Eingebettete Systeme und ihre Anwendungen

TU Darmstadt

Stefan Wittmann

GENERIERUNG VON GEPIPELINTEN HARDWARE-RECHENEINHEITEN AUS KONTROLLFLUßGRAPHEN

AGENDA

- Aufgabenstellung und Zielsetzung
- Systemarchitektur
- Eingabeformat und Ausgabeformat
- Der Transformationsvorgang
- Einblick: Die Frameworks
- Zukünftige Aufgaben

AUFGABENSTELLUNG

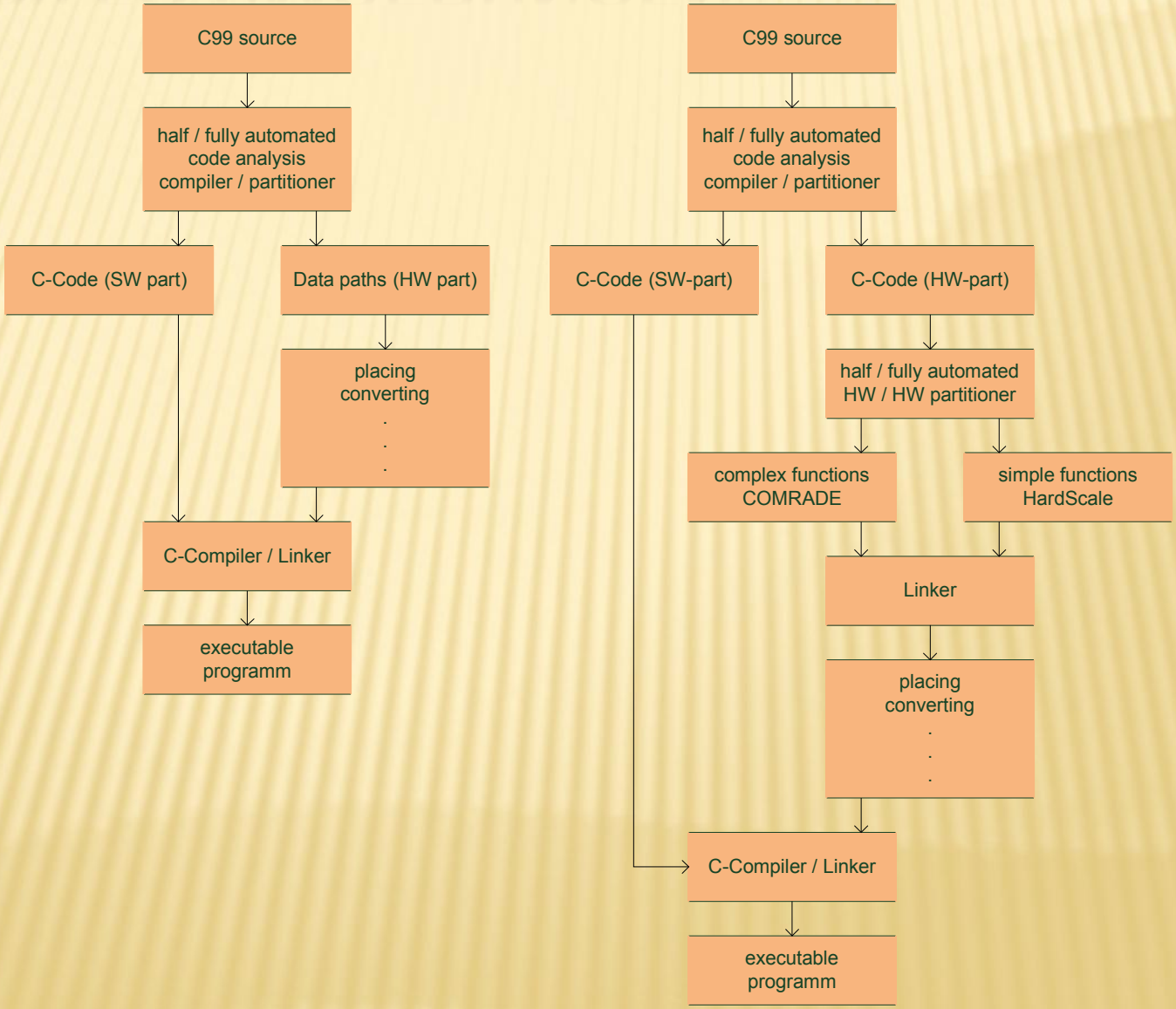
- Evaluierung bestehender Compilerframeworks
- Implementierung eines Hardwarecompilers von C99 nach Verilog für adaptive Computer Systeme auf Basis des ausgewählten Compilers

ZIELSETZUNG

- Aufbau eines „leichten“ Hardwarecompilers
- Umsetzung des gesamten Umfang der Eingabesprache*
- Umsetzung von sogenannten HotSpots zum leichten Erweitern des Compilers
- Eingliedern von HardScale in den Compileprozess von COMRADE

*mit einigen Einschränkungen

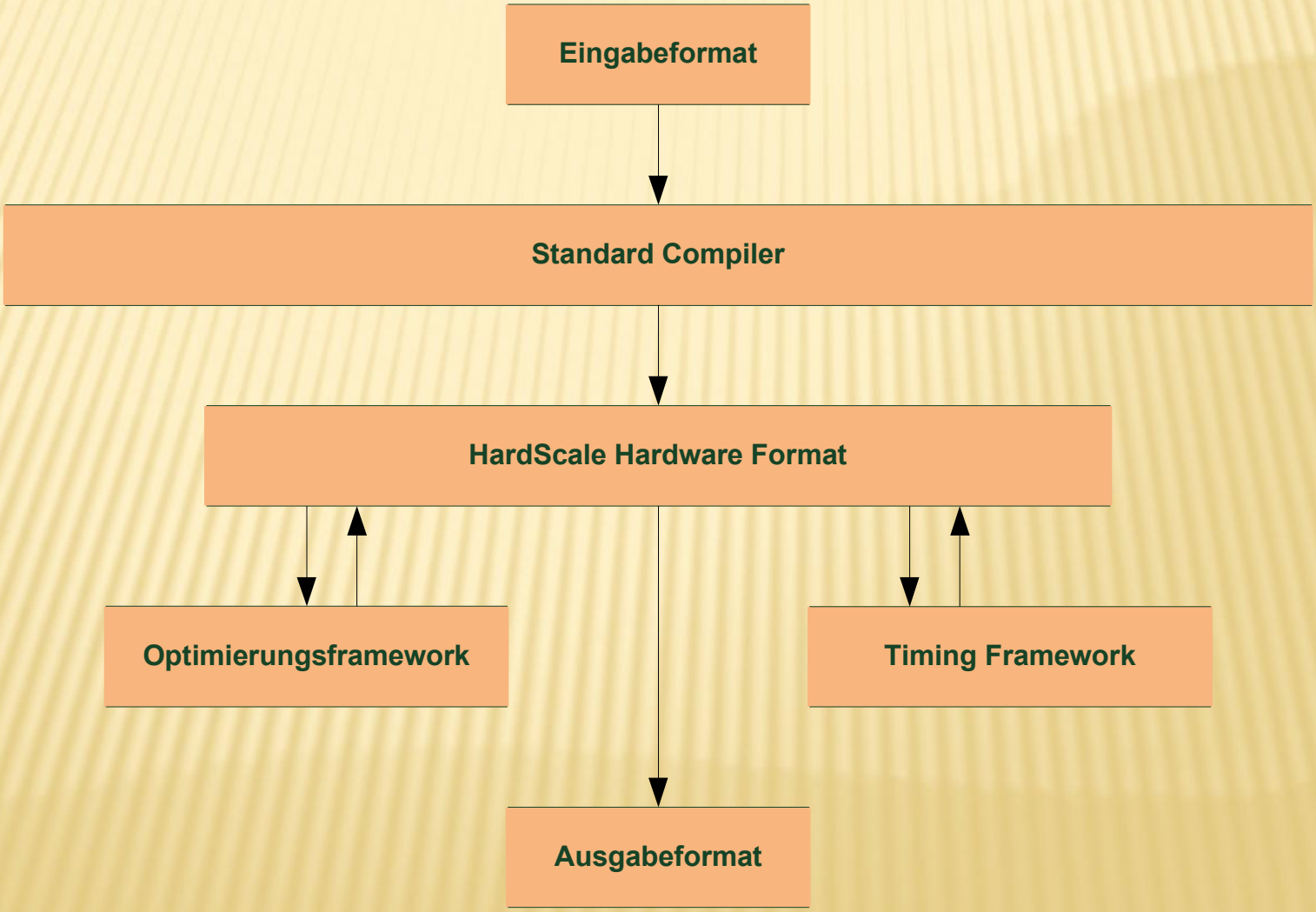
COMPILER UMGEBUNGEN

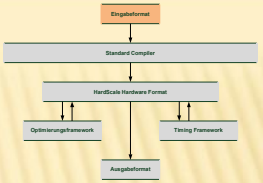


FOCUS DES COMPILERS

- Straight-line Code
- If-Then-Else Konstrukte
- Speicherzugriffe
- Software Calls
- Schleifen

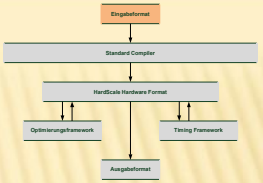
SYSTEMARCHITEKTUR





EINGABEFORMAT - ANNAHMEN

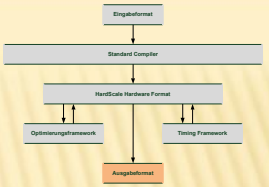
- C99 TC3 Standard
  - Erste Methode in jeder Quelldatei wird kompiliert
    - Beliebig viele Eingabeparameter
    - Ein Rückgabewert, in der Regel `structs`
    - SoftwareCalls werden über zusätzliche Methoden angesprochen



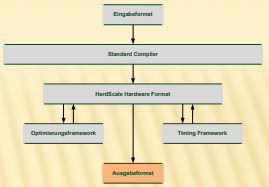
EINSCHRÄNKUNGEN C99

- Keine Sprünge
- Keine Zugriffe auf globale Variablen
- Alle Parameter, die vom Pointertyp sind müssen das Schlüsselwort `restrict` verwenden
- Switch-Statement leicht angepasst
  - Keine Breaks (Sprünge)
  - Jedes nichtleere Statement endet mit implizitem Break
  - Jedes leere Statement führt zu einem Fallthrough

AUSGABEFORMAT

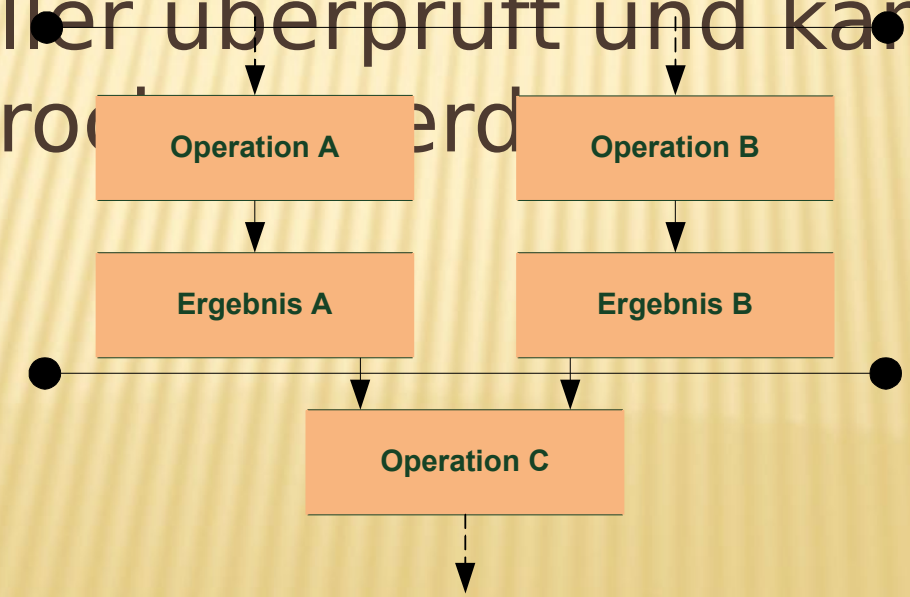


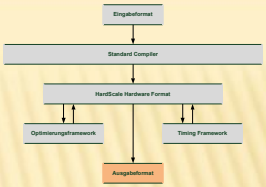
- Hardware wird durch Verilog Module dargestellt
- Werte werden in Schieberegistern gespeichert
- Operationen werden durch Module implementiert, die aus kombinatorischen Schaltungen bestehen
- Verwaltet wird das Rechenmodul durch einen Controller



KONTROLLE DES DATENPFADS

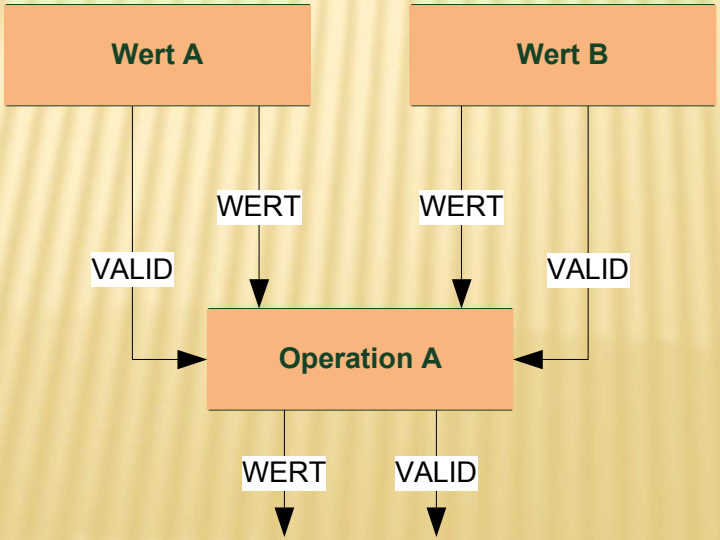
- Das Rechenmodul wird von außen angetriggert
- Danach kontrolliert sich der Datenpfad selbst, wird aber dennoch vom Controller überprüft und kann unterbrochen werden

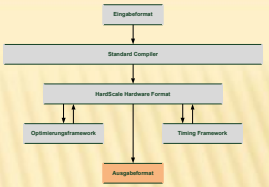




BEISPIELMODUL

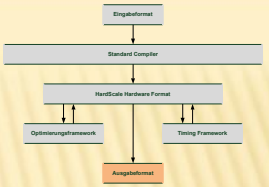
```
wire newdata_ADD_cycle2_3 = valid_a_s_6 & valid_b_s_5;
ADD #( .WA(32), .WB(32), .WR(32), .DEPTH(0), .SIGN(0))
       ADD_cycle2_3
       (.A(w_a_s_6), .B(w_b_s_5), .R(w_ADD_cycle2_3),
       .RESET(RESET), .CLK(CLK), .CE(CE),
       .NEWDATA(newdata_ADD_cycle2_3),
       .VALID(valid_ADD_cycle2_3));
```





DER CONTROLLER

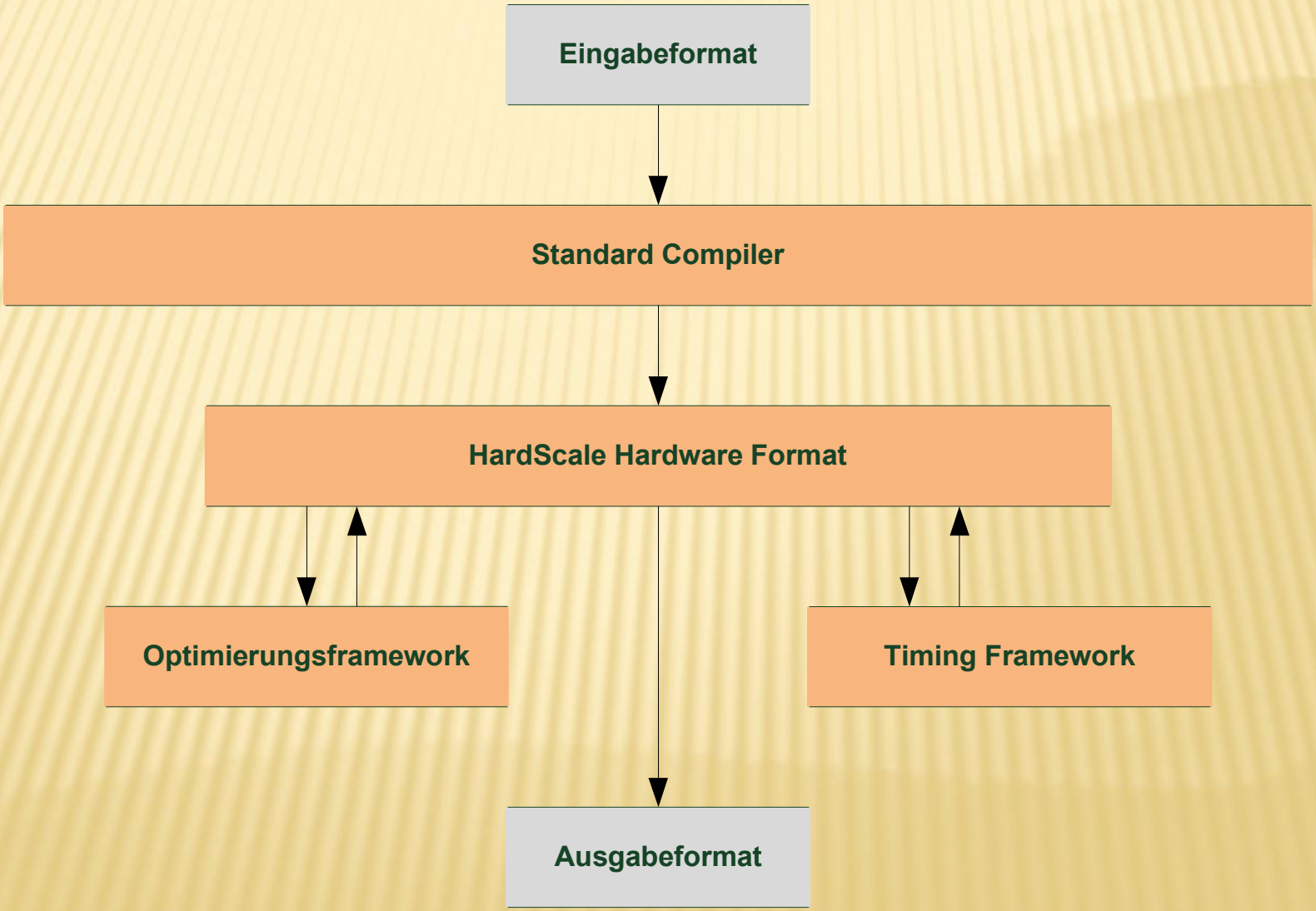
- Kontrolle durch Abfangen der `Valid` und `Newdata` Signale
- Speicherzugriffe erfordern das Anhalten des Datenpfades (sowohl Read als auch Write Anweisungen), bis Operation durchgeführt wurde
- Software Calls erfordern ebenso das Anhalten des Datenpfades, jedoch wird hier die Kontrolle komplett abgegeben
  - Wiedererlangen der Kontrolle durch erneutes Startsignal

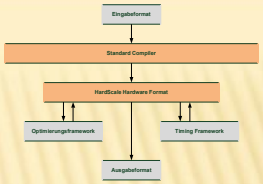


SPEZIELLE MODULE

- INREG – Laden der Parameterwerte
- OUTREG – Übertragen der Rückgabewerte
- IRQREG – Interruptwurf an die Umgebung mit entsprechendem Interruptwert

AKTUELLER STAND

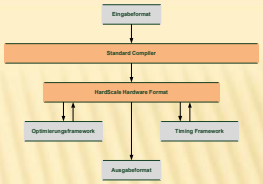


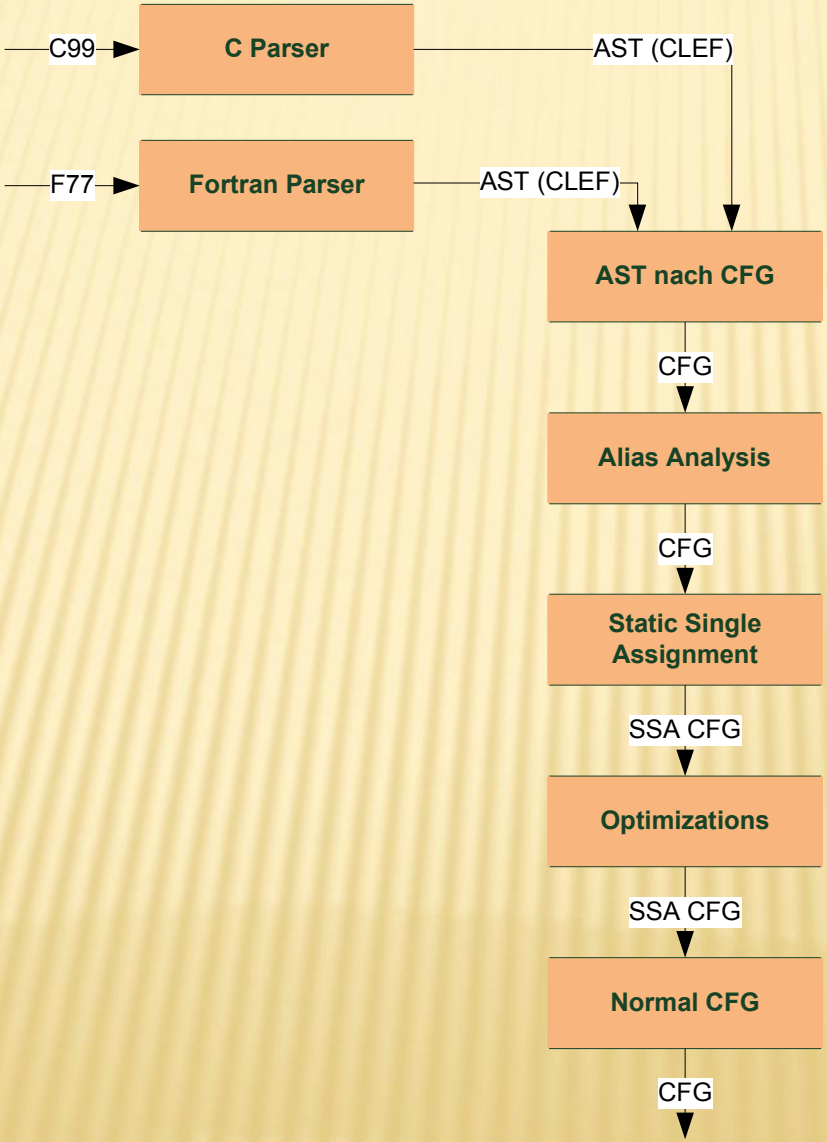


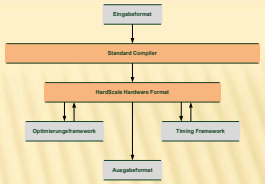
SCALE

- A Scalable Compiler for Analytical Experiments
- Compiler für C, C99, K&R, F77 und F90
- Stellt Eingabesprachen als AST und schließlich CFG dar.
- Bietet schon diverse Optimierungen, z.B.:
  - Copy Propagation (SSA)
  - Loop Unrolling (SSA)
  - Dead Code Elimination (AST)
  - Dead Variable Eliminiation (non-SSA CFG)

SCALE COMPILE FLOW



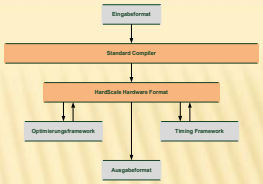




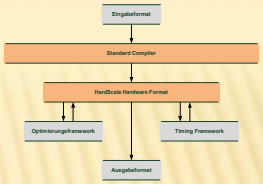
TRAVERSIERUNG DES AST UND CFG

- AST
  - Information der Methodendeklaration
    - Parameter
    - Rückgabetyp
- CFG
  - Informationen zum Code Ablauf innerhalb der Methode

CFG TRANSFORMATION



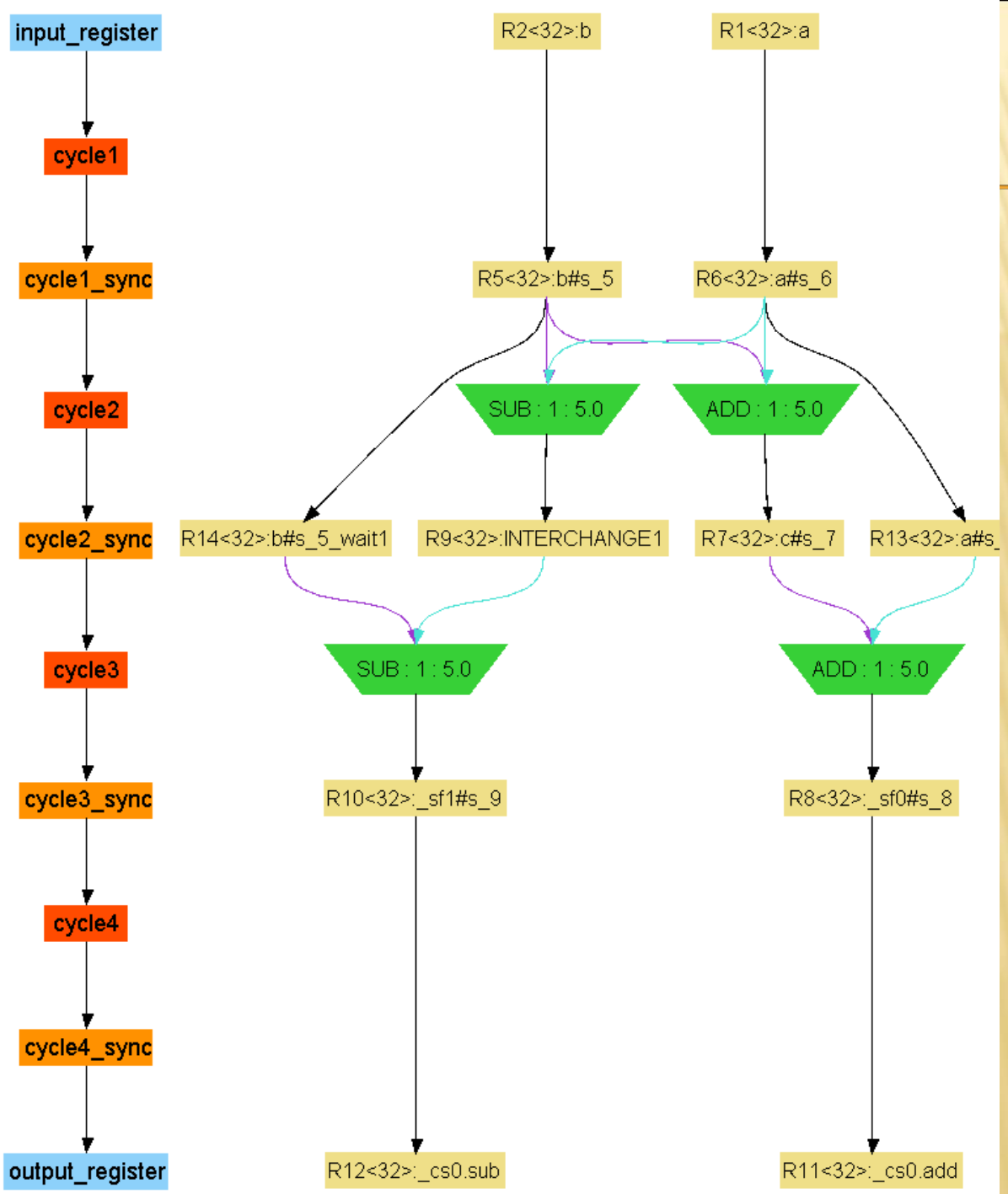
- Der CFG wird in eine Hardwaredarstellung überführt
  - Variablen werden zu Registern
  - Instruktionen werden zu Hardware Operationen und Leitungen (Wires) umgesetzt
  - Methodenaufrufe zu SoftwareCalls



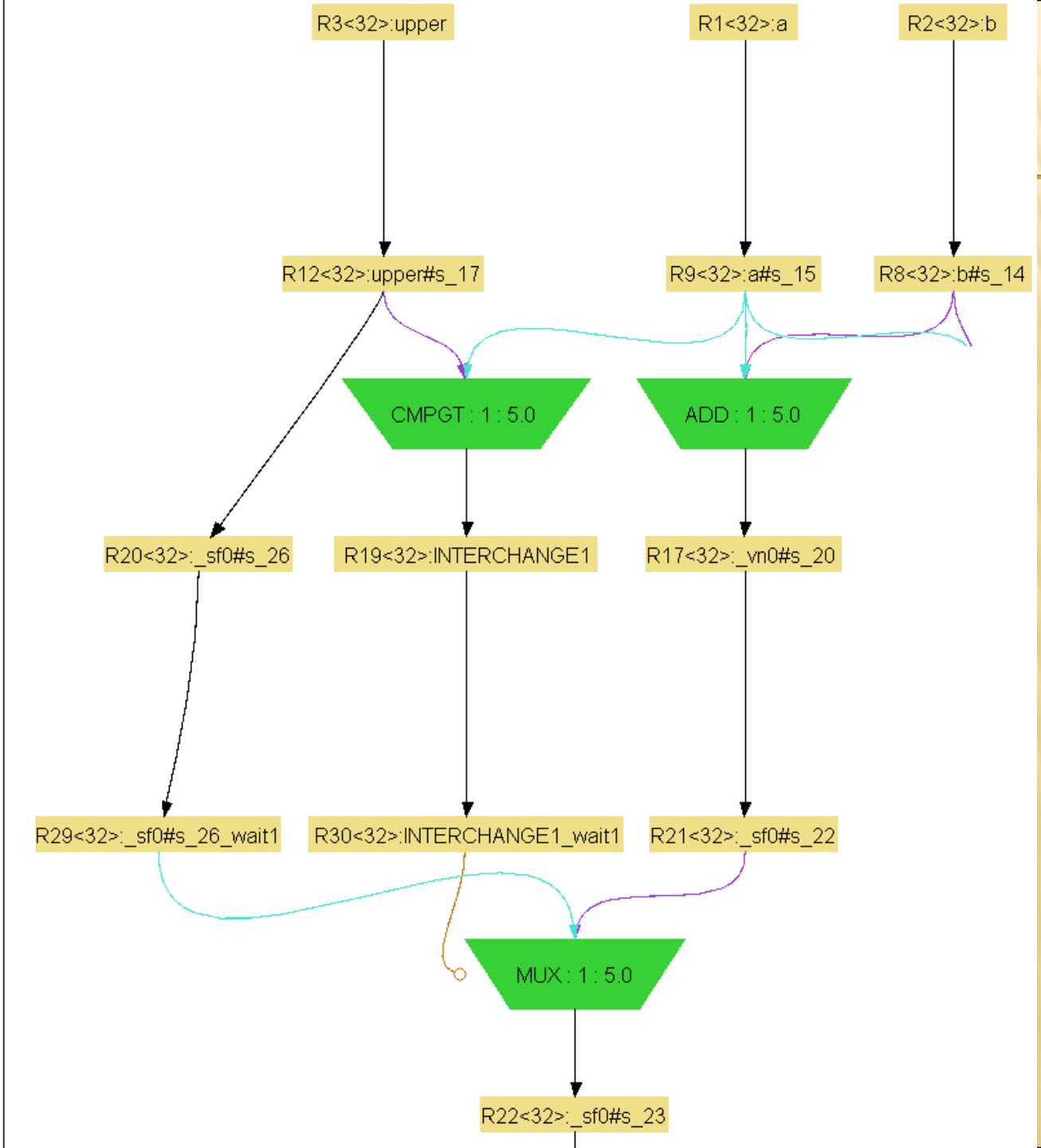
DARSTELLUNG DER HARDWARE

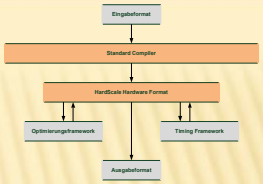
- Register
  - Speichern Werte
  - Verzögern bis zur nächsten steigenden Taktflanke
- Operationen
  - Sind kombinatorisch
  - Speichern Ergebnisse meistens in Register (Sync)
- Beide Elemente sind gegenseitig verlinkt

```
typedef struct
{
    int add;
    int sub;
} RESULT;

RESULT hwsumdiff(int a, int b) {
    RESULT result;

    int c       = a + b;
    result.add  = a + c;
    result.sub  = a - b - b;

    return result;
}
```



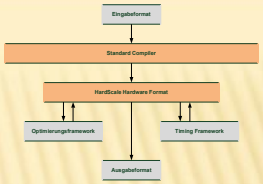
```
result.sum = a + b;

if ( a > upper)
    result.sum = upper;
```





WEITERE KONSTRUKTE

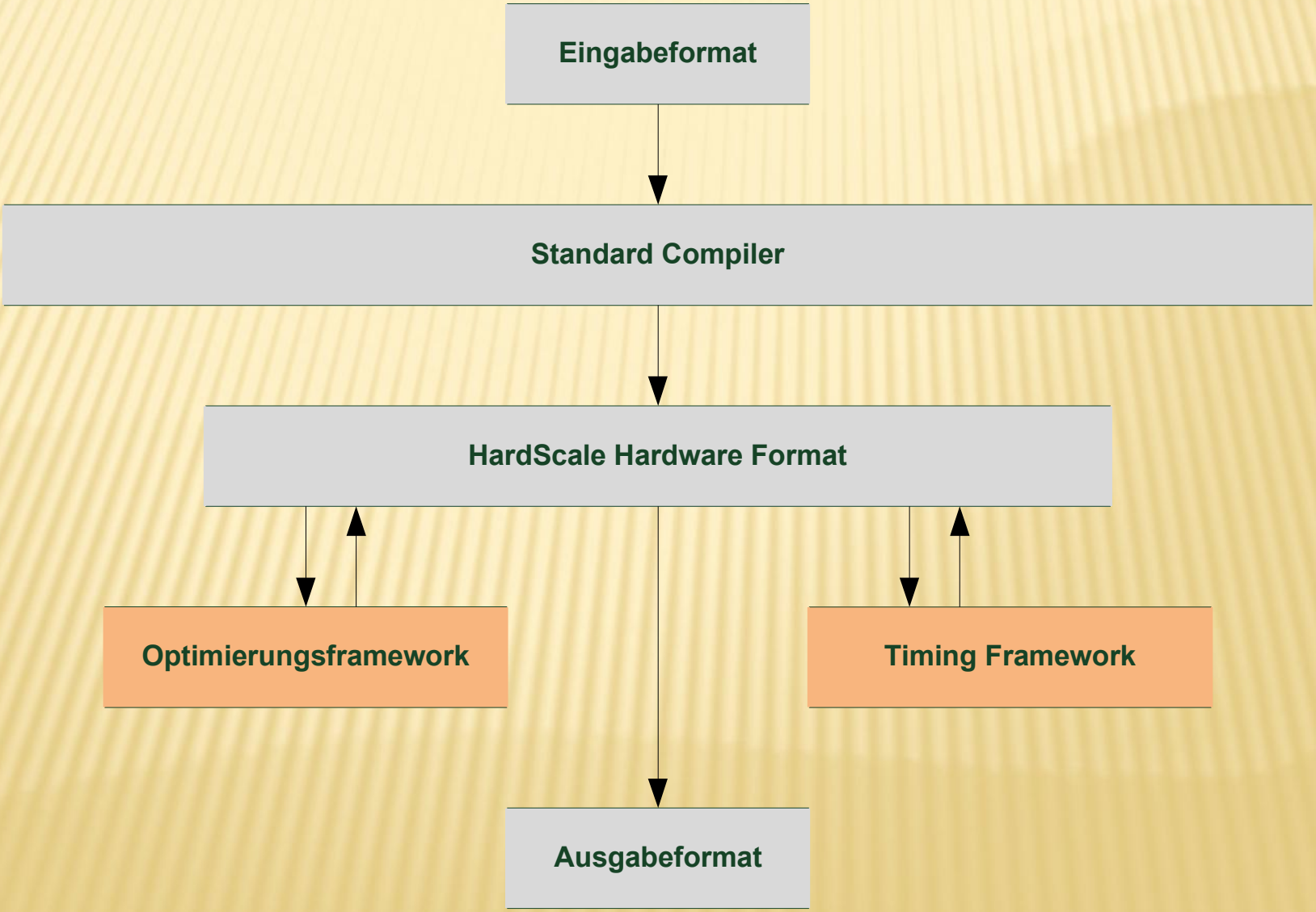
- Speicherzugriffe werden durch eigene Module dargestellt
  - Datenpfad muss solange angehalten werden, bis die Operation durchgeführt wurde
- SoftwareCalls werden durch einen Interrupt dargestellt

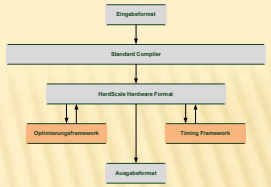


SCHLEIFEN

- Schleifen sind die einzigen Konstrukte, die gepipelint werden
- Erreicht wird dies durch die Einführung von sogenannten Unterschaltungen (Subcircuits)
- Jedes Untermodul ist einzeln aufrufbar und gepipelint
- Noch in der Entwicklung

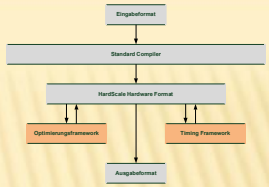
AKTUELLER STAND



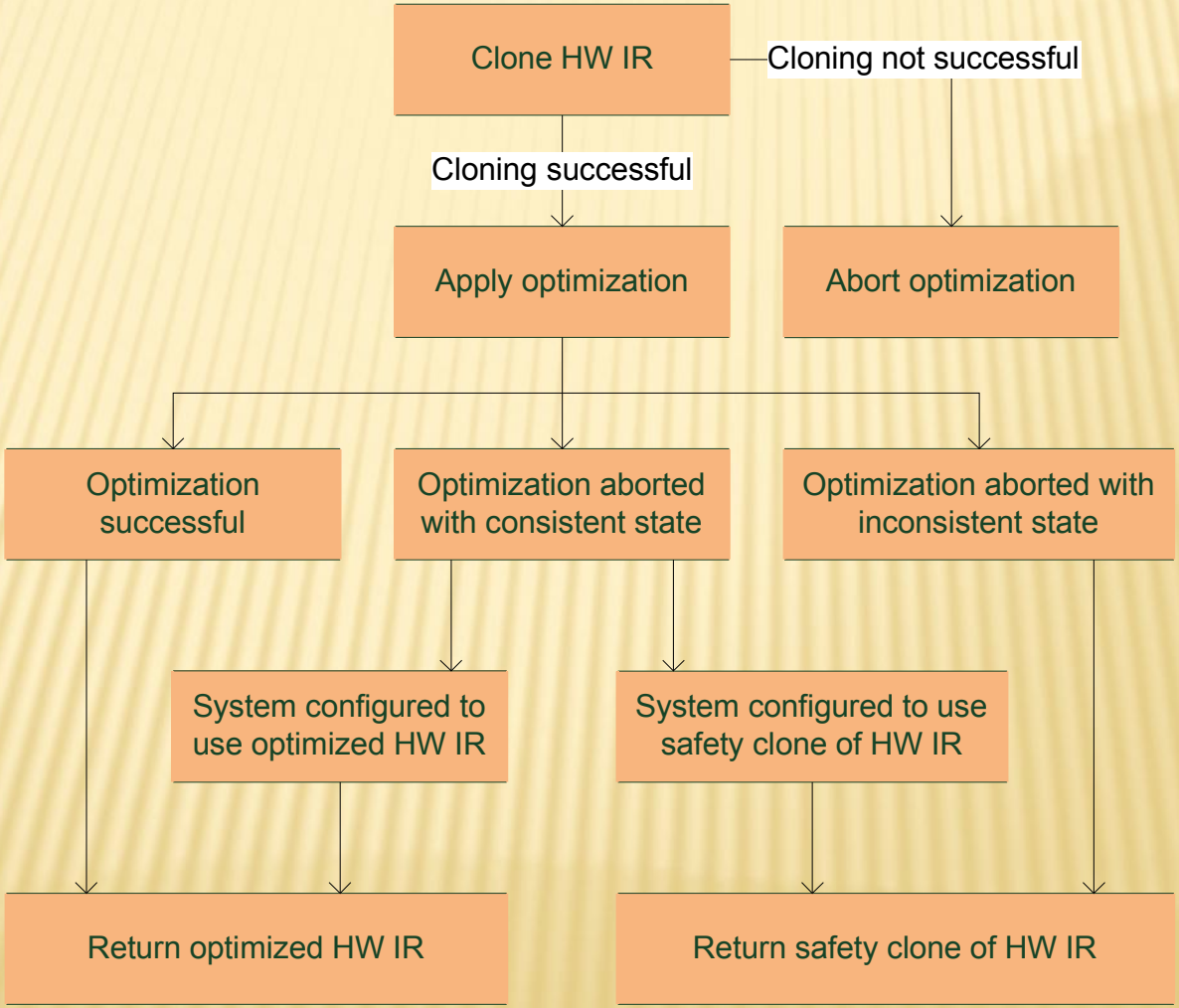


DAS TIMINGFRAMEWORK

- Liefert Latenzen und Ausführungszeiten
- Es sind auch multi-cycle Operationen möglich (Latenz > 1). Die Ausführungszeit bezieht sich auf die maximale Taktung eines Zyklus.
- Zur Zeit nur ein statisches Layer implementiert.



DAS OPTIMIERUNGSFRAMEWORK



ZUKÜNFTIGE AUFGABEN

- Timing Layer zu FLAME einbauen
- Weitere Optimierungen implementieren
- Floating-Point unterstützen
- Automatisches Chaining
- Einführen von HardScale in den aktuellen Compiledurchgang mit COMRADE

Danke für die Aufmerksamkeit