5. Übung	**M9 VLSI Anwendungen** Hochschule für Technik und Wirtschaft	SS 2021

Ausgabe: 17./24. Juni 2021 Abgabe: 1./8. Juli 2021

Schwerpunkte

- Arithmetische Funktionseinheiten
- Synchrones Design
- CORDIC Algorithmus

CORDIC Teil II

Die Realisierung des CORDIC-Elementes enhthält neben dem Datenpfad bestehend aus Addierer/-Subtrahierer und Barreleshifter ein ROM, das in Abhängigkeit von den Steuervariablen m und i die zugehörigen Winkel α_i ausgibt. Aufgrund des geringen Wertebereiches für i und m sind nur wenige Werte im ROM zu speichern. Dabei wird jeder Schrittweite $2^{S(m,i)}$ ein Winkel $\alpha_{m,i}$ zugeordnet.

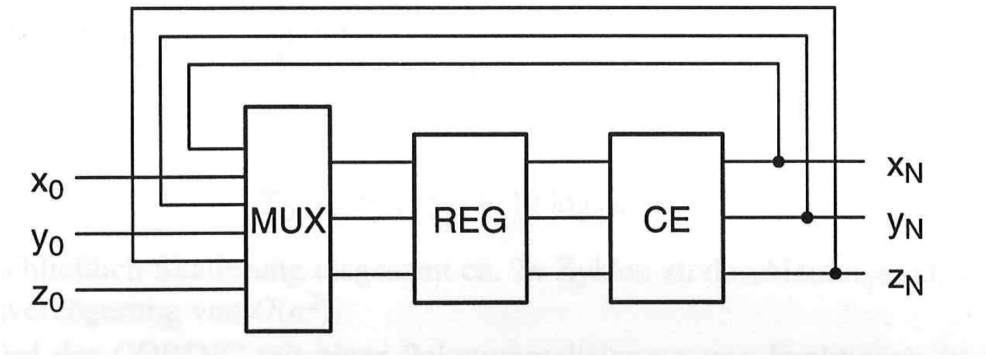

Abbildung 1: Rekursive CORDIC Architektur

Bei der in Abbildung 1 dargestellten rekursiven CORDIC Architektur ist das eigentliche CORDIC-Element (CE) eine rein kombinatorische Kombonente mit vorgelagerten Registerstufen für x_i, y_i und z_i. Das CE berechnet dann die jeweiligen Iterationen für x_{i+1}, y_{i+1} und z_{i+1}. Vor den Registerstufen befinden sich Multiplexer, die ein Initialisieren der Register vor dem Iterationsprozess auf x_0, y_0 und z_0 erlauben.

Fragen

1. Wie sind die Register für x_0, y_0 und z_0 zu setzten, wenn im *Rotationsmodus* die Werte $cos(\Theta)$ und $sin(\Theta)$ für einen gegebenen Winkel Θ berechnet werden sollen?
2. Wie sehen die Werte für einen konkreten Winkel $\Theta = 0.6$ (im Bogenmaß) bei einer 32 Bit Fixpunktdarstellung mit 24 Bit Nachkommastellen aus?

CORDIC-Element

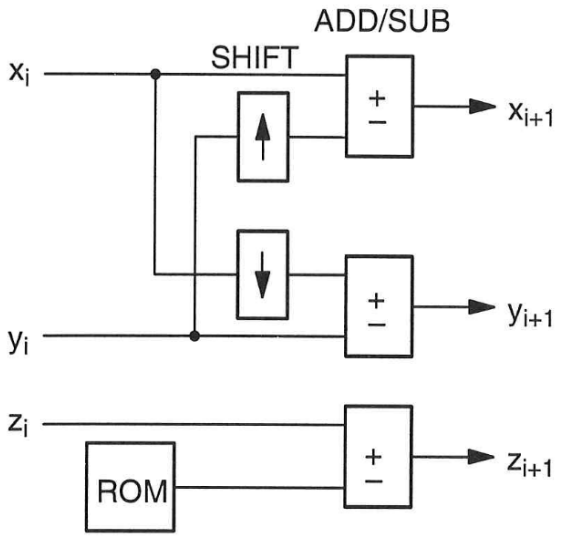

Abbildung 2: Realisierung des CORDIC-Elements.

Barrelshifter sowie Addierer/Subtrahierer und der Koeffizienten-ROM sollen im Folgenden zum CORDIC-Element entsprechend dem obigen Blockdiagramm als kombinatorische Komponente zusammengefügt werden.

```
entity m9_ce is
  port (m                   : in  std_logic;
        sigma               : in  std_logic;
        i                   : in  std_logic_vector(4 downto 0);
        x_in, y_in, z_in    : in  std_logic_vector(31 downto 0);
        x_out, y_out, z_out : out std_logic_vector(31 downto 0));
end m9_ce;
```

Die Ausgänge `x_out`, `y_out` und `z_out` dürfen bei einer rekursiven Implementierung *nicht* über Registerstufe geführt werden. Der Eingang `i` entspricht der jeweiligen Iterationsebene und steuert die Shifter als auch Koeffizienten ROM an. Der Eingang `sigma` entspricht der Vorzeicheninformation σ_i der Iterationsgleichungen während der Eingang `m` zwischen Rotations- und Vektormodus umschaltet.

Steuerwerk

Nach der Umsetzung des CORDIC-Elementes `m9_ce` fehlt zur Fertigstellung einer rekursiven Architektur noch ein entsprechendes Steuerwerk als synchroner Automat. Als Anhaltspunkt für den zugrundeliegenden Kontrollfluß ist folgende C-Implementierung zur Berechnung von $cos(\Theta)$ und $sin(\Theta)$ auf Fließkommadarstellung skizziert.

```
int main (void)
{
  int n = 1;
  double g = 1.64676025812, delta = 1;
  double tmpx, tmpy, sigma;
  double x = 1, y = 0, z = 0.6; // Initialwert fuer z entspricht Theta

  do {
    sigma = z < 0 ? -1 : 1;
    tmpx = x - (sigma * delta * y);
    tmpy = y + (sigma * delta * x);
    x = tmpx;
    y = tmpy;
    z = z - sigma * atan(delta);
    delta = delta / 2;
    n++;
  } while (n <= 24);

  x = x / g;
  y = y / g;

  printf("x = %1.8lf, y = %1.8lf\n", x, y);
  return 0;
}
```

Der Schleifenkern entspricht dem Datenpfad, der bereits durch das CORDIC-Elemente `m9_ce` realisiert wurde. Im Folgenden soll das Steuerwerk sowie die Integration vorgenommen werden.

Vorüberlegungen

1. Entwicklen Sie aus der obigen Implementierung in C ein Zustandsgraphen für Ihr Steuerwerk. Berücksichtigen Sie dabei, dass der Automat auch einen Idle-Zustand als „Grundzustand" vorsehen sollte.
2. Welche Abbruchbedingung für die Schleife ist zwingend? Welche optionale Abbruchbedingung existiert weiterhin?
3. Berücksichtigen Sie weiterhin, dass der Automat neben der eigentlichen Schleife ebenfalls ein sinnvolle Initialisierung der Register für x_i, y_i und z_i vornehmen muss.
4. Ein wesentlicher Aspekt in der Steuerung des CORDIC-Elementes ist die Berechnung der Vorzeicheninformation σ aus dem jeweiligen Zustand der Iterationsvariable z_i. Ein dem C-Programm entsprechender Vergleicher ist jedoch weder für den Daten- noch Kontrollpfad vorgesehen. Wie könnnen Sie trotzde das Steuersignal `sigma` aus z_i ableiten?
5. „Klammern" Sie die abschließende Skalierung von x und x um den Faktor g aus.

Implementierung

Auf dem vorangegangenen Aufgaben war in Abbildung 1 die rekursive CORDIC-Architektur abgebildet. Nachstehend erfolgt die Integration von Datenpfad und Steuerwerk zur Komponente `m9_cordic`.

```
entity m9_cordic is
  port (reset        : in  std_logic;
        clk          : in  std_logic;

        valid_in     : in  std_logic;
        theta        : in  std_logic_vector(31 downto 0);

        valid_out    : out std_logic;
        cos_theta    : out std_logic_vector(31 downto 0);
        sin_theta    : out std_logic_vector(31 downto 0));
end m9_cordic;
```

Hierbei soll ein 32 Bit Fixpoint-Wert für den Winkel Θ durch das Eingangssignal `theta` an die Schaltung übergeben werden. Die Gültigkeit wird über das Signal `valid_in = 1` signalisiert und startet den Iterationsprozess. Ist letzterer abgeschlossen, werden die Werte von $sin(\Theta)$ und $cos(\Theta)$ auf den entsprechenden Ausgangssignalen `sin_theta` und `cos_theta` ausgegeben. Die Schaltung signalisiert einen Abschluß des Iterationsprozesses und die Gültigkeit der Signale über das Ausgangssignal `valid_out = 1`.