Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

HARDWARE / SOFTWARE-SCHNITTSTELLEN

Hardwareentwurf mit VHDL

23. Juni 2014
(Revision: 1341)

Prof. Dr. Steffen Reith

Theoretische Informatik
Studienbereich Angewandte Informatik
Hochschule **RheinMain**

PIPELINING

Notizen

Notizen

GRUNDLAGEN

Pipelining ist eine grundlegende Technik, um die **Performanz** eines Systemes zu **verbessern**.

- → Idee: Tasks (wenn möglich) zeitlich **überlappend auszuführen**
- → Idee: kombinatorische Schaltkreise in **Teilschritte aufteilen**
- → Idee: verwende **Register** um **Zwischenergebnisse** für den nächsten Schritt zu **speichern**

Hier gibt es zwei wichtige Kenngrößen:

- → Delay: **Zeit** die die **Bearbeitung eines Tasks** benötigt
- → Durchsatz: **Anzahl** der bearbeiteten **Tasks pro Zeiteinheit**

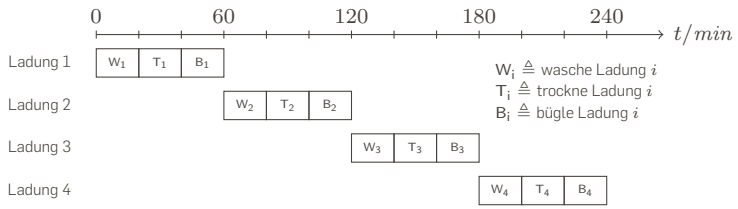
Pipelining **verbessert** den **Delay nicht** (ehr schlechter) und vergrößert (evtl.) den **Durchsatz**.

Notizen

EIN BEISPIEL: WASCHEN OHNE PIPELINING

In einer Wäscherei sind die Tasks „waschen“, „trocknen“ und „bügeln“ durchzuführen:

Ohne Pipeline



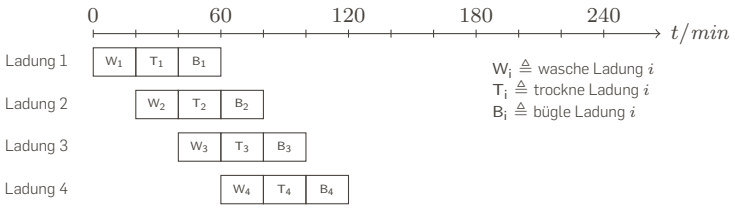
Damit ergibt sich:

- → **Delay**: 60 **Minuten** (Zeit für die Bearbeitung einer Ladung Wäsche)
- → **Durchsatz**: $4/(4 \cdot 3 \cdot 20) = 1/60$ **Ladungen pro Minute**

Notizen

EIN BEISPIEL: WASCHEN MIT PIPELINING

Mit Pipeline



Damit ergibt sich (idealisiert):

- → **Delay**: unverändert 60 **Minuten** (Zeit für die Bearbeitung einer Ladung Wäsche)
- → **Durchsatz**: Für k Ladungen wird die Zeit $40 + 20k$ benötigt. In diesem Beispiel ergibt sich $4/(40 + 20 \cdot 4) = 1/30$ **Ladungen pro Minute**

Notizen

EIN BEISPIEL: WASCHEN MIT PIPELINING (II)

Werden sehr **viele Ladungen Wäsche gewaschen**, so ergibt sich sogar

$$\lim_{k \to \infty} \frac{k}{40 + 20 \cdot k} = \frac{1}{20}$$

der dreifache Durchsatz.

Die dargestellte Situation ist stark idealisiert, denn:

- → Die drei Teilaufgaben „waschen“, „bügeln“ und „trocknen“ haben den **identischen Zeitbedarf**.
- → Der **zusätzliche Zeitbedarf** (z.B. Ablage in einem Zwischenspeicher) für die **Überlappung** der Aufgaben wurde **vernachlässigt**.

Notizen

PIPELINING MIT SCHALTKREISEN

Der gleiche Ansatz kann auf Schaltkreise angewendet werden[3].

Ziel: Teile den kombinatorischen Schaltkreis in möglichst **identisch lang** arbeitende Teile auf (**Stages**).

Seien T_1, T_2, T_3 und T_4 die Delays der Stages 1-4, dann ist der Delay $T_{\max}$ des gepipelineten Schaltkreises:

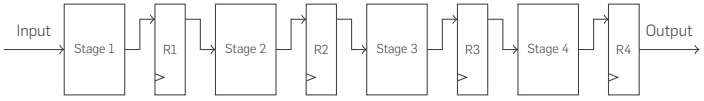
$$T_{\max} = \max\{T_1, T_2, T_3, T_4\}.$$

[3] Das Prinzip des Pipelinings in CPUs wird in D. Patterson und J. Hennessy, Computer Organization and Design, Morgan Kaufmann, 2012 in den Kapiteln 4.5 und 4.6 beschrieben.

Notizen

PIPELINING MIT SCHALTKREISEN (II)

Zur Regulierung des Signalflusses werden die (synchronen) Register R_1, R_2 und R_3 eingeführt. Register R_4 dient als **Ausgabebuffer**:



Sei T_{cq} die **Verzögerung eines Registers** R_i mit der das Clocksignal am Q-Ausgang ankommt, dann beträgt die minimale Periodendauer T_c

$$T_c = T_{\max} + T_{\text{set}} + T_{\text{cq}}$$

T_c gibt dabei die Dauer eines **Teilschritts** an.

Notizen

PIPELINING MIT SCHALTKREISEN (III)

Der ursprüngliche kombinatorische Schaltkreis braucht für die **gesamte** Aufgabe (Delay):

$$T_{\text{comb}} = T_1 + T_2 + T_3 + T_4$$

Für die Version mit Pipeline ergibt sich der schlechtere Delay

$$T_{\text{pipe}} = 4T_c = 4T_{\text{max}} + 4(T_{\text{set}} + T_{\text{cq}})$$

Hier zeigt sich, dass die **einzelnen Stages** möglichst den **gleichen Delay** haben sollten. Dies muss man evtl. durch mehrere Implementierungsversuche „ausprobieren“ und dann die Stages anpassen (Retiming).

Notizen

PIPELINING MIT SCHALTKREISEN (IV)

Der **Durchsatz des kombinatorischen Schaltkreises** beträgt

$$\frac{1}{T_{\text{comb}}}$$

Um k Tasks zu bearbeiten, benötigt die Variante mit Pipeline

$$3T_c + kT_c$$

Zeit und liefert den Durchsatz (für große k)

$$\frac{k}{3T_c + kT_c} \approx \frac{1}{T_c}$$

Unter den Annahmen, dass $T_{\text{set}} + T_{\text{cq}}$ vernachlässigbar **klein** und $T_{\text{max}} = T_{\text{comb}}/4$ (perfekt balancierte Stages) gilt:

$$T_{\text{pipe}} = 4T_c \approx 4T_{\text{max}} = T_{\text{comb}}$$

Notizen

PIPELINING MIT SCHALTKREISEN (V)

Damit ergibt sich

$$\frac{1}{T_c} \approx \frac{1}{T_{\max}} = \frac{4}{T_{\text{comb}}},$$

d.h. der **vierfache Durchsatz**. Diese Betrachtung kann auch auf eine Pipeline mit n **Stufen übertragen** werden.

Achtung: Verwendet man sehr viele Pipelinestufen, so wird T_c klein, aber $T_{\text{set}} + T_{\text{cq}}$ ist nicht mehr vernachlässigbar!

Für den **effektiven Einsatz** von Pipilining sollte ein Schaltkreis

- → kontinuierlich grosse Mengen an Daten verarbeiten müssen,
- → den Durchsatz als wichtigstes Designkriterium haben,
- → in möglichst gleiche Stages aufteilbar sein und
- → der Delay einer Stage ist groß gegenüber $T_{\text{set}} + T_{\text{cq}}$.

Notizen

Notizen