UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Effiziente Überwachung von Laufzeiteigenschaften in Soft- und Hardware

Normann Decker[1] Philip Gottschling[2]

[1]Institut für Softwaretechnik und Programmiersprachen
Universität zu Lübeck
decker@isp.uni-luebeck.de

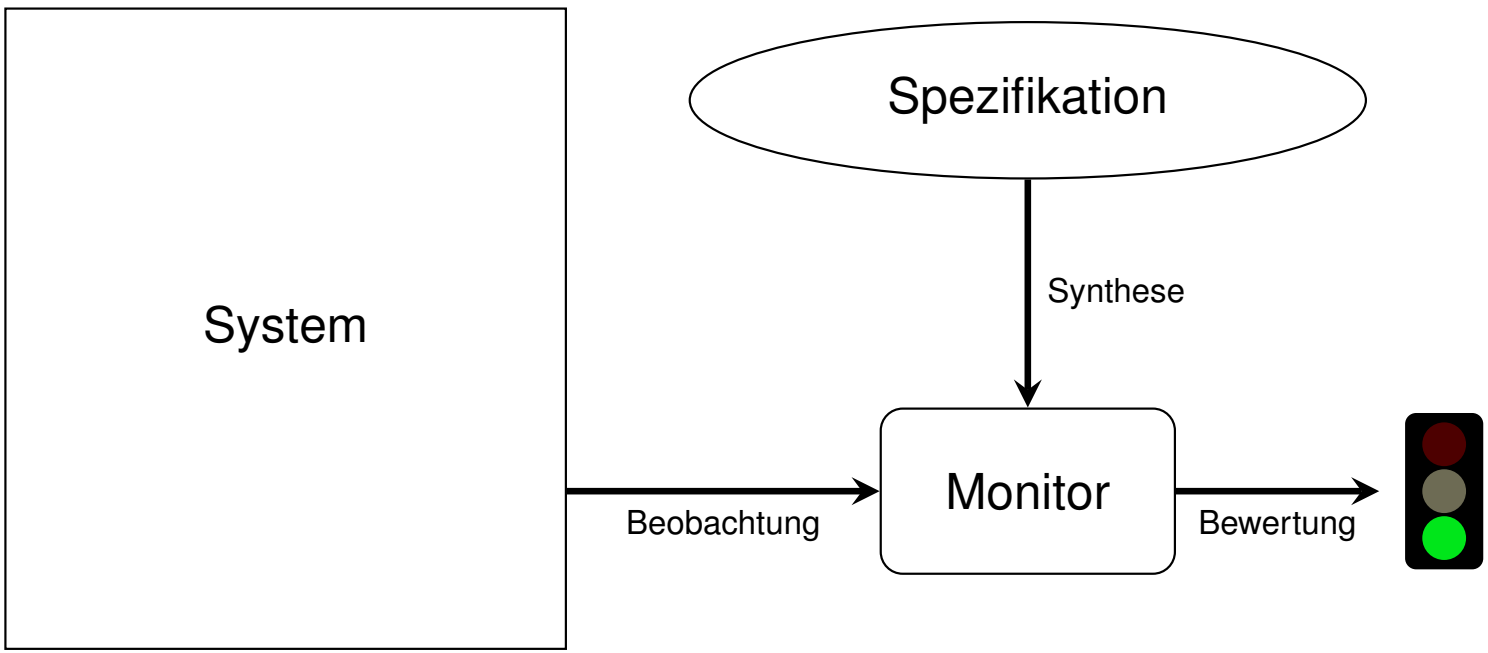
[2]Institut für Rechnersysteme
TU Darmstadt
gottschling@rs.tu-darmstadt.de

KoSSE-Workshop 2014

Lübeck, 11. Dezember 2014

Übersicht

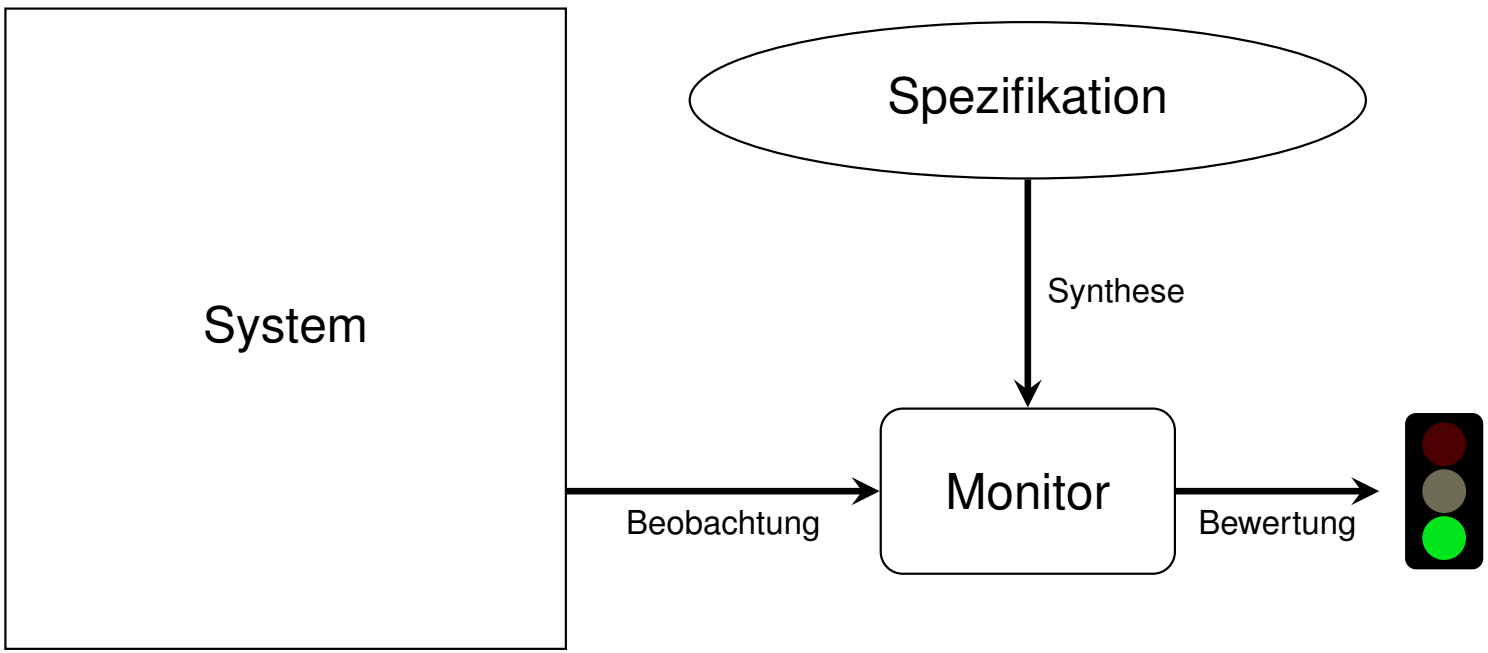
Laufzeitverifikation



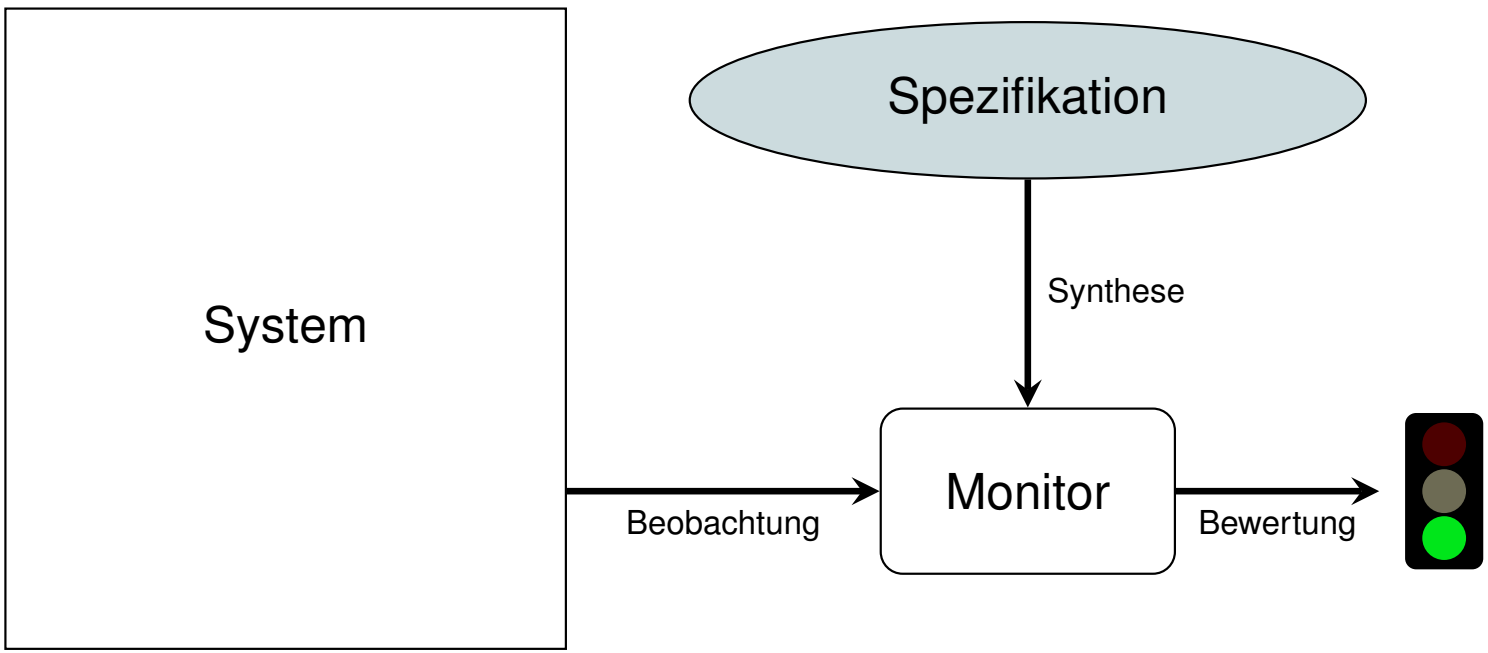
Laufzeitverifikation

- Vollständige Verifikation nicht immer möglich
  - Reaktive Systeme abhängig von Eingaben
  - System bei Auslieferung unvollständig (Dynamische Bibliotheken, Komponenten)
  - Physische Einflüsse (Hardware, Betriebsbedingungen)
- Überwachung existierender Systeme
- Testen, Analyse von Log-Daten

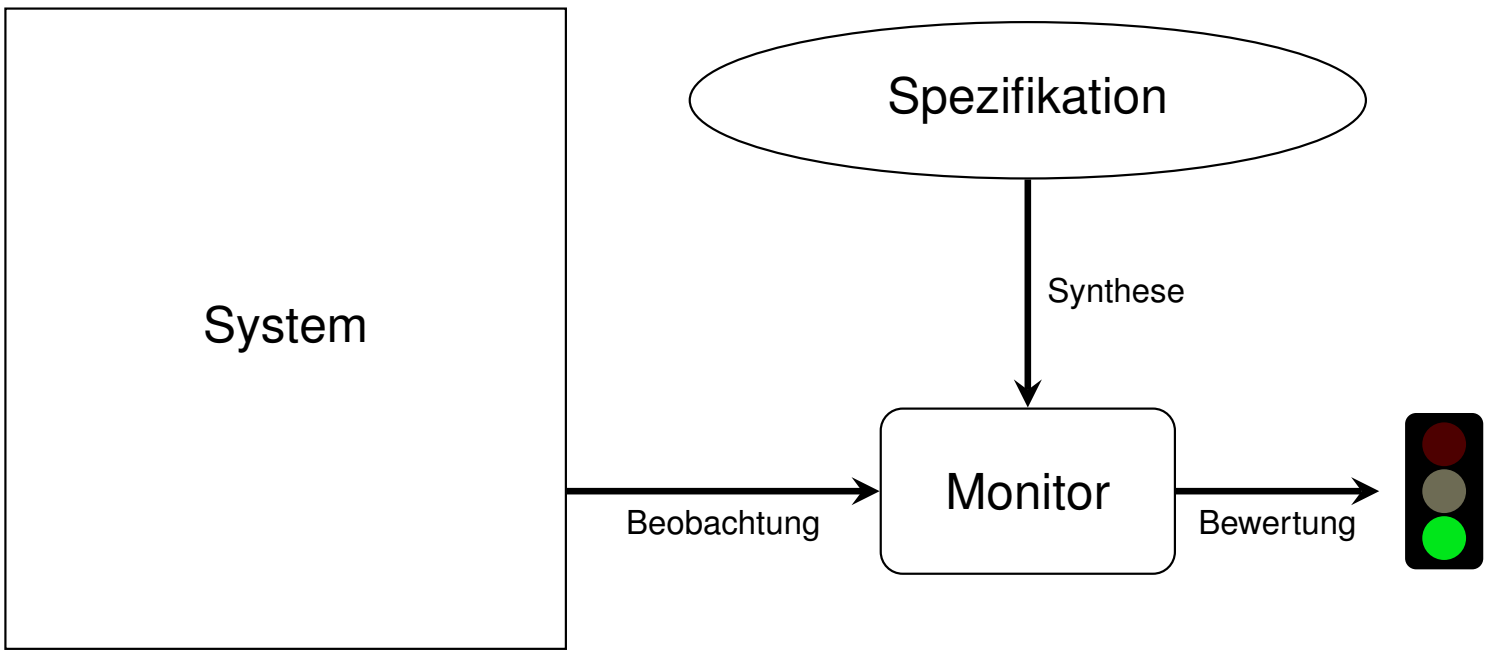
Laufzeitverifikation



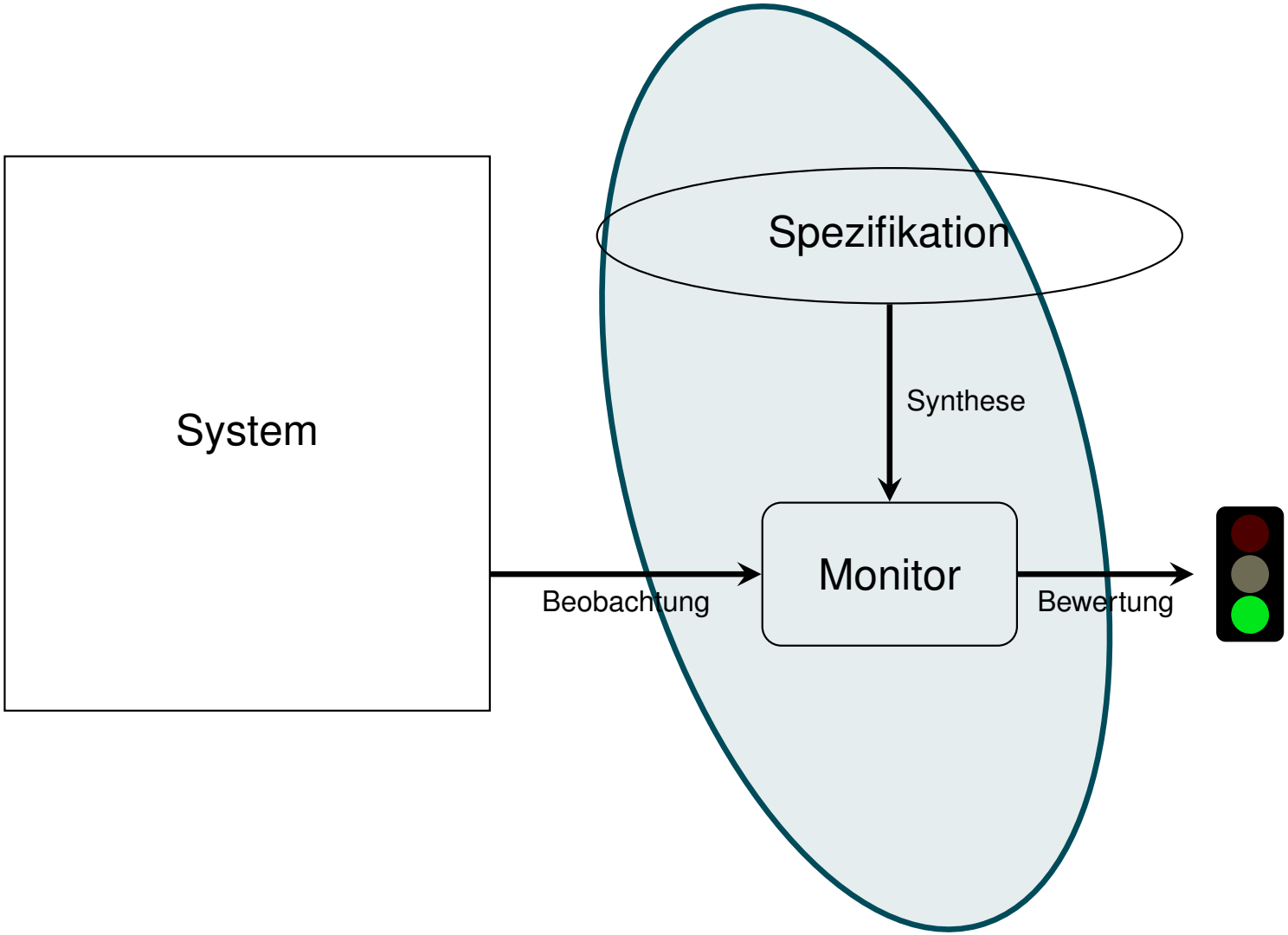
Laufzeitverifikation



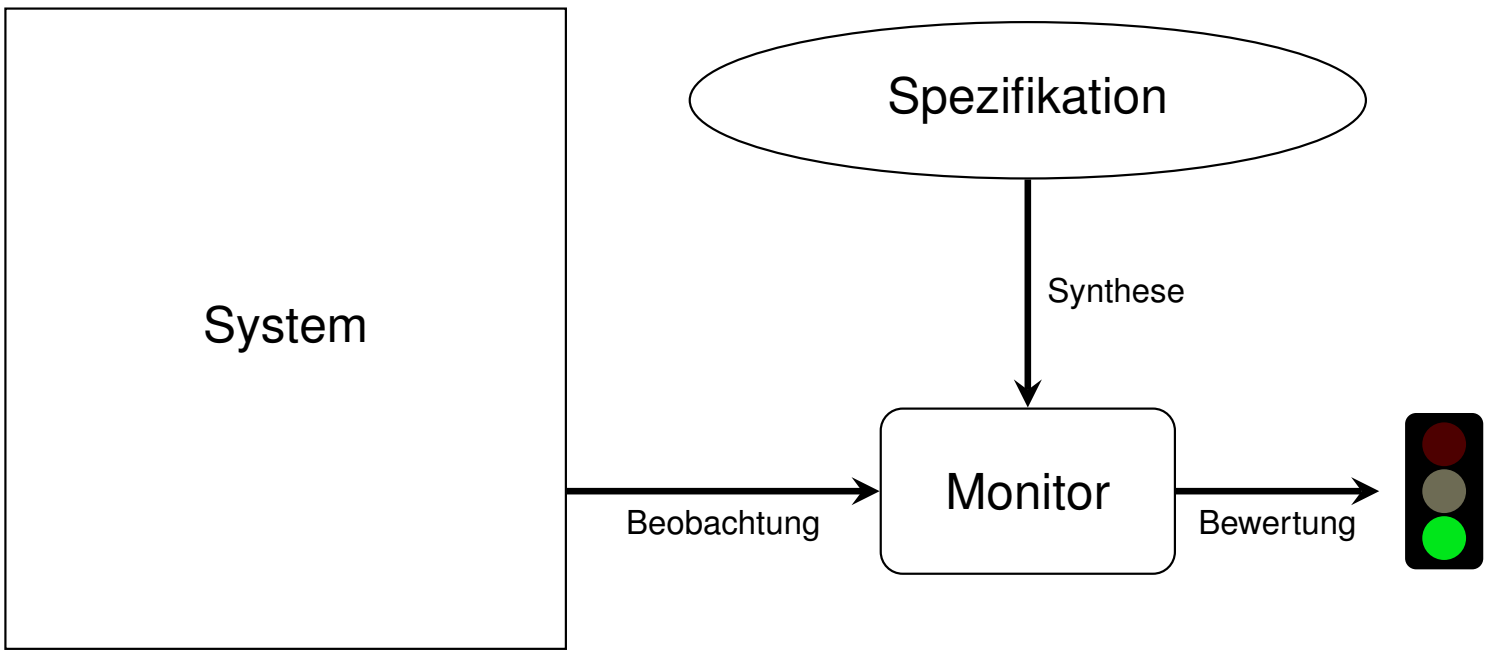
Laufzeitverifikation



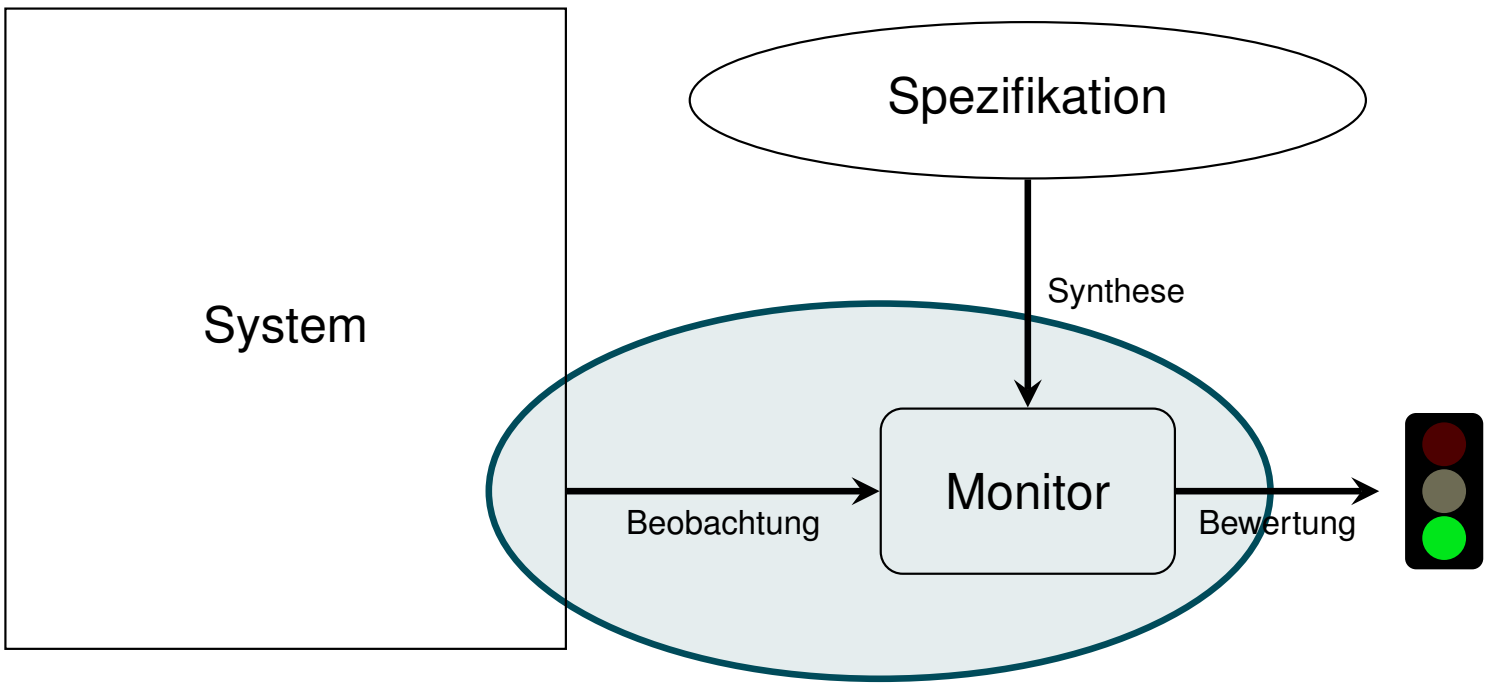
Laufzeitverifikation



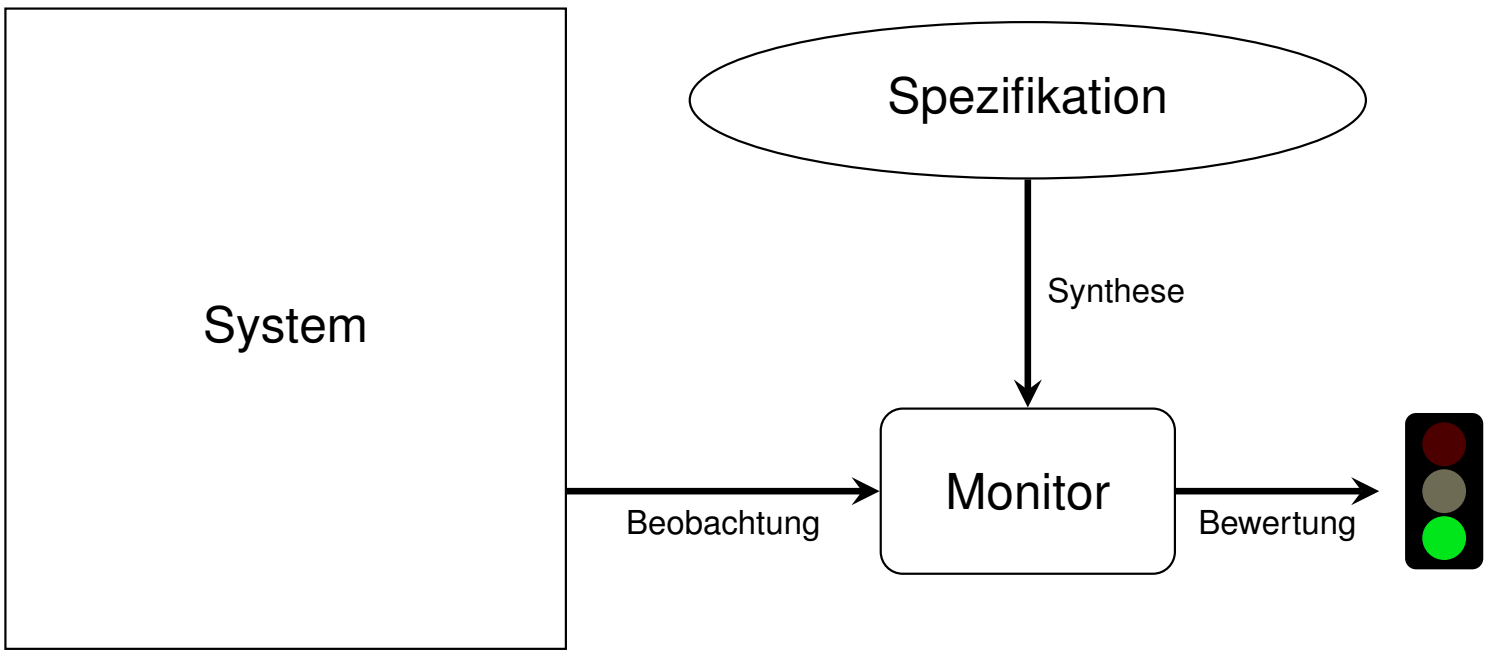
Laufzeitverifikation



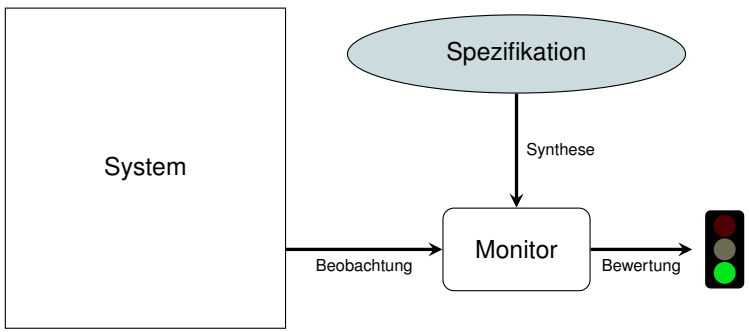
Laufzeitverifikation



Laufzeitverifikation

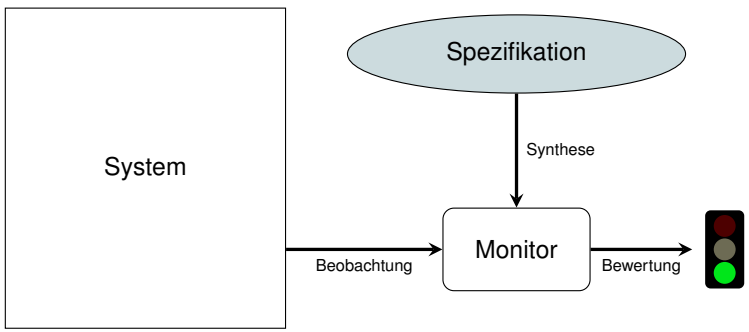


Spezifikation



- Beobachtung: Sequenz von Ereignissen $e_1 e_2 e_3 \ldots$
- Spezifikation: Eigenschaften (Form) von Sequenzen

Spezifikation



- Beobachtung: Sequenz von Ereignissen $e_1 e_2 e_3 \ldots$
- Spezifikation: Eigenschaften (Form) von Sequenzen

Spezifikation - Beispiele

- Ereignisse definieren: $a, b, c, \ldots$, z. B.
  - $a \mathrel{\hat{=}}$ „Fehler aufgetreten“
  - $b \mathrel{\hat{=}}$ „Initialisierung abgeschlossen“
  - $c \mathrel{\hat{=}}$ „Funktion `f()` aufgerufen“
  - ...
- „Niemals a“: Always(Not a)
- „Kein Fehler in der Initialisierungsphase“: (Not a) Until b
- (Always Not error) And (Eventually b) How

Spezifikation - Beispiele

- Ereignisse definieren: $a, b, c, \ldots$, z. B.
  - $a \mathrel{\hat{=}}$ „Fehler aufgetreten“
  - $b \mathrel{\hat{=}}$ „Initialisierung abgeschlossen“
  - $c \mathrel{\hat{=}}$ „Funktion `f()` aufgerufen“
  - ...
- „Niemals a“: $\mathrm{Always}(\mathrm{Not}\ a)$
- „Kein Fehler in der Initialisierungsphase“: $(\mathrm{Not}\ a)\ \mathrm{Until}\ b$
- $(\mathrm{Always}\ \mathrm{Not}\ \mathit{error})\ \mathrm{And}\ \mathrm{Eventually}\ \mathit{job\_done}$

Spezifikation - Beispiele

- Ereignisse definieren: $a, b, c, \ldots$, z. B.
  - $a \mathrel{\hat{=}}$ „Fehler aufgetreten“
  - $b \mathrel{\hat{=}}$ „Initialisierung abgeschlossen“
  - $c \mathrel{\hat{=}}$ „Funktion `f()` aufgerufen“
  - ...
- „Niemals a“: $\mathrm{Always}(\mathrm{Not}\ a)$
- „Kein Fehler in der Initialisierungsphase“: $(\mathrm{Not}\ a)\ \mathrm{Until}\ b$
- $(\mathrm{Always}\ \mathrm{Not}\ \mathit{error})\ \mathrm{And}\ \mathrm{Eventually}\ \mathit{job\_done}$

Spezifikation - Beispiele

- Ereignisse definieren: $a, b, c, \ldots$, z. B.
  - $a \mathrel{\hat{=}}$ „Fehler aufgetreten“
  - $b \mathrel{\hat{=}}$ „Initialisierung abgeschlossen“
  - $c \mathrel{\hat{=}}$ „Funktion `f()` aufgerufen“
  - ...
- „Niemals a“: $\mathrm{Always}(\mathrm{Not}\ a)$
- „Kein Fehler in der Initialisierungsphase“: $(\mathrm{Not}\ a)\ \mathrm{Until}\ b$
- $(\mathrm{Always}\ \mathrm{Not}\ \mathit{error})\ \mathrm{And}\ \mathrm{Eventually}\ \mathit{job\_done}$

Spezifikation

- Abstrakt, deklarativ (kein „programmieren“ der Spezifikation)
- Eindeutige Bedeutung
  - Automatische Verarbeitung (z. B. Monitorsynthese)
  - Optimierung

Spezifikation

- Abstrakt, deklarativ (kein „programmieren“ der Spezifikation)
- Eindeutige Bedeutung
  - Automatische Verarbeitung (z. B. Monitorsynthese)
  - Optimierung

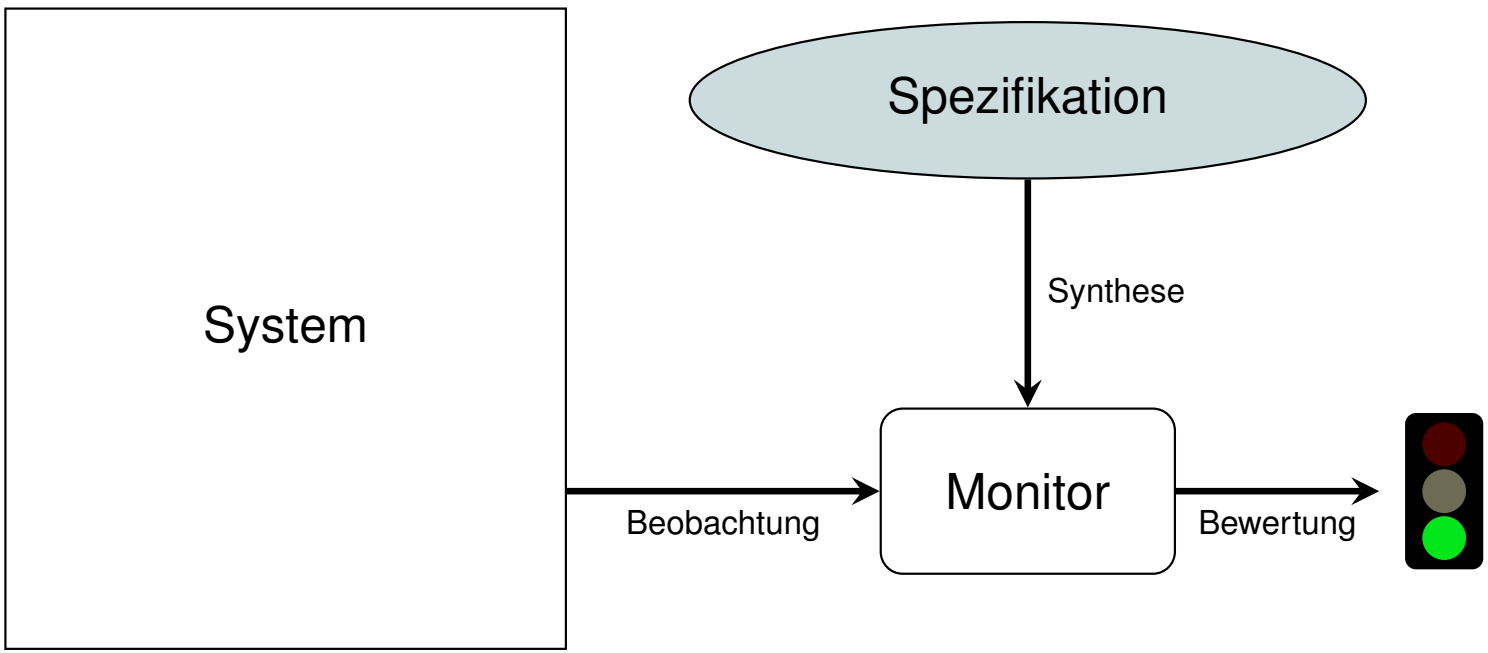
Spezifikation

- Abstrakt, deklarativ (kein „programmieren“ der Spezifikation)
- Eindeutige Bedeutung
  - Automatische Verarbeitung (z. B. Monitorsynthese)
  - Optimierung

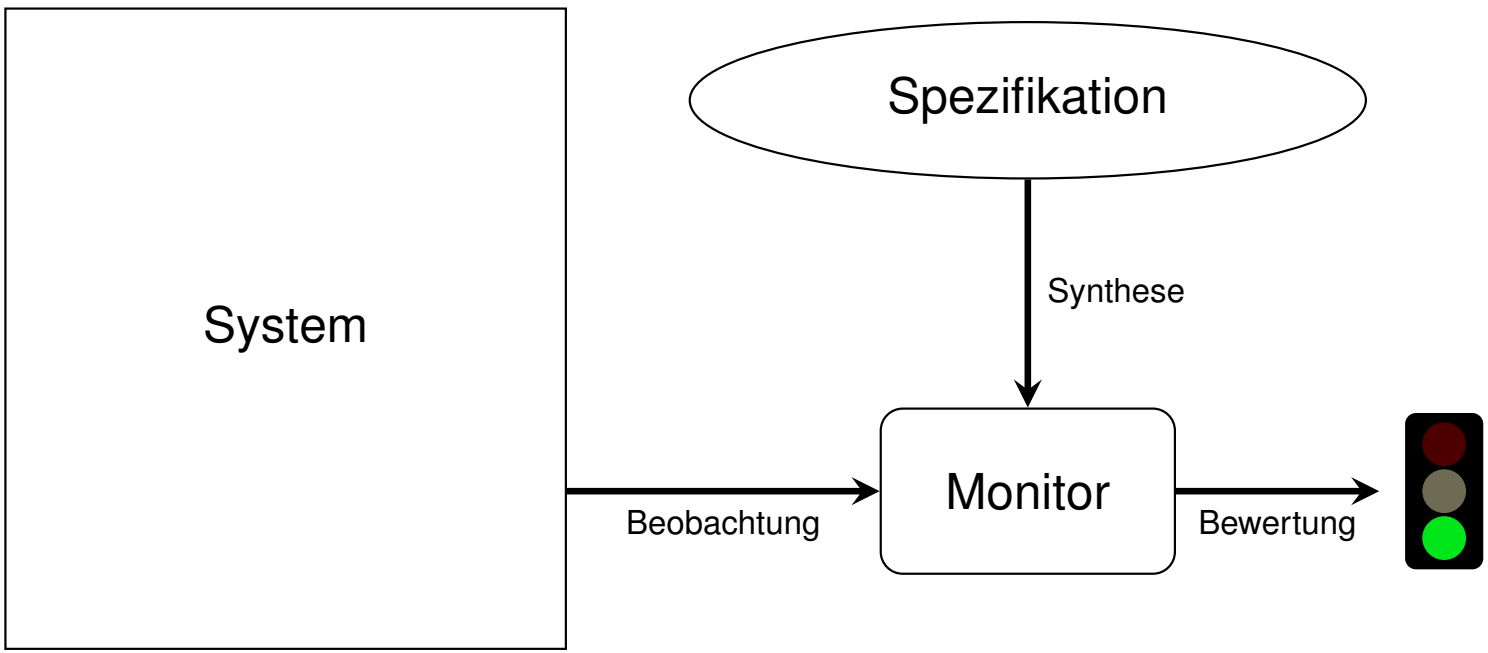
Spezifikation

- Abstrakt, deklarativ (kein „programmieren“ der Spezifikation)
- Eindeutige Bedeutung
  - Automatische Verarbeitung (z. B. Monitorsynthese)
  - Optimierung

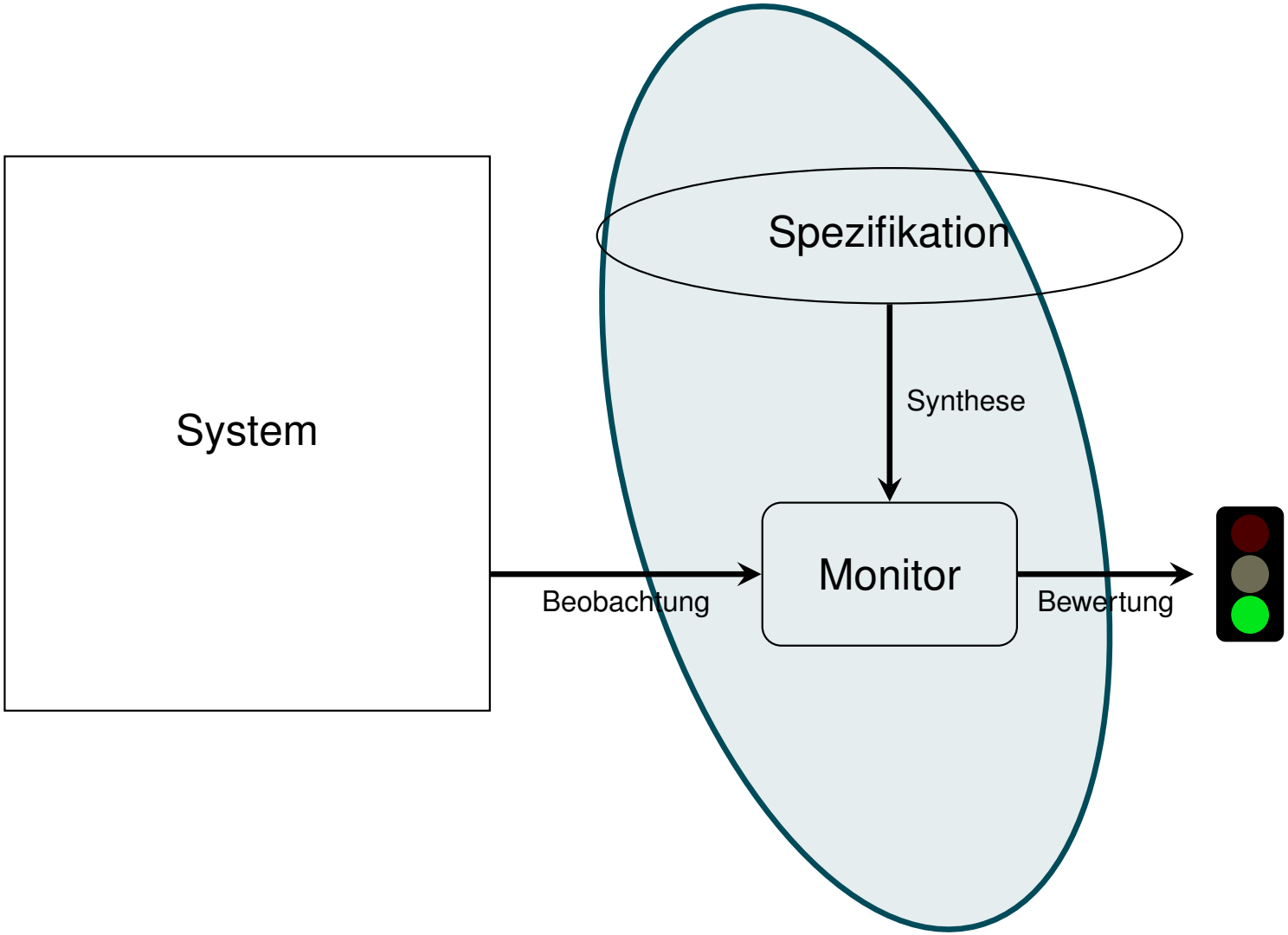
Laufzeitverifikation



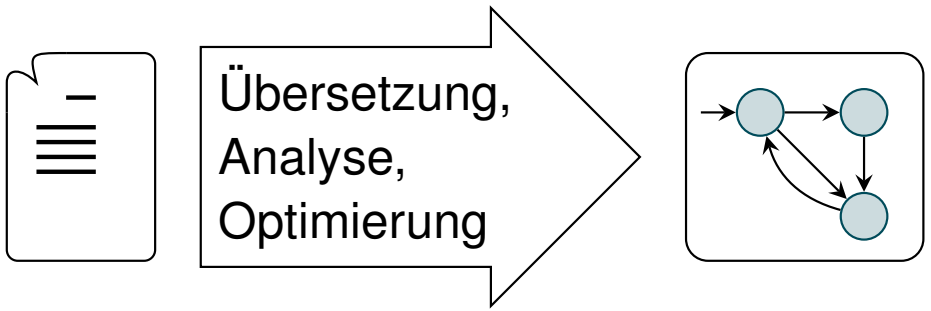
Laufzeitverifikation



Laufzeitverifikation



Monitorsynthese

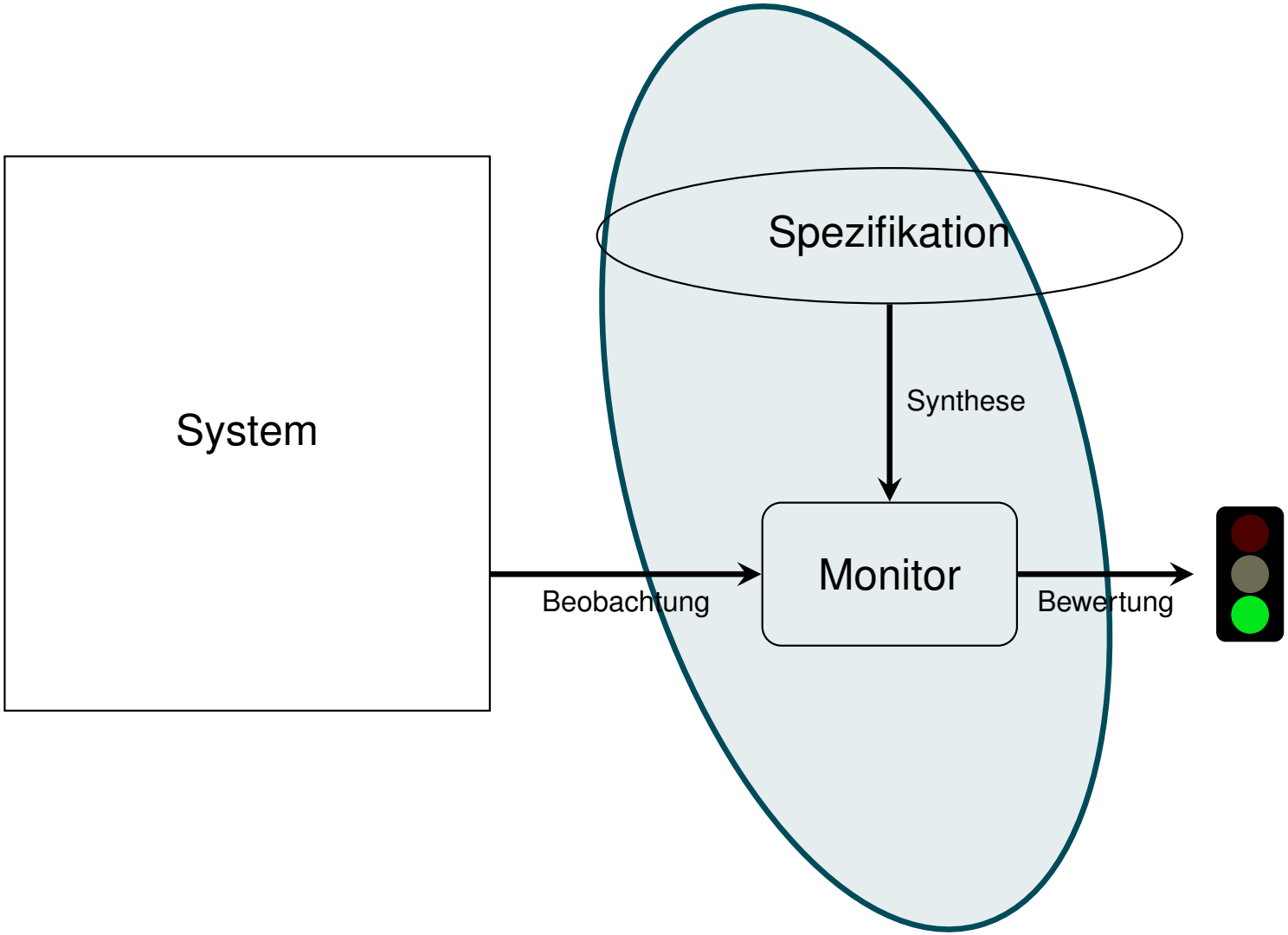


Spezifikation (deklarativ)

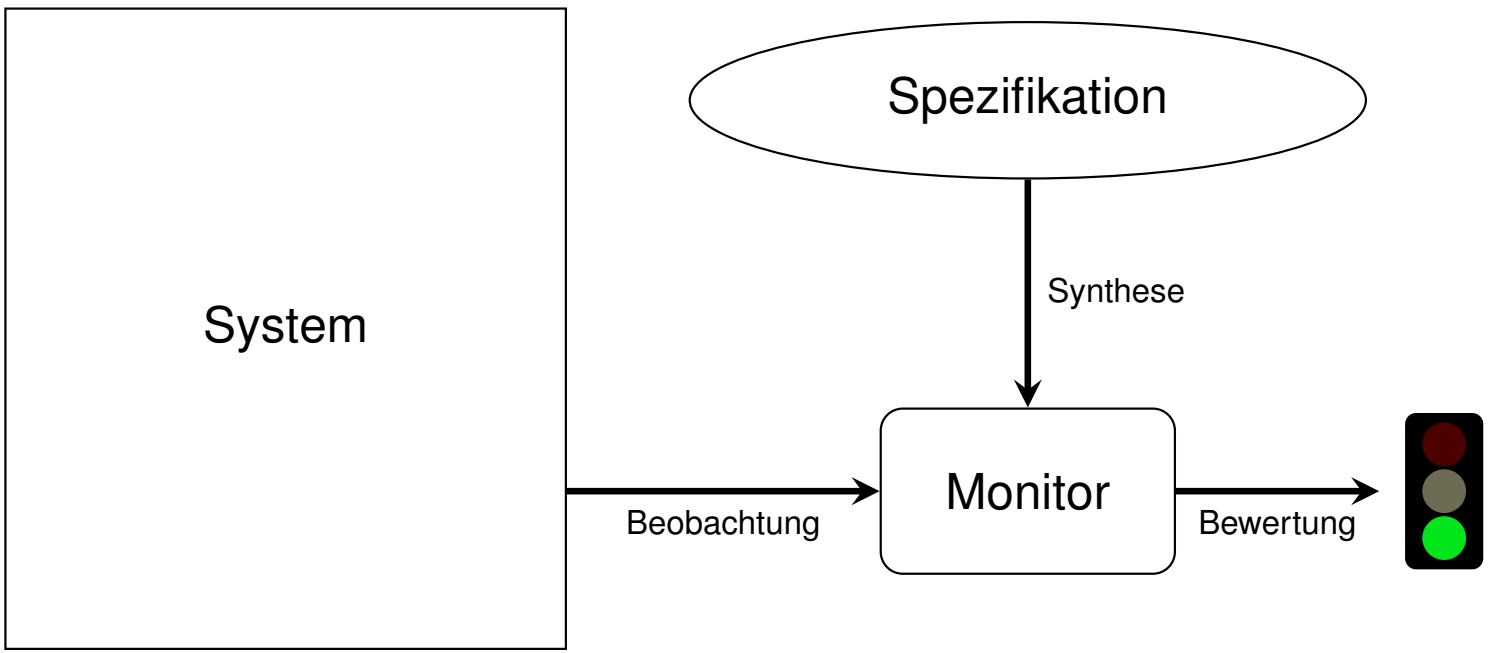
Monitor (operational, mit Ausgabe)

- Synthese eines operationalen Modells (z. B. Zustandsmaschine)
- Optimierung (z. B. semantische Redundanz, spezifische Implementierung)
- Vollautomatisch

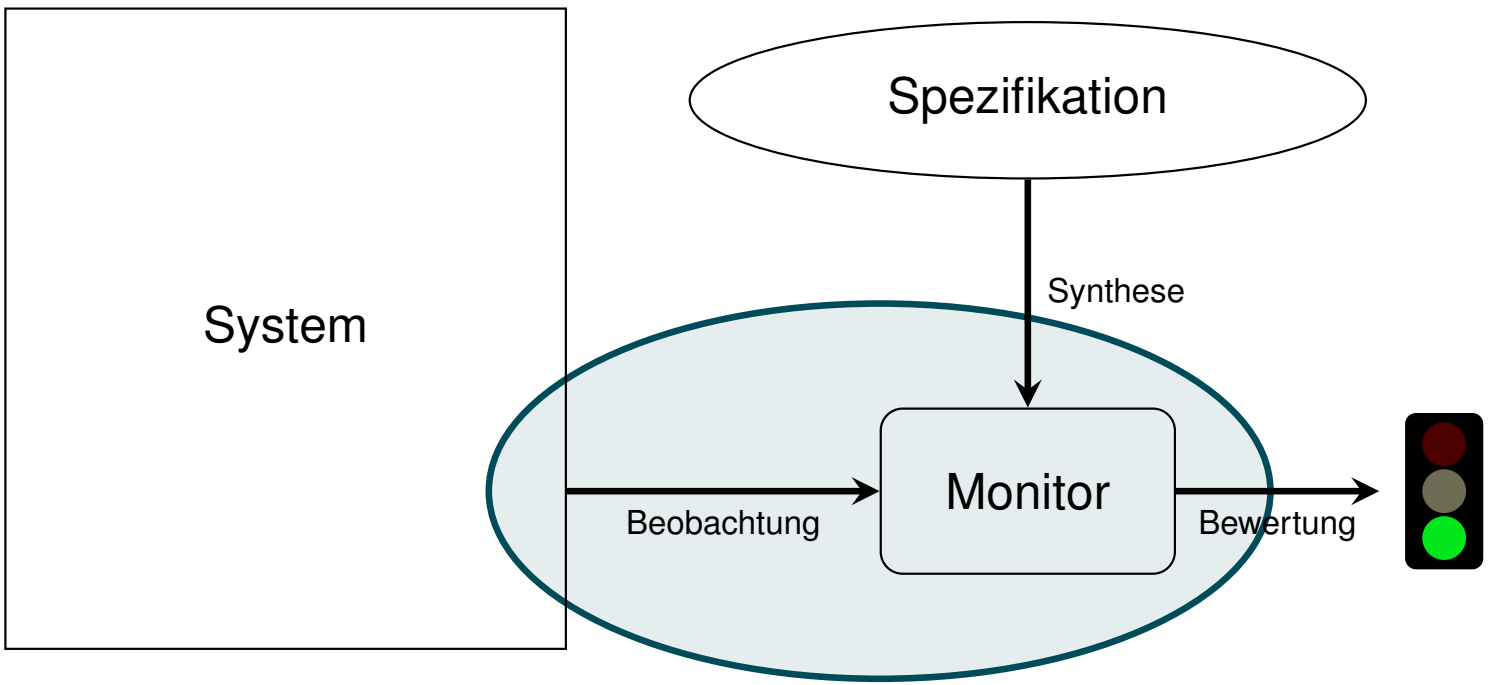
Laufzeitverifikation



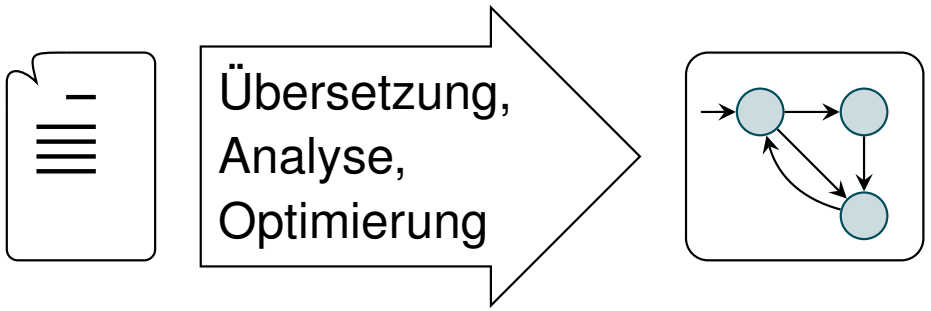
Laufzeitverifikation



Laufzeitverifikation



Beobachtung und Ausführung



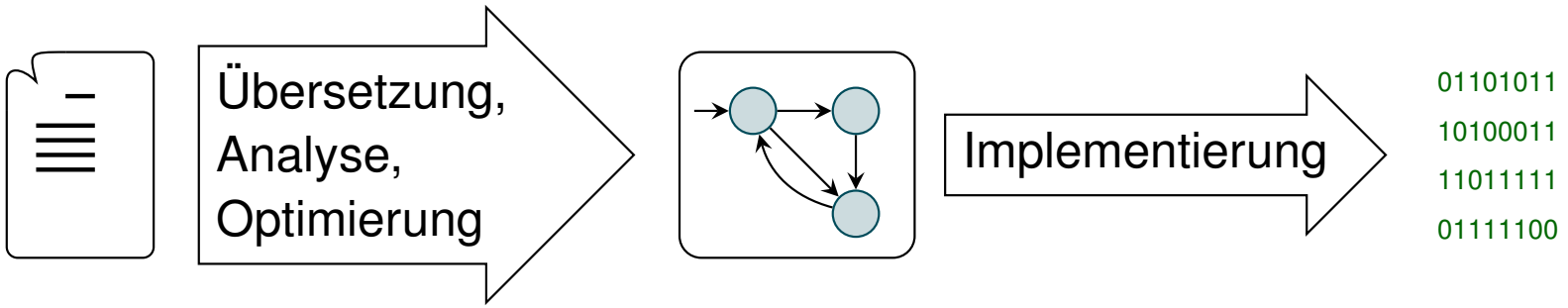
Spezifikation

(deklarativ)

Monitor

(operational, mit Ausgabe)

Beobachtung und Ausführung



Spezifikation

(deklarativ)

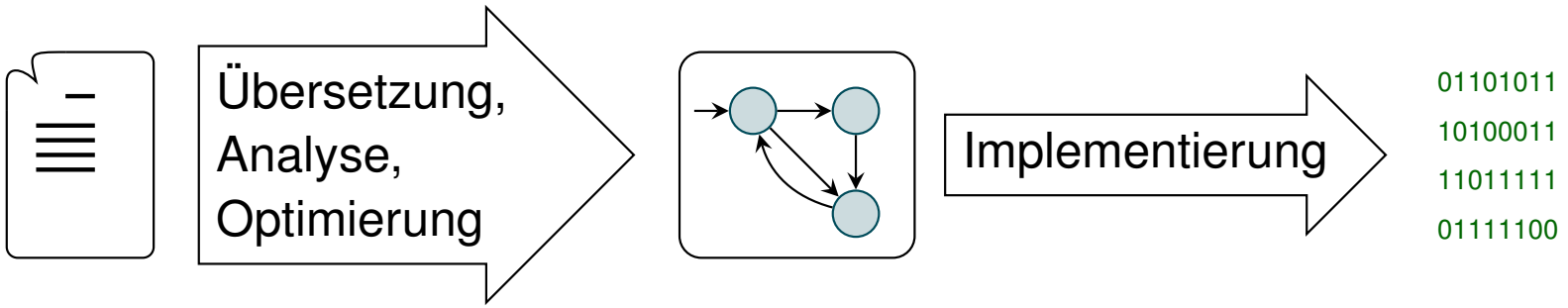
Monitor

(operational, mit Ausgabe)

Programmcode

(ausführbar)

Beobachtung und Ausführung

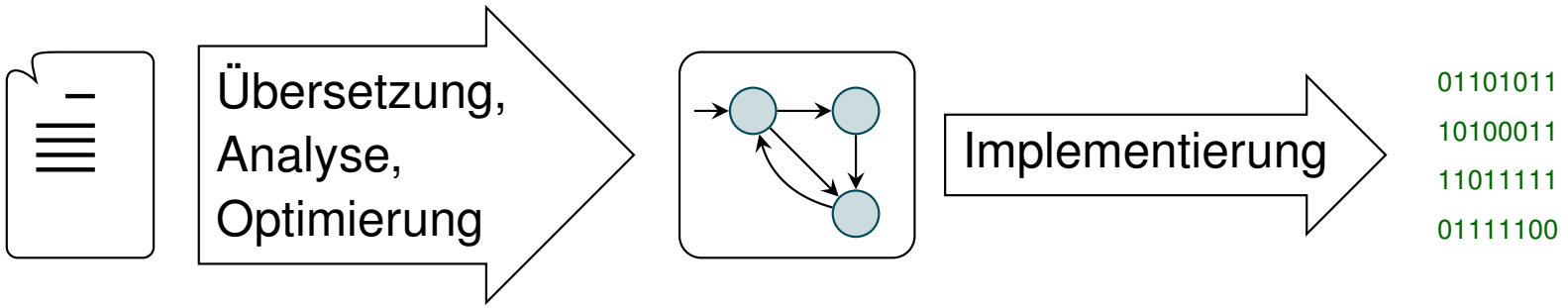


Spezifikation (deklarativ) | Monitor (operational, mit Ausgabe) | Programmcode (ausführbar)

Beobachtung des Systems

- Ausgabe oder Kommunikation
- (interne) Systemereignisse, -zustände (z. B. Funktionsaufrufe, Speicherzugriffe)
  - Trace-Schnittstellen
  - Instrumentierung des Programmcodes

Beobachtung und Ausführung

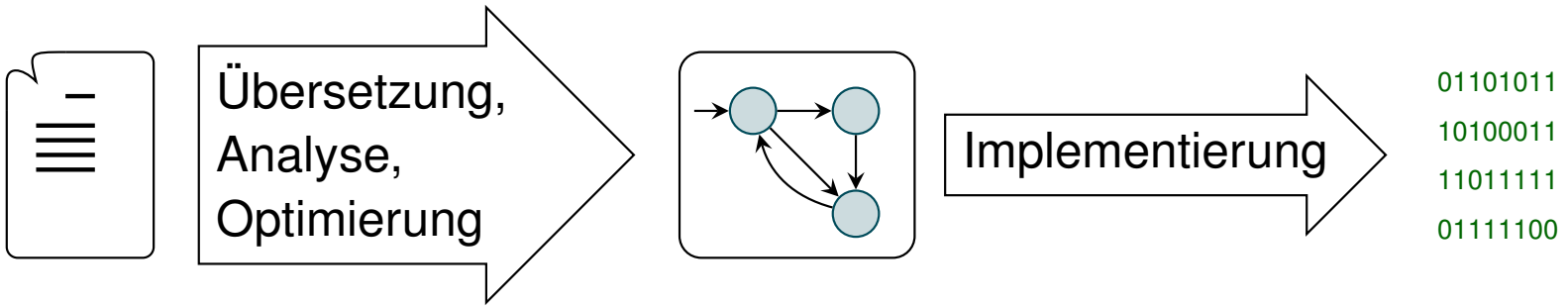


Spezifikation (deklarativ) | Monitor (operational, mit Ausgabe) | Programmcode (ausführbar)

Beobachtung des Systems

- Ausgabe oder Kommunikation
- (interne) Systemereignisse, -zustände (z. B. Funktionsaufrufe, Speicherzugriffe)
  - Trace-Schnittstellen
  - Instrumentierung des Programmcodes

Beobachtung und Ausführung

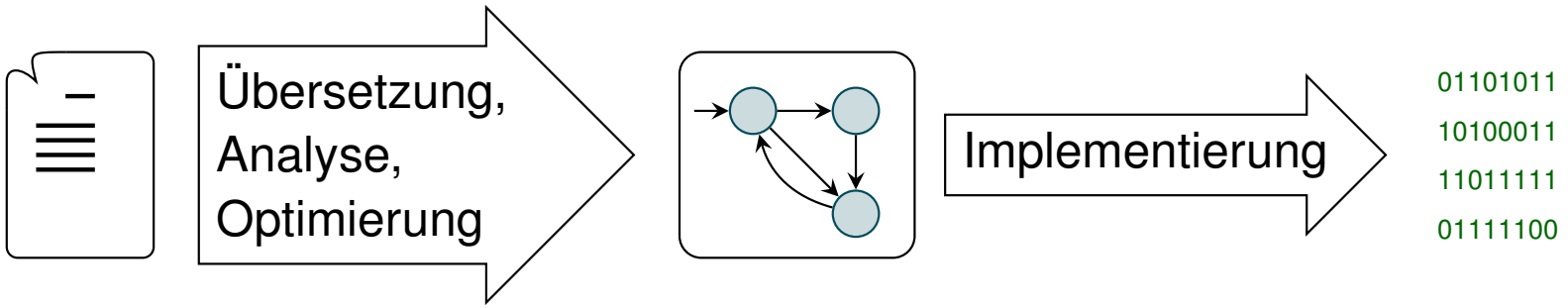


Spezifikation (deklarativ) | Monitor (operational, mit Ausgabe) | Programmcode (ausführbar)

Beobachtung des Systems

- Ausgabe oder Kommunikation
- (interne) Systemereignisse, -zustände (z. B. Funktionsaufrufe, Speicherzugriffe)
  - Trace-Schnittstellen
  - Instrumentierung des Programmcodes

Beobachtung und Ausführung



Spezifikation (deklarativ)

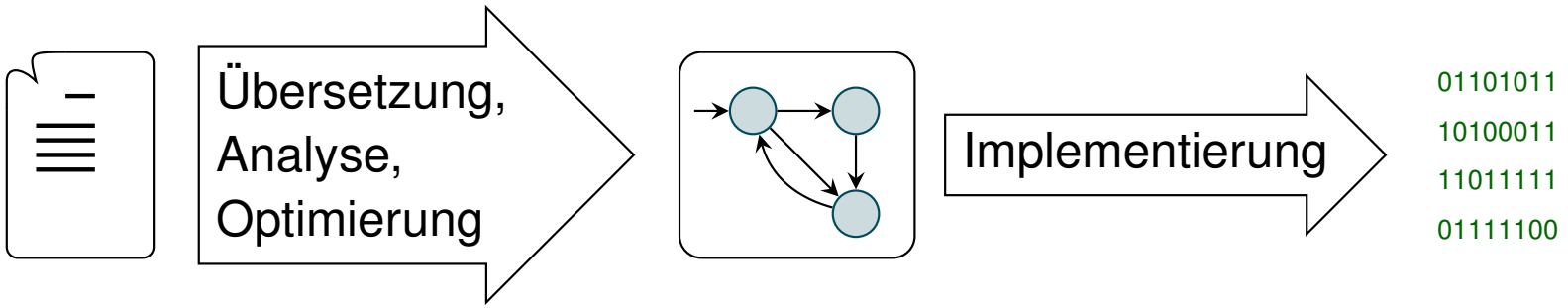
Monitor (operational, mit Ausgabe)

Programmcode (ausführbar)

Ausführung des Monitors

- im gleichen Prozess (z. B. bei Code-Instrumentierung)
- als separate Komponente (z. B. als Dienst in einer SOA)
- auf dedizierter Hardware (z. B. FPGA)

Beobachtung und Ausführung



Spezifikation (deklarativ)

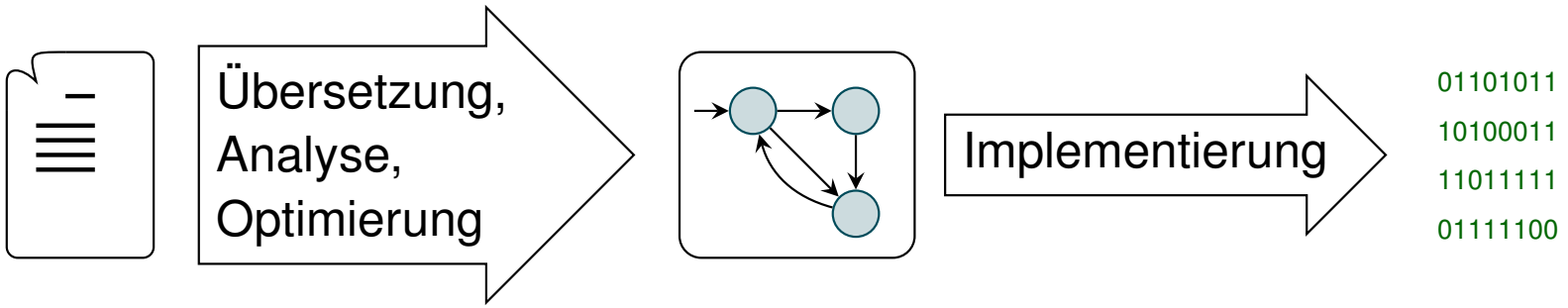
Monitor (operational, mit Ausgabe)

Programmcode (ausführbar)

Ausführung des Monitors

- im gleichen Prozess (z. B. bei Code-Instrumentierung)
- als separate Komponente (z. B. als Dienst in einer SOA)
- auf dedizierter Hardware (z. B. FPGA)

Beobachtung und Ausführung



Spezifikation (deklarativ)

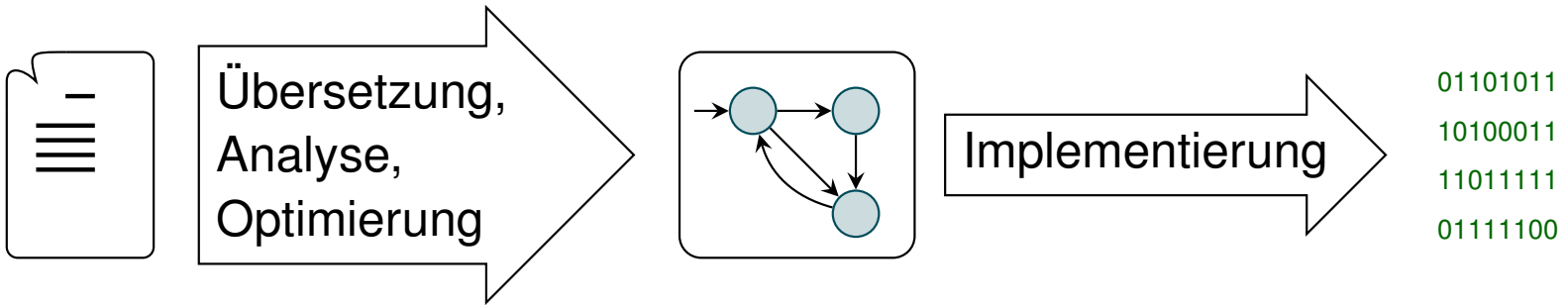
Monitor (operational, mit Ausgabe)

Programmcode (ausführbar)

Ausführung des Monitors

- im gleichen Prozess (z. B. bei Code-Instrumentierung)
- als separate Komponente (z. B. als Dienst in einer SOA)
- auf dedizierter Hardware (z. B. FPGA)

Beobachtung und Ausführung



Spezifikation (deklarativ)

Monitor (operational, mit Ausgabe)

Programmcode (ausführbar)

Ausführung des Monitors

- im gleichen Prozess (z. B. bei Code-Instrumentierung)
- als separate Komponente (z. B. als Dienst in einer SOA)
- auf dedizierter Hardware (z. B. FPGA)

Anwendung: Software

- Überwachung der Kommunikation von Diensten
- Analyse von Logfiles
- Unit Testing: jUnit^{RV}

Anwendung: Software

- Überwachung der Kommunikation von Diensten
- Analyse von Logfiles
- Unit Testing: jUnit^{RV}

Anwendung: Testen mit jUnit^{RV}

Example (Instrumentierung: Definition von Ereignissen)

```
String classUnderTest = "myPackage.classUnderTest";
private static Event start = called(classUnderTest, "startTask");
private static Event initok = returned(classUnderTest, "initialize");
private static Event end = called(classUnderTest, "endTask");
```

- Definition der Ereignisse $\Rightarrow$ Beobachtung
- Weitere Beispiele
  - „Variable $\mathrm{x} > 100$“,
  - „Zugriff auf Variable x“,
  - „Exception wurde geworfen“

Anwendung: Testen mit jUnit^{RV}

Example (Eigenschaft: Definition eines Monitors)

```
private static Monitor myMonitor1 = new Monitor( Always(a) );
```

- Komplexere Eigenschaften durch zeitliches und logisches Verknüpfen von Ereignissen
- „Auf *start* folgt irgendwann *end*“ und „Kein *start* vor *initok*“
- Always (*start* $\Rightarrow$ Eventually *end*) And ((Not *start*) Until *initok*)

Anwendung: Testen mit jUnit^{RV}

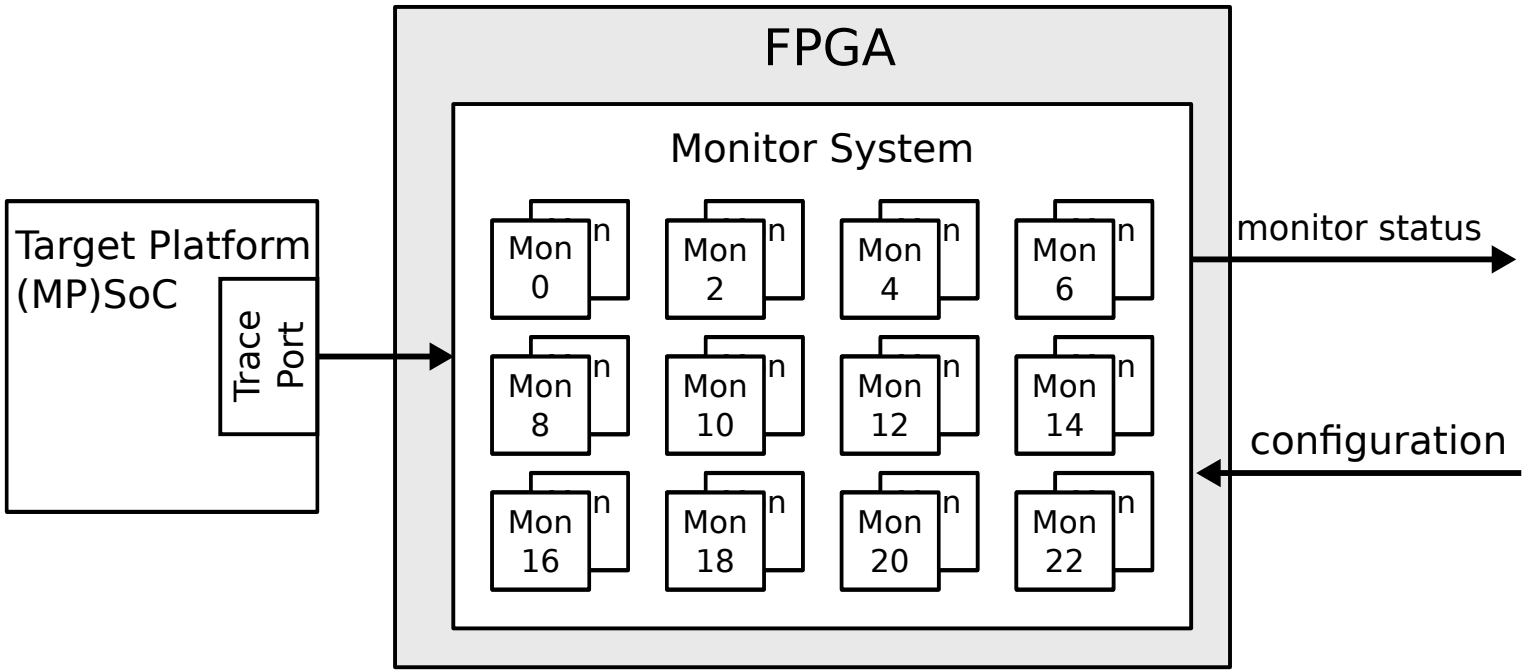
Example (Ausführung: Definition von Testfällen)

```
@RunWith(RVRunner.class)
public class MyTest {

    // Instrumentierung: Definition von Ereignissen
    ...
    // Eigenschaften: Definition der Monitore
    ...

    @Test
    @Monitors({"myMonitor1", "myMonitor2",...})
    public void test1() {
        ...
    }
}
```

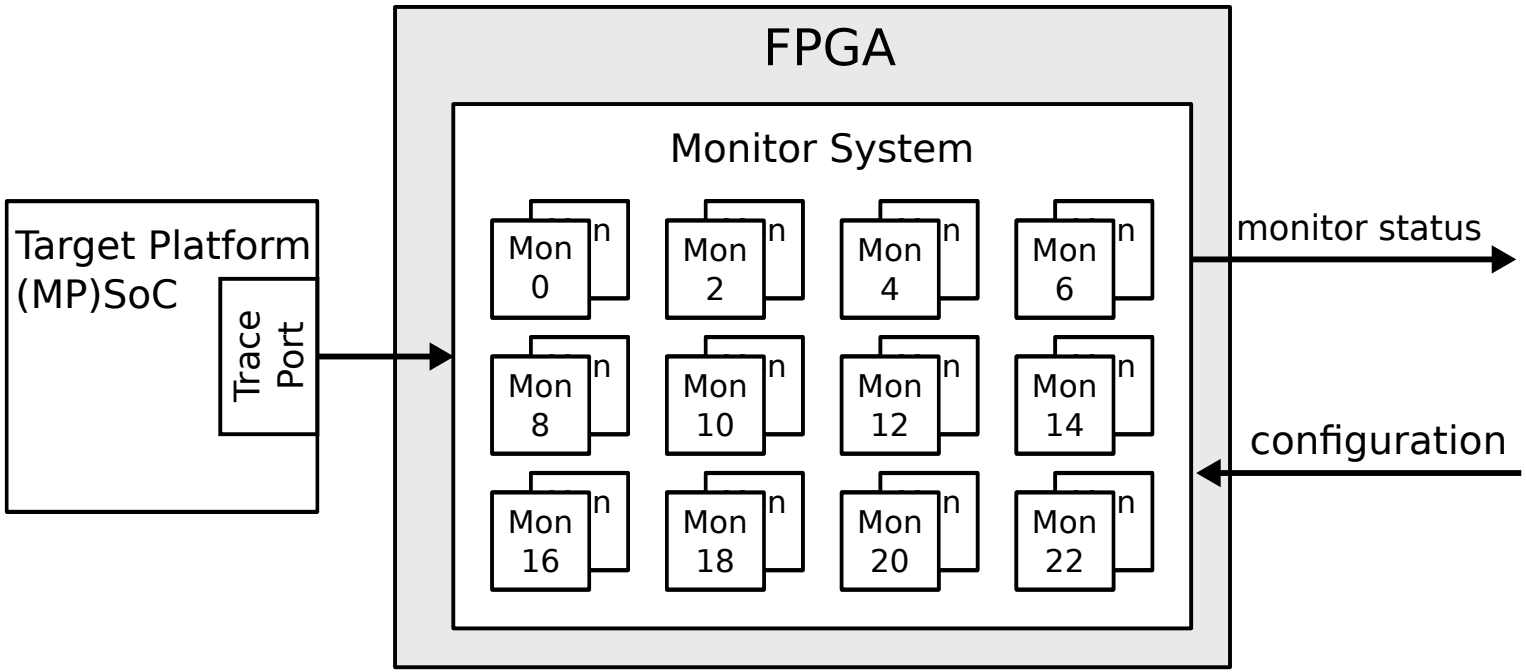
Laufzeitverifikation in Hardware



CONIRAS – Kontinuierliche nicht-intrusive Laufzeitanalyse von SoCs

- AbsInt, Accemic, TU Darmstadt (RS), U zu Lübeck (ISP)
- Abgriff von Trace-Daten, WCET-Analysen, Laufzeitverifikation

Laufzeitverifikation in Hardware

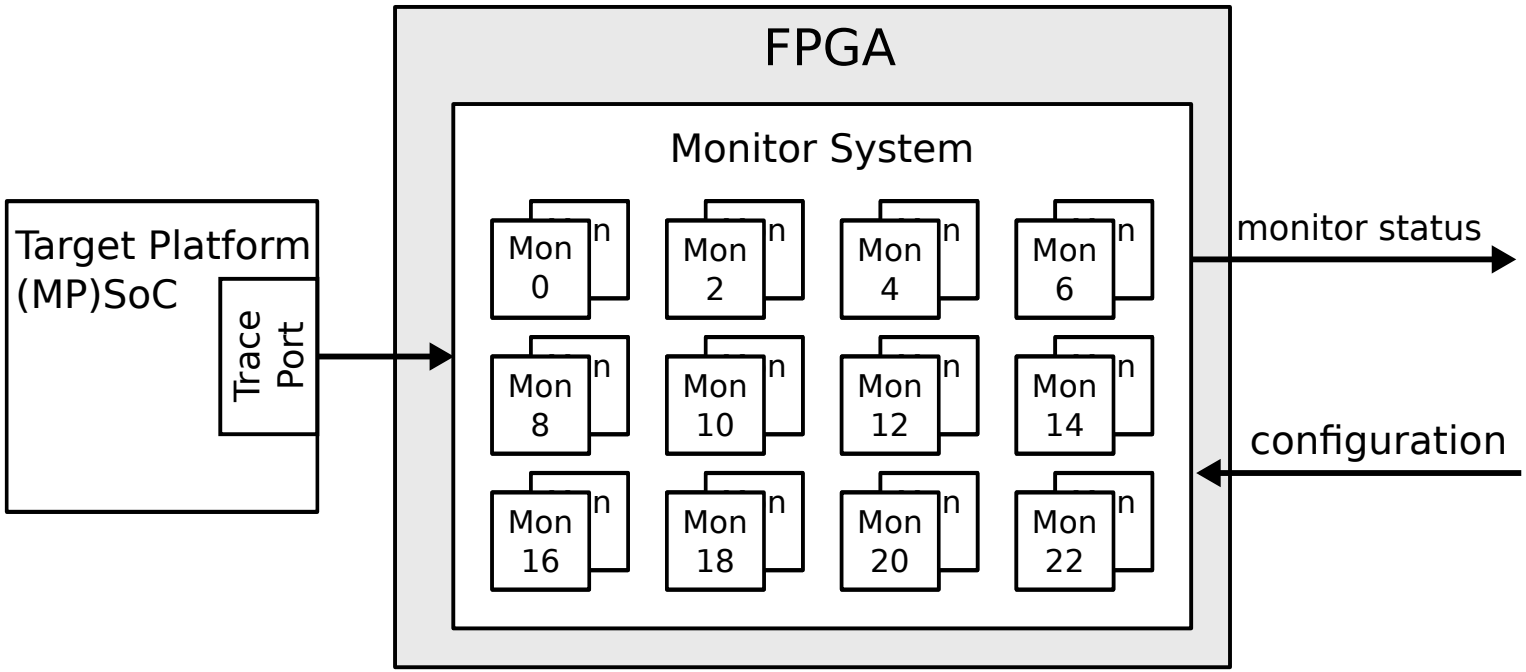


CONIRAS – Kontinuierliche nicht-intrusive Laufzeitanalyse von SoCs

- AbsInt, Accemic, TU Darmstadt (RS), U zu Lübeck (ISP)
- Akquise von Trace-Daten, WCET-Analyse, Laufzeitverifikation

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Laufzeitverifikation in Hardware

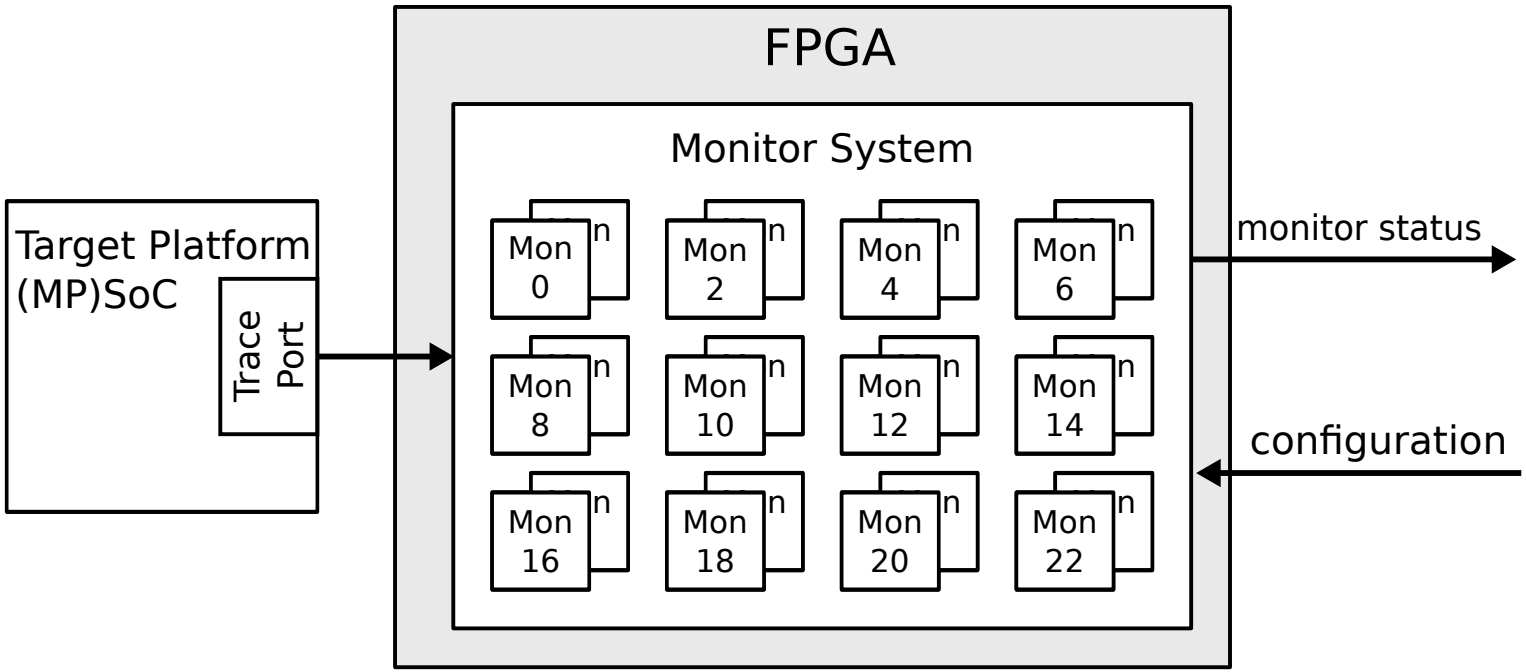


CONIRAS – Kontinuierliche nicht-intrusive Laufzeitanalyse von SoCs

- AbsInt, Accemic, TU Darmstadt (RS), U zu Lübeck (ISP)
- Akquise von Trace-Daten, WCET-Analyse, Laufzeitverifikation

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Laufzeitverifikation in Hardware

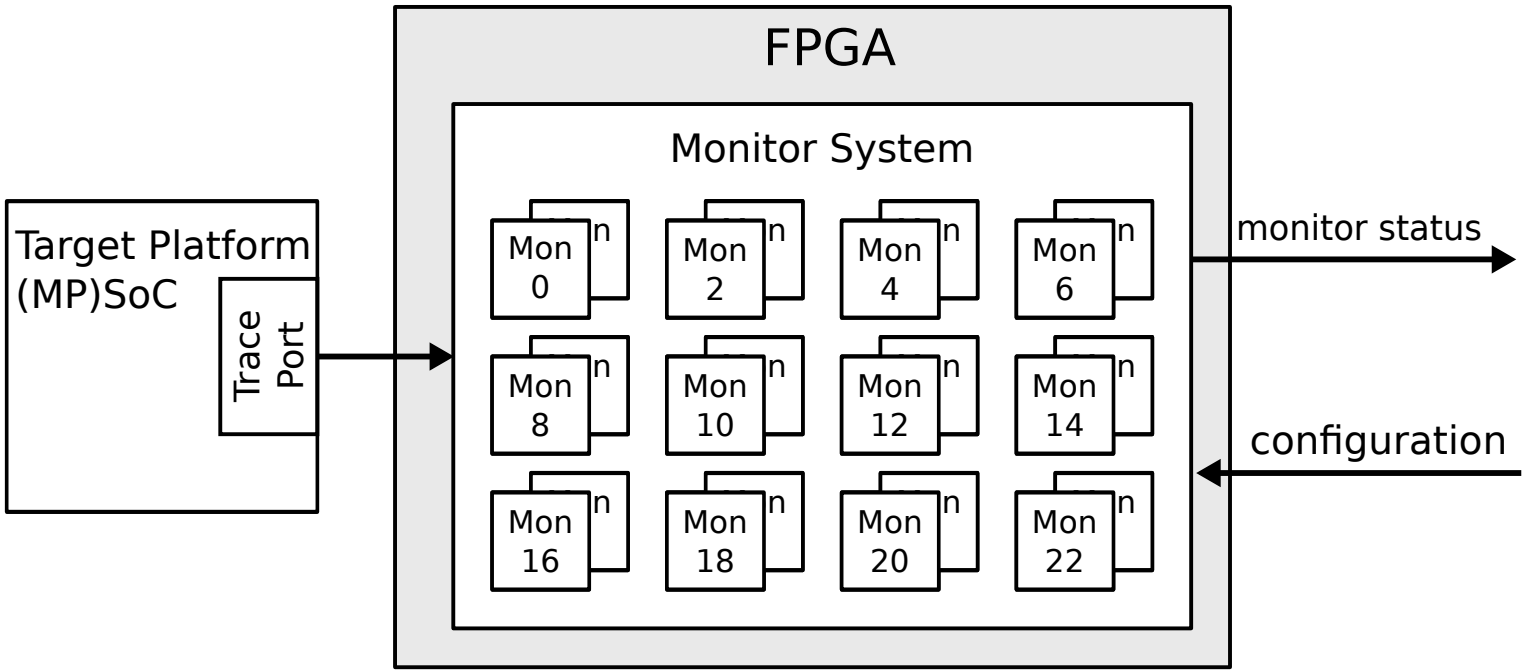


CONIRAS – Kontinuierliche nicht-intrusive Laufzeitanalyse von SoCs

- AbsInt, Accemic, TU Darmstadt (RS), U zu Lübeck (ISP)
- Akquise von Trace-Daten, WCET-Analyse, Laufzeitverifikation

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Laufzeitverifikation in Hardware



CONIRAS – Kontinuierliche nicht-intrusive Laufzeitanalyse von SoCs

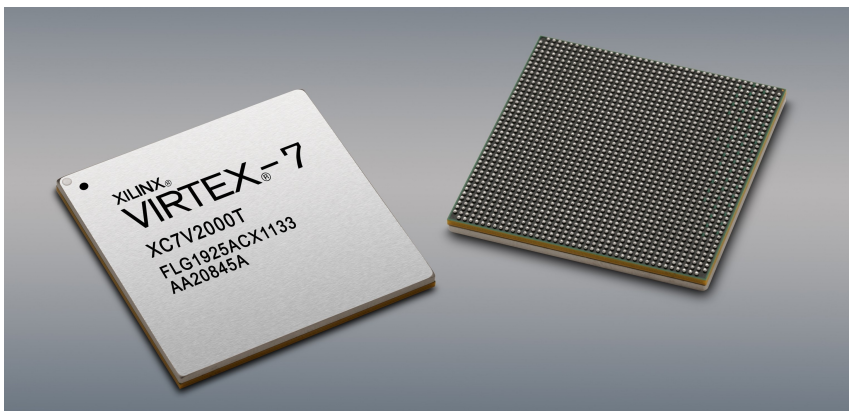
- AbsInt, Accemic, TU Darmstadt (RS), U zu Lübeck (ISP)
- Akquise von Trace-Daten, WCET-Analyse,

  Laufzeitverifikation

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Vorteile von FPGAs

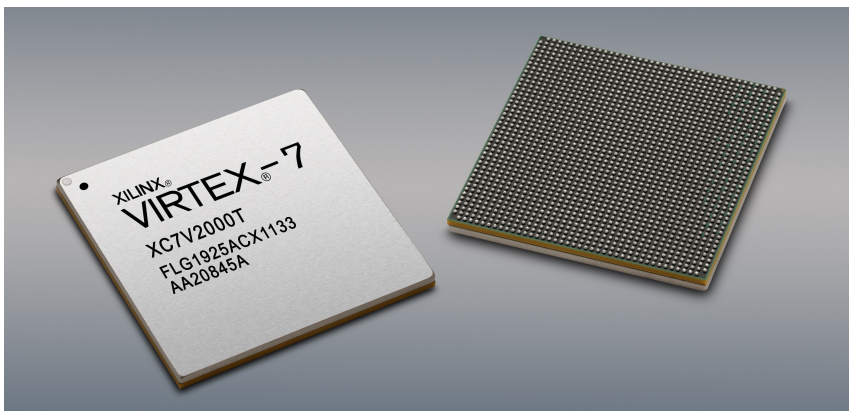
- **Schnelle und parallele Verarbeitung von Daten**
- Extrem viele Freiheitsgrade
- Beliebig oft neu konfigurierbar



www.xilinx.com

Vorteile von FPGAs

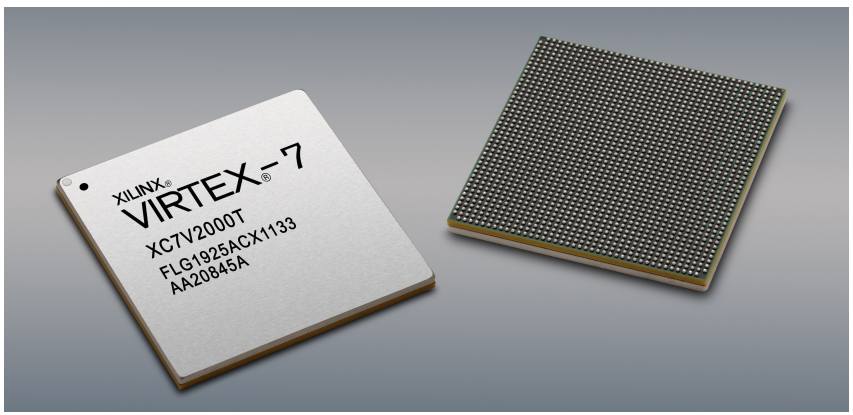
- Schnelle und parallele Verarbeitung von Daten
- **Extrem viele Freiheitsgrade**
- Beliebig oft neu konfigurierbar



www.xilinx.com

Vorteile von FPGAs

- Schnelle und parallele Verarbeitung von Daten
- Extrem viele Freiheitsgrade
- **Beliebig oft neu konfigurierbar**



www.xilinx.com

Probleme bei der Verwendung von FPGAs

- **Entwicklungstools sind schlecht zu integrieren**
- Hardwarebeschreibung nicht intuitiv
- Synthese von Monitoren in Hardware dauert lange

Probleme bei der Verwendung von FPGAs

- Entwicklungstools sind schlecht zu integrieren
- **Hardwarebeschreibung nicht intuitiv**
- Synthese von Monitoren in Hardware dauert lange

Probleme bei der Verwendung von FPGAs

- Entwicklungstools sind schlecht zu integrieren
- Hardwarebeschreibung nicht intuitiv
- **Synthese von Monitoren in Hardware dauert lange**

Ansatz: Generische Monitore in Hardware

- **Hardware (für Monitore) wird nur ein mal Synthetisiert**
- Reduzierung der Freiheitsgrade durch Parameter
- Parameter lassen sich nachträglich ohne erneute Synthese ändern

Ansatz: Generische Monitore in Hardware

- Hardware (für Monitore) wird nur ein mal Synthetisiert
- **Reduzierung der Freiheitsgrade durch Parameter**
- Parameter lassen sich nachträglich ohne erneute Synthese ändern

Ansatz: Generische Monitore in Hardware

- Hardware (für Monitore) wird nur ein mal Synthetisiert
- Reduzierung der Freiheitsgrade durch Parameter
- **Parameter lassen sich nachträglich ohne erneute Synthese ändern**

Monitore in Hardware

- Einmalig (dauert lange):
  - Beschreibung des Monitorsystems
  - Syntheseprozess
- Beliebig oft (geht schnell):
  - Anpassen von Parametern
  - Anpassen des Verhaltens (Automaten)

Monitore in Hardware

- Einmalig (dauert lange):
  - **Beschreibung des Monitorsystems**
  - Syntheseprozess
- Beliebig oft (geht schnell):
  - Anpassen von Parametern
  - Anpassen des Verhaltens (Automaten)

Monitore in Hardware

- Einmalig (dauert lange):
  - Beschreibung des Monitorsystems
  - **Syntheseprozess**
- Beliebig oft (geht schnell):
  - Anpassen von Parametern
  - Anpassen des Verhaltens (Automaten)

Monitore in Hardware

- Einmalig (dauert lange):
  - Beschreibung des Monitorsystems
  - Syntheseprozess
- Beliebig oft (geht schnell):
  - Anpassen von Parametern
  - Anpassen des Verhaltens (Automaten)

Monitore in Hardware

- Einmalig (dauert lange):
  - Beschreibung des Monitorsystems
  - Syntheseprozess
- Beliebig oft (geht schnell):
  - **Anpassen von Parametern**
  - Anpassen des Verhaltens (Automaten)

Monitore in Hardware

- Einmalig (dauert lange):
  - Beschreibung des Monitorsystems
  - Syntheseprozess
- Beliebig oft (geht schnell):
  - Anpassen von Parametern
  - **Anpassen des Verhaltens (Automaten)**

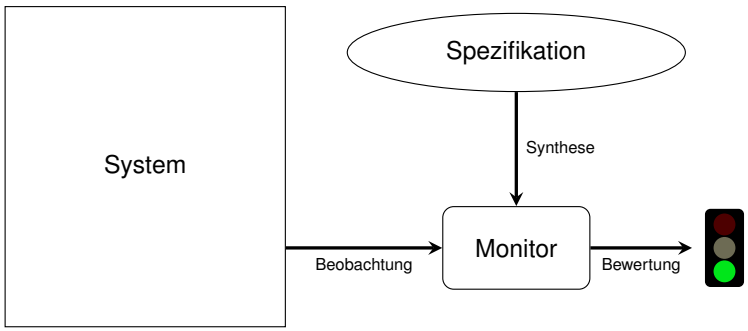
Vorteile des Prinzips

- **Automatische Überwachung von Laufzeiteigenschaften**
- Parallele Ausführung vieler Beobachtungsaufgaben

Vorteile des Prinzips

- Automatische Überwachung von Laufzeiteigenschaften
- **Parallele Ausführung vieler Beobachtungsaufgaben**

Zusammenfassung




```
private static Monitor myMonitor =
  new Monitor( Always(a) );
```