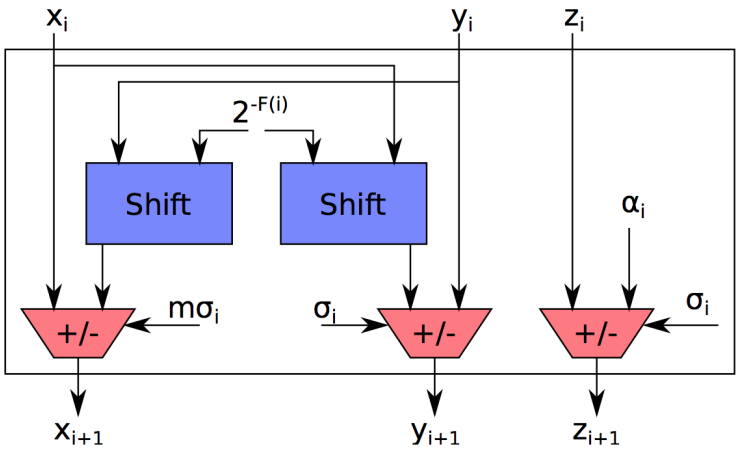
4. Übung	**M9 VLSI Anwendungen** Hochschule für Technik und Wirtschaft	SS 2021

Ausgabe: 3./10. Juni 2021 Abgabe: 17./24. Juni 2020

Schwerpunkte

- Arithmetische Funktionseinheiten
- Synchrones Design
- CORDIC Algorithmus

Der CORDIC-Algorithmus kann prinzipiell durch N in Kaskade betriebene CORDIC-Elemente (CE) realisiert werden. Jedes dieser CORDIC-Elemente führt eine Mikrorotation durch. Als steuernde Größe muß jedem Element der Mode-Parameter (Rotation oder Vectoring), der Koordinatensystemparameter m und die Iterationsebene i zugeführt werden.



Eine effiziente Ausnutzung der Hardware wird durch eine rekursive Anordnung eines CE mit Registerstufen und Rückführung über Multiplexer realisiert. Das CORDIC-Element besteht, wie in der obigen Abbildung entnommen werden kann, im Wesentlichen aus Schiebeeinheiten. Addierer/Subtrahierer und dem Winkelspeicher, der die Werte α_i bereitstellt.

Barrel-Shifter

Implementieren Sie eine Architektur `behaviour` der Shifter-Entität `m9_barrel32`, die Rechtsschiebeoperation als *Barrel Shifter* realisiert. Die Schiebeeinheit soll der folgenden Entity-Spezifikation genügen und auf 32 Bit breiten Operanden arbeiten.

```
entity m9_barrel32 is
  port (x     : in  std_logic_vector(31 downto 0);
        pos   : in  std_logic_vector(4 downto 0);
        y     : in  std_logic_vector(31 downto 0));
end m9_barrel32;
```

Der Eingang `pos` legt die Anzahl von Stellen fest, um der Eingangsverktor `x` nach recht (`lnr = '0'`) geschoben werden soll. Achten Sie darauf, dass Ihr Barrelshifter *arithmetische* Schiebeoperationen durchführt. Testen Sie Ihre Schaltung mittels einer geeigneten Testbench.

Addierer-Subtrahierer

Die zweite Komponente der CORDIC-Einheit stellt der Addierer-Subtrahierer `m9_addsub32` dar.

```
entity m9_addsub32 is
  port (a, b  : in  std_logic_vector(31 downto 0);
        ans   : in  std_logic;
        y     : in  std_logic_vector(31 downto 0));
end m9_barrel;
```

In Ahängigkeit von der Steuerleitung `ans` wird entweder die Summe (`ans = '1'`) der Eingangsvektoren a und b bzw. die Differenz (`ans = '0'`) berechnet. Implementieren Sie eine Architektur `structure` der Entität `m9_addsub32` unter Verwendung des von Ihnen bereits realisierten Carry-Select-Addierers `m9_adder` für eine Operandenbreite von 32 Bit. Testen Sie Ihre Schaltung mittels einer geeigneten Testbench.

Winkel ROM

Implementieren Sie eine Architektur `behaviour` eines ROMs `m9_artan_rom`, das im Rotationsmodus jedem der möglichen Teilwinkel $tan\alpha_i = 2^{-i}$ einen Wert zuordnet. Dabei ergibt sich der Eingangswert i aus dem jeweiligen Iterationsschritt. Der Ausgangswert sollte als Zweierkomplement in Fixpunktdarstellung abgelegt werden. Wobei 24 Bit für Nachkommastellen vorzusehen sind.

```
entity m9_artan_rom is
  port (i     : in  std_logic_vector(4 downto 0);
        d     : out std_logic_vector(31 downto 0));
end m9_artan_rom;
```

Hinweis: Für Xilinx FPGAs synthesefähige Beschreibungen von ROMs lassen sich als Arrays konstanter Werte darstellen:

```
type fix824 is array(31 downto 0) of std_logic_vector(31 downto 0);

constant rom_arctan: fix824:= (
  "00000000000000000000000000011111",
  "00000000000000000000000000111111",
  ...
  "00000000000000000000000001111111");
```