# Leonidas

## TP 2 Constraints für den Layoutentwurf

2.3 Constraint Management im Full-Custom Entwurfsablauf

A. Müller, B. Walliser, L. Kronenberg, Infineon Technologies AG

## 1. Problem

Nebenbedingungen/Constraints sind:

- oft *nicht formalisiertes Erfahrungswissen* von Designern und Layoutern
- falls formalisert, für *verschiedene* EDA-Tools *mehrmals* in proprietären Formaten einzugeben
- meist *nicht wiederverwendbar* für Neuentwürfe (Redesigns), da nicht systematisch erfaßt und verwaltet

Fehlende durchgängige Verwaltung von Constraint-Informationen.

## 2. Lösung

Zentrales und tool-unabhängiges Constraint-Management

## 3. Definition

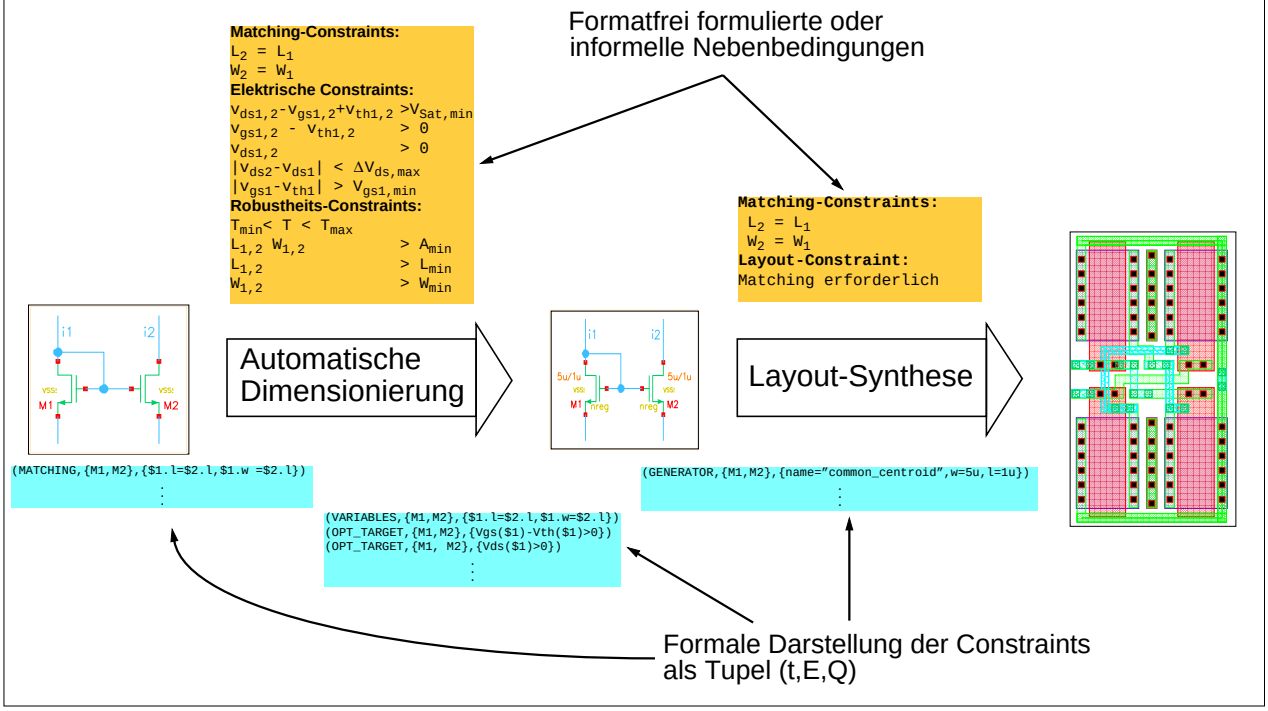
Constraints sind schaltungsspezifische Zusatzinformationen zu Designdaten (Schematic-/Layout-Daten), die das Verhalten von EDA-Werkzeugen beeinflussen/steuern.

## 4. Formale Darstellung

Einführung von Parametern für Gruppen von Designelementen. Darstellung der Parameter in Form eines Tupels (t,E,Q) mit:

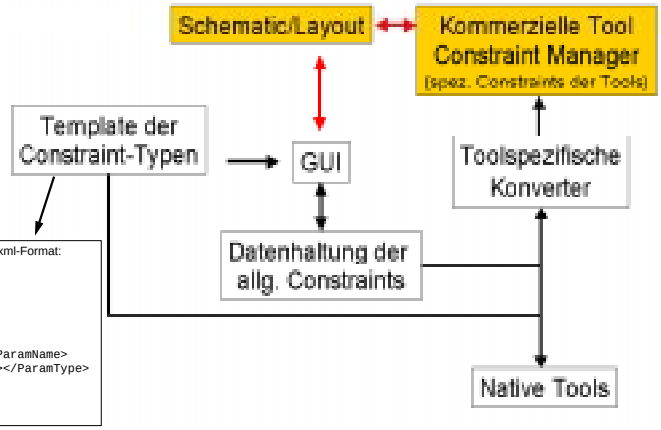
- t: Constraint-Typ
- E: Menge von Design-Elementen
- Q: Qualifier: Werte/Eigenschaften

## Beispiel: Stromspiegel




## 5. Implementierungsvorhaben

- Formale Deklaration von Constraint-Typen in einer XML-Beschreibung (Constraint-Template)
- C++-API und eventuell auch APIs für Design-System spezifische Programmiersprachen (Skill, Ample)
- Datenhaltung Design-System unabhängig und spezifisch möglich
- Grafisches Benutzerinterface (GUI) mit Anbindung an mehrere Design-Systeme
- Automatische Konfiguration des GUIs über Constraint-Template




Einheitliche Deklaration der Constraint-Typen im xml-Format:

```
<Type>
  <Name>Matching</Name>
  <ElementType type="instance"/>
  <ElementLength min="2"/>
  <ParamDeclaration>
    <ParamName>Matching_Conditions</ParamName>
    <ParamType type="equationsystem"></ParamType>
    <ParamFlag flag="mandatory"/>
  </ParamDeclaration>
</Type>
```