Lernzettel

Programmierbare Logik: PLA/PAL/FPGA-Grundlagen, LUT-basierte Implementierung

Universität: Technische Universität Berlin
Kurs/Modul: Technische Grundlagen der Informatik (TechGI) - Digitale Systeme
Erstellungsdatum: September 6, 2025

Lernzettel: Programmierbare Logik – PLA/PAL/FPGA-Grundlagen, LUT-basierte Implementierung
Kurs: Technische Grundlagen der Informatik (TechGI) - Digitale Systeme

(1) Grundbegriffe und Bauarten. Programmierbare Logik stellt flexible Bausteine bereit, die sich per Konfiguration an unterschiedliche Funktionen anpassen lassen.

PLA. Eine PLA besitzt eine programmierbare AND-Ebene (Produktterm) und eine programmierbare OR-Ebene (Summe der Produkte).

$$F_j = \bigvee_i \left(\bigwedge_{k \in S_{j,i}} x_k \right)$$

wobei $S_{j,i}$ die Eingänge des j-ten Produktterms beschreibt.

PAL. PALs verwenden eine programmierbare AND-Ebene und eine feste OR-Ebene. Die OR-Verknüpfung ist hierbei fest verdrahtet und damit unverändert.

FPGA. FPGAs nutzen Look-Up-Tables (LUTs) als fundamentale Bausteine, ergänzt durch CLBs, Router-Netze und Speicherelemente. Eine LUT der Größe n implementiert beliebige Funktionen von n Eingängen.

(2) LUT-basierte Implementierung. Eine LUT mit n Eingängen besitzt 2^n Konfigurationswerte. Diese Werte definieren die Ausgabe für alle möglichen Eingangszustände.

Beispiel: 3-Eingangs-LUT. Eine 3-Eingangs-LUT implementiert jede Funktion von drei Booleschen Variablen a, b, c durch eine eindeutige Konfigurations-Tabelle mit $2^3 = 8$ Einträgen.

Beispiel-Funktion.

$$f(a, b, c) = a\, b + c$$

Die Konfigurationsbits einer 3-Eingangs-LUT Die Ausgaben folgen der Eingangsreihenfolge $(a, b, c) = (0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)$ und lauten:

$$0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1.$$

(3) Mapping von Funktionen auf LUTs.

- Definiere die Ziel-Funktion f und ihre Eingänge.
- Transformiere oder minimiere die Funktion so, dass sie durch LUTs abgebildet werden kann.
- Ordne die Eingänge oder Teilfunktionen auf einzelne LUTs zu.
- Verknüpfe die LUT-Ausgaben über Interconnects zu der gewünschten Gesamtausgabe.

(4) Praktische Beispiele. Beispiel 1: XOR mit einer 2-Eingangs-LUT. Für eine 2-Eingangs-LUT (Inputs a, b) lauten die Konfigurationsbits (für die Ausgänge bei 00, 01, 10, 11):

$$0,\ 1,\ 1,\ 0.$$

(5) Vorteile und Leistungsparameter.

- Hohe Parallelität durch zahlreiche LUTs; geringe Logik-Tiefe.
- Flexible, skalierbare Struktur durch Interconnects.
- Leistungsparameter: dynamische Leistung durch Schaltvorgänge, lokale Speicherelemente, Standby-Verlust.

(6) Entwurfsvorgehen (Mapping und Implementierung).

- Funktionsdefinition und Eingangs-Auswahl.
- Funktionsminimierung bzw. Transformation in LUT-kompatible Strukturen.
- Zuordnung der Teilfunktionen zu LUTs.
- Platzierung und Routen (Place Route) unter Berücksichtigung von Verzögerungen.
- Timing-Analyse: Propagation Delay, Setup-/Hold-Zeiten.

(7) Kurzes Praxisbeispiel: Kleinster LUT-basierter Implementierungsweg. Betrachte die Funktion $f(a, b, c) = ab+c$. Diese kann durch eine einzige 3-Eingangs-LUT implementiert werden. Die Konfigurationsbits ergeben die Ausgabereihenfolge $000, 001, 010, 011, 100, 101, 110, 111$ wie oben angegeben.

(8) Fazit. Programmierbare Logik (PLA/PAL) bietet flexible Strukturen, während FPGA-LUT-basierte Implementierungen effiziente, skalierbare Lösungen für komplexe digitale Systeme ermöglichen.