3. Übung

M9 VLSI Architekturen

Hochschule für Technik und Wirtschaft

Schwerpunkte

- Arithmetische Funktionseinheiten
- Synchrones Design
- Pipelines

Aufgabe 1

Grundstruktur eines seriellen Multiplizierers dargestellt. Die Quelloperanden A und B werden durch iterierte Addition in das Produkt Y = A * B Überführt (das Ergebnisregister Y ist anfänglich auf Null gesetzt). Dabei wird in jedem Iterationsschritt wie folgt verfahren:

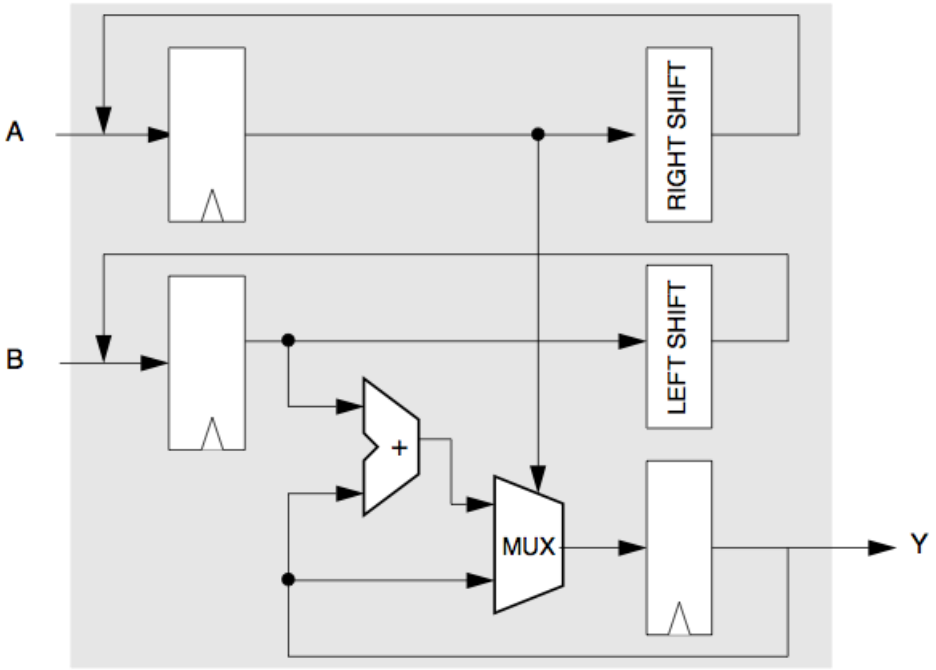

Abbildung 1: Struktur eines seriellen Multiplizierers

Das niederwertigste Bit (LBS) des Operanden A entscheidet, ob im aktuellen Schritt der Operand B zum Ergebnisregister Y hinzuaddier werden muß (LSB = ‚1') oder nicht (LSB = ‚0'). Anschließend wird für den nächsten Iterationsschritt der Operand A um eine Stelle nach rechts bzw. der Operand B um eine Stelle nach links verschoben. Die Multiplikation ist abgeschlossen, wenn eines der Operandenregister Null ist. Serielle Multiplizierer zeichnen sich durch einen geringen Flächenaufwand aus, sind aber auch entsprechend langsam; im ungänstigsten Fall benötigt z.B. ein 16-Bit Multiplizierer 16 Takte zur Ausführung. Im folgenden soll die Struktur eines seriellen Multiplizierer abgerollt und in eine Pipeline-Architektur überführt werden.

Aufgabe 1.1

In Abbildung 2 ist die Struktur des abgerollten Multiplizierers dargestellt. Sie besteht aus einer

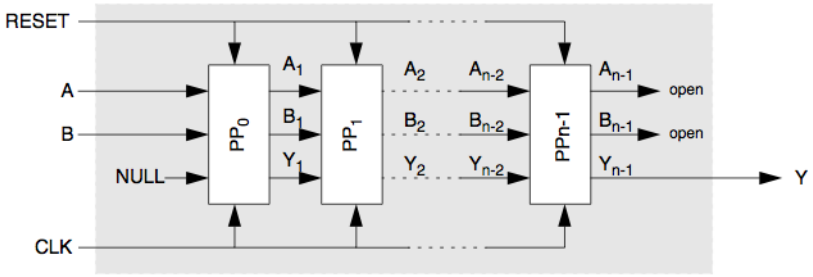

Abbildung 2: Multiplizierzelle zur Berechnung von Partialprodukten

Verkettung von Komponenten zur Berechnung von Partialprodukten, die dem Kern des seriellen Multiplizierers entsprechen. Entwerfen Sie eine generische Komponente `m9_pp` zur Berech- nung des Partialproduktes, die der folgenden Entity-Spezifikation gerecht wird:

```
 entity m9_pp is
         generic (width_opa_in   : positive;
                  width_opb_in   : positive;
                  width_prod_in  : positive;
                  width_opa_out  : positive;
                  width_opb_out  : positive;
                  width_prod_out : positive);
port (clk   : in std_logic;
      reset : in std_logic;
      a_in  : in std_logic_vector(width_opa_in - 1 downto 0);
      b_in  : in std_logic_vector(width_opb_in - 1 downto 0);
      y_in  : in std_logic_vector(width_prod_in - 1 downto 0);
      a_out : out std_logic_vector(width_opa_out - 1 downto 0);
      b_out : out std_logic_vector(width_opb_out - 1 downto 0);
      y_out : out std_logic_vector(width_prod_out - 1 downto 0));
end m9_pp;
```

Achten Sie beim Entwurf der Architektur synchronous der Entity `m9_pp darauf`, daß alle Signale krauser Logik vor dem ‘Verlassen’ der Komponente über Registerstufen führt werden.

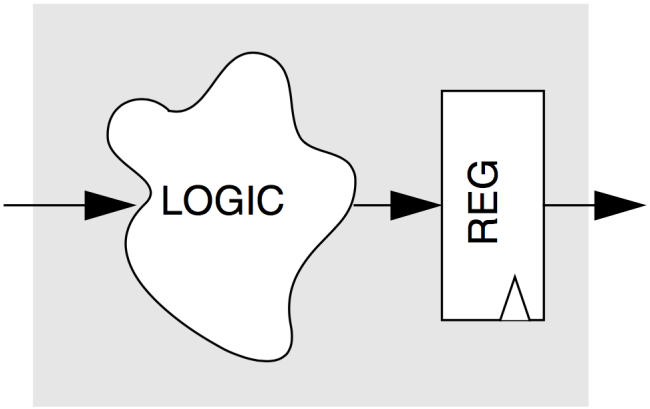

Abbildung 3: Grundstruktur von Komponenten im synchronen Schaltungsdesign

Aufgabe 1.2

Implementieren Sie eine Architektur pipelined der folgenden Entity-Spezifikation `m9_mul`, die die in Abbildung 2 dargestellte Struktur des abgerollten Multiplizierers realisiert und sich der generate Anweisung bedient.

```
entity m9_mul
         generic (width : positive := 8);
   port (clk, reset : in  std_logic;
         a, b : in   std_logic_vector(width - 1 downto 0);
         y    : out  std_logic_vector((2 * width) - 1 downto 0));
end m9_mul;
```

Testen Sie ihre Implementierung für eine Operandenbreite von `width = 4` mittels einer geeigneten Testbench.

Aufgabe 1.2

Für die CE81 0.18 μm Gatearraytechnologie von Fujitsu wurden für nachstehende Komponenten mit dem Synopsys Design Compiler folgende Verzögerungszeiten ermittelt:

- 32-Bit Ripple Carry Addierer tpadd32 = 9124 ps
- 32-Bit 2:1 Multiplexer tpmux = 1898 ps
- 32-Bit Links/Rechtsschieber (1 Position) tpshift = 1092 ps

Geben Sie eine grobe Schätzung der zu erwartenden Systemfrequenz des Pipeline-Multiplizieres für eine eingangsseitige Operandenbreite von 16 Bit ab.