= REG3(WIDTH-1 downto WIDTH -10); -- S-Kurve y immer 10 Bit Ausgang
end VERHALTEN;
34

```

CE - DS 2



Modell des Steuerpfades (1)

```
-- FSM-Timer-Konzept
entity STEUERPFAD is -- FSM erweitert mit Rampenzähler und Zyklenzähler
port(
    PB: in bit; -- externer Startschalter
    RESET_N: in bit;
    CLK: in bit;
    DONE: out bit; -- Endwert erreicht
    R1_EN, R2_EN, R3_EN: out bit; -- EN Register Freigaben
    SEL: out bit; -- S MUX Select
    X: out bit_vector(7 downto 0) -- XQ Rampe
);
end STEUERPFAD;

architecture VERHALTEN of STEUERPFAD is
-- interne signale -- Zyklenzähler- und Rampen-Power-Up
    signal COUNTER: std_logic_vector(2 downto 0) := (others => '0');
    signal X_RAMPE: std_logic_vector(7 downto 0) := (others => '0');

    type STATES is (IDLE, S0, S1, S2);
    signal ZUSTAND, FOLGE_Z: STATES := IDLE;
    signal TC: bit; -- Zyklenzähler Terminal Count
    signal STOP: bit; -- Rampenendwert erreicht
end architecture VERHALTEN;
```

35

CE - DS 2



Modell des Steuerpfades (2)

```
ZYKLENSZAEHLER: process(CLK) -- Periodische wrap-arounds: Sägezahn
begin
    if CLK'event and CLK = '1' then
        if RESET_N = '0' then
            COUNTER <= (others => '0') after 5 ns;
        elsif PB = '1' then -- externe Freigabe wie bei FSM
            COUNTER <= COUNTER + 1 after 5 ns;
        end if;
    end if;
end process ZYKLENSZAEHLER;
TC <= '1' after 5 ns when COUNTER = 7 else '0' after 5 ns; -- Einzelpuls
RAMPENGGENERATOR: process(CLK)
begin
    if CLK'event and CLK = '1' then
        if RESET_N = '0' then
            X_RAMPE <= (others => '0') after 5 ns;
        elsif TC = '1' and STOP = '0' then -- Selbsthaltung des Rampenendwertes
            X_RAMPE <= X_RAMPE + 1 after 5 ns;
        end if;
    end if;
end process RAMPENGGENERATOR;
STOP <= '1' after 5 ns when X_RAMPE = 127 else '0' after 5 ns; -- 1 = Dauerpegel
Z_SPEICHER: process(CLK)
begin
    if (CLK = '1' and CLK'event) then
        if RESET_N = '0' then
            ZUSTAND <= IDLE after 5 ns;
        elsif PB = '1' then -- externer Start (CE)
            ZUSTAND <= FOLGE_Z after 5 ns;
        end if;
    end if;
end process Z_SPEICHER;
```

36

CE - DS 2



Modell des Steuerpfades (3)

```


UE_A_SN: process(ZUSTAND, TC, STOP)
begin
    -- Defaults
    SEL <= '0' after 5 ns; -- Rampe an MUL
    R1_EN <= '0' after 5 ns; -- kein Register Update
    R2_EN <= '0' after 5 ns;
    R3_EN <= '0' after 5 ns;
    FOLGE_Z <= IDLE after 5 ns; -- keine Transition
    DONE <= '0' after 5 ns; -- kein Endergebnis S(xq=127)
    case ZUSTAND is
        -- MOORE FSM
        when IDLE =>
            DONE <= STOP after 5 ns; -- Anzeige des S(xq=127) Endwertes
            if TC='1' and STOP = '0' then -- 1. Rechnung mit erster Rampenstufe
                FOLGE_Z <= S0 after 5 ns; -- Ende mit Rampenstopp
            end if;
        when S0 =>
            R1_EN <= '1' after 5 ns; -- t1 = xx speichern
            FOLGE_Z <= S1 after 5 ns;
        when S1 =>
            SEL <= '1' after 5 ns; -- R1 an MUL
            R2_EN <= '1' after 5 ns; -- t2 = xx*xx speichern
            FOLGE_Z <= S2 after 5 ns;
        when S2 =>
            R3_EN <= '1' after 5 ns; -- Subtraktion speichern
        end case;
    end process UE_A_SN;
    X <= To_BitVector(X_RAMPE) after 5 ns;
end VERHALTEN;
37

```




FSM-Timer Konzept

- Dieses Konzept zur Kopplung eines Datenpfades mit dem Steuerpfad ist aufgrund folgender Aspekte zu empfehlen:
 - Sofern in Anwendungen ein Timer-Grenzwert in mehreren Zuständen geprüft werden muss, ist es sinnvoll, den Komparator nicht mehrfach in der FSM zu realisieren. Der in der Timer-Entity platzierte Komparator spart damit Logik-Ressourcen ein.
 - Zähler, die große Speicherbereiche adressieren, wie z.B. Bildzwischen-speicher mit >1 MB in Farbbildverarbeitungsanwendungen, sind ggf. mehr als 20 Bit breit. Ein Austausch von einzelnen Statusleitungen mit der FSM anstelle der Einkopplung aller Zählerleitungen in das kombinierte Übergangs- und Ausgangsschaltnetz kann die Verdrahtungsressourcen reduzieren. Da das FPGA-Routing in den Signallaufzeitpfaden je nach Anwendung bis zu 50 % ausmacht, kann diese vorgeschlagene Funktionspartitionierung auch die Timingkennwerte verbessern.



Informatik
HAW Hamburg

Top-Entity (1)



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

```

-- CE DS 2 im SS 10 SWR
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed.all;

entity MOTORANLAUF is -- S-Kurven-Approximation Top-Entity
  port (
    PB      : in bit;
    RESET_N : in bit;
    CLK     : in bit;
    DONE    : out bit; -- Maximalwert S(XQ=1) = 1 erreicht
    S_XQ    : out std_logic_vector(9 downto 0) -- S-Kurve
  );
end MOTORANLAUF;
architecture VERHALTEN of MOTORANLAUF is


  signal R1_CE, R2_CE, R3_CE, S: bit; -- portmap Kopplung
  signal XQ: bit_vector(7 downto 0); -- Rampe

  component STEUERPFAD -- FSM erweitert mit Rampenzähler und Zyklenzähler
  port (
    PB      : in bit; -- Externer Start
    RESET_N : in bit;
    CLK     : in bit;
    DONE    : out bit; -- Endwert erreicht
    R1_EN, R2_EN, R3_EN: out bit; -- Register-Freigabe
    SEL     : out bit; -- MUX Select für MUL Operanden
    X       : out bit_vector(7 downto 0) -- Rampe
  );
end component;

```


CE - DS 2

39



Informatik
HAW Hamburg

Top-Entity (2)



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

```

component DATENPFAD -- Multizyklus DP
  port (CLK, RESET_N : in bit; -- RESET LOW ACTIVE
        SEL          : in bit; -- MUX Select für MUL Operanden
        R1_EN, R2_EN, R3_EN : in bit; -- Register-Freigabe
        X              : in bit_vector(7 downto 0); -- Rampe
        F_X           : out std_logic_vector(9 downto 0) -- S-Kurve
  );
end component;

begin
  ST_I: STEUERPFAD -- Instanziierung
  port map (PB => PB, RESET_N => RESET_N, CLK => CLK, DONE => DONE,
            R1_EN => R1_CE, R2_EN => R2_CE, R3_EN => R3_CE,
            SEL => S, X => XQ); -- formal => actual

  DP_I: DATENPFAD
  port map (
    CLK => CLK, RESET_N => RESET_N, SEL => S,
    R1_EN => R1_CE, R2_EN => R2_CE, R3_EN => R3_CE,
    X => XQ, F_X => S_XQ);
end VERHALTEN;

```

CE - DS 2

40



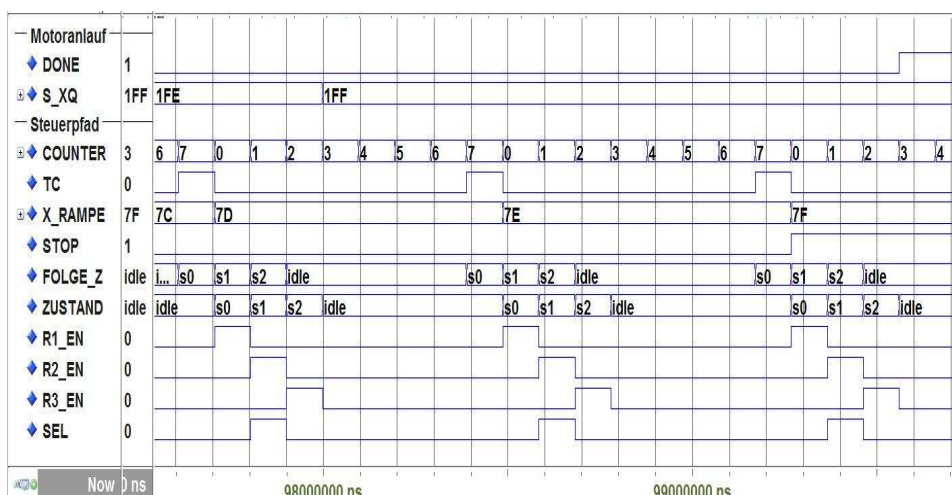
Testbench (1)

```

entity TB is
generic (PERIODE10_24KHZ_HALBE: time := 48828125 ps);
-- Frequenz fuer Anlaufdauer 100 ms
-- 100 ms / 8 / 128 = Periodendauer -> f = 1/T = 10,24 kHz
end TB;
architecture VERHALTEN of TB is
signal PB, CLK, RESET_N : bit;
component MOTORANLAUF
port( PB, RESET_N, CLK : in bit;
      DONE          : out bit;
      S_XQ          : out std_logic_vector(9 downto 0) );
end component;
begin
DUT: MOTORANLAUF
port map (PB => PB, CLK => CLK, RESET_N => RESET_N, S_XQ => open, DONE => open);
TESTBENCH_CLK: process
begin
  CLK <= '0'; wait for PERIODE10_24KHZ_HALBE;
  CLK <= '1'; wait for PERIODE10_24KHZ_HALBE;
end process TESTBENCH_CLK;
TESTBENCH_RESET: process
begin
  RESET_N <= '0'; wait for (2*PERIODE10_24KHZ_HALBE);
  RESET_N <= '1'; wait;
end process TESTBENCH_RESET;
TESTBENCH_PB: process
begin
  PB <= '0'; wait for (2*PERIODE10_24KHZ_HALBE);
  PB <= '1'; wait;
end process TESTBENCH_PB;
end VERHALTEN;
  
```



VHDL-Simulation der S-Kurve





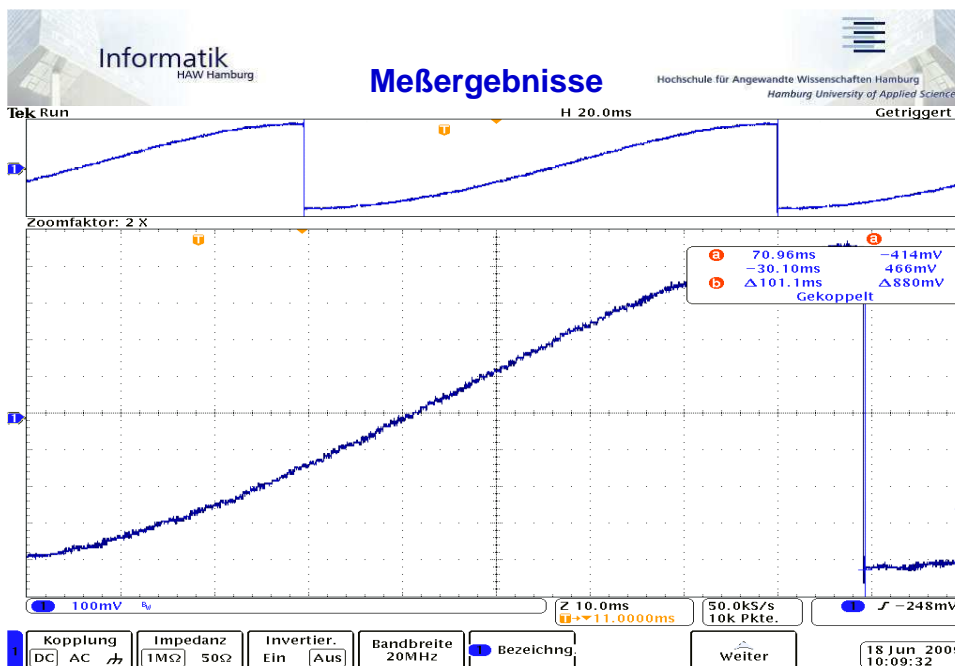
Auszug aus .do File für die Signalauswahl

```
# Fuer S-Kurvenapproximation
restart
view wave
radix hex
# Displays all signals in waves window:
#add wave -noupdate -divider PushButton
#add wave -height 30 -label PB sim:/tb/dut/pb
#add wave -noupdate -divider "Clock & Reset(LowActive)"
#add wave -height 30 -label CLK sim:/tb/dut/clk
add wave -noupdate -divider "Motoranlauf"
add wave -height 30 -label DONE sim:/tb/dut/done
add wave -height 200 -analog-step -min 0 -max 512 -label "S_XQ(analog)"
sim:/tb/dut/s_xq

add wave -height 30 -label COUNTER sim:/tb/dut/st_i/counter
add wave -noupdate -divider "Datenpfad"
add wave -height 30 -label REG3 sim:/tb/dut/dp_i/reg3
add wave -height 30 -label MUL sim:/tb/dut/dp_i/mul
# geforderte Anlaufzeit
run 102 ms
```

43

CE - DS 2



44

CE - DS 2



Längster Laufzeitpfad (1)

Delay (setup path): 10.496 ns (data path - clock path skew + uncertainty)

Source: ST_I/X_RAMPE_7 (FF)
 Destination: DP_I/REG1_5 (FF)
 Data Path Delay: 10.496ns (Levels of Logic = 2)
 Clock Path Skew: 0.000ns
 Source Clock: CLK_BUFGRP rising
 Destination Clock: CLK_BUFGRP rising
 Clock Uncertainty: 0.000ns

Maximum Data Path: ST_I/X_RAMPE_7 to DP_I/REG1_5

Location	Delay type	Delay(ns)	Physical Resource Logical Resource(s)
SLICE_X13Y107.YQ	Tcko	0.511	ST_I/X_RAMPE<6> ST_I/X_RAMPE_7
SLICE_X13Y96.G4	net (fanout=3)	0.772	ST_I/X_RAMPE<7>
SLICE_X13Y96.Y	Tilo	0.612	DP_I/TMP_mux0000<0> DP_I/TMP_mux0000<12>1
MULT18X18_X0Y12.B15	net (fanout=12)	2.499	DP_I/TMP_mux0000<12>
MULT18X18_X0Y12.P17	Tmult	4.331	DP_I/Mmult_RESULT_mult0000 DP_I/Mmult_RESULT_mult0000
SLICE_X2Y97.BX	net (fanout=2)	1.458	DP_I/MUL<5>
_SLICE_X2Y97.CLK	Tdick	0.313	DP_I/REG1<5> DP_I/REG1_5

Total 10.496ns (5.767ns logic, 4.729ns route)
 (54.9% logic, 45.1% route)

45

CE - DS 2



Spartan 3E Switching Characteristics (1)

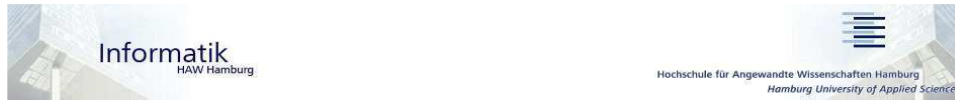
Configurable Logic Block (CLB) Timing

Table 98: CLB (SLICEM) Timing

Symbol	Description	Speed Grade				Units
		-5		-4		
		Min	Max	Min	Max	
Clock-to-Output Times						
T _{CKO}	When reading from the FFX (FFY) Flip-Flop, the time from the active transition at the CLK input to data appearing at the XQ (YQ) output	-	0.52	-	0.60	ns
Setup Times						
T _{AS}	Time from the setup of data at the F or G input to the active transition at the CLK input of the CLB	0.46	-	0.52	-	ns
T _{DICK}	Time from the setup of data at the BX or BY input to the active transition at the CLK input of the CLB	1.58	-	1.81	-	ns
Hold Times						
T _{AH}	Time from the active transition at the CLK input to the point where data is last held at the F or G input	0	-	0	-	ns
T _{CKDI}	Time from the active transition at the CLK input to the point where data is last held at the BX or BY input	0	-	0	-	ns

46

CE - DS 2



Spartan 3E Switching Characteristics (2)

Propagation Times						
T_{ILO}	The time it takes for data to travel from the CLB's F (G) input to the X (Y) output	-	0.66	-	0.76	ns

18 x 18 Embedded Multiplier Timing

Table 102: 18 x 18 Embedded Multiplier Timing

Symbol	Description	Speed Grade				Units
		-5		-4		
		Min	Max	Min	Max	
Combinatorial Delay						
T _{MULT}	Combinatorial multiplier propagation delay from the A and B inputs to the P outputs, assuming 18-bit inputs and a 36-bit product (AREG, BREG, and PREG registers unused)	-	4.34 ⁽¹⁾	-	4.88 ⁽¹⁾	ns

47

CE - DS 2



Längster Laufzeitpfad (2)

Delay (setup path): 10.332ns (data path - clock path skew+uncertainty)

Source: ST_I/ZUSTAND_FSM_FFd2_1 (FF)

Destination: DP_I/REG1_5 (FF)

Data Path Delay: 10.332ns (Levels of Logic = 2)

Clock Path Skew: 0.000ns

Source Clock: CLK_BUFPG rising

Destination Clock: CLK_BUFPG rising

Clock Uncertainty: 0.000ns

Maximum Data Path: ST_I/ZUSTAND_FSM_FFd2_1 to DP_I/REG1_5

Location	Delay type	Delay(ns)	Physical Resource Logical Resource(s)
SLICE_X13Y103.YQ	T_{cko}	0.511	ST_I/ZUSTAND_FSM_FFd2_1
SLICE_X13Y96.G3	net (fanout=13)	0.608	ST_I/ZUSTAND_FSM_FFd2_1
SLICE_X13Y96.Y	T_{ilo}	0.612	ST_I/ZUSTAND_FSM_FFd2_1
MULT18X18_X0Y12.A17	net (fanout=12)	2.499	DP_I/TMP_mux0000<0>
MULT18X18_X0Y12.P17	T_{mult}	4.331	DP_I/TMP_mux0000<12>1
SLICE_X2Y97.BX	net (fanout=2)	1.458	DP_I/TMP_mux0000<12>
SLICE_X2Y97.CLK	T_{dick}	0.313	DP_I/Mmult_RESULT_mult0000
			DP_I/Mmult_RESULT_mult0000
			DP_I/MUL<5>
			DP_I/REG1<5>
			DP_I/REG1_5

Total 10.332ns (5.767ns logic, 4.565ns route)
(55.8% logic, 44.2% route)

48

CE - DS 2