Simon Lohmann Dietmar Tutsch

Bergische Universität Wuppertal
Fakultät für Elektrotechnik, Informationstechnik und Medientechnik
Lehrstuhl für Automatisierungstechnik / Informatik

Hard Real-Time Memory-Management in a Single Clock Cycle (on FPGAs)

Speicherverwaltung in harter Echtzeit in einem Takt (auf FPGAs)

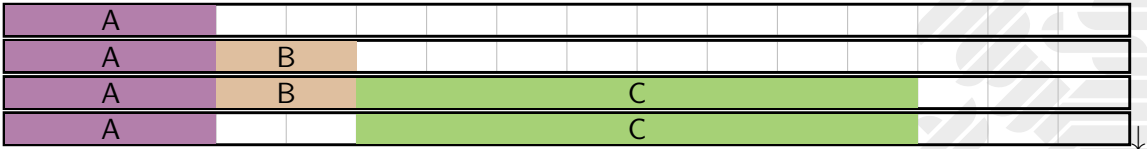
Echtzeit 2020
20. November

Übersicht

Hintergrund
- Echtzeitdatenbanksystem
- Dynamische Speicherverwaltung

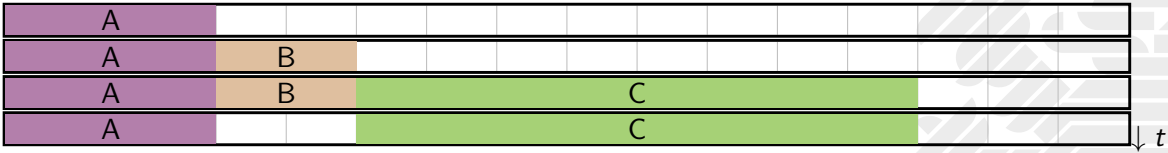
Zielplattform
- FPGA (Programmierbare Logik)
- *Aber auch anwendbar auf CPU*

Dynamische Speicherverwaltung:
„Zufälliges“ Reservieren und Freigeben



Dynamische Speicherverwaltung:

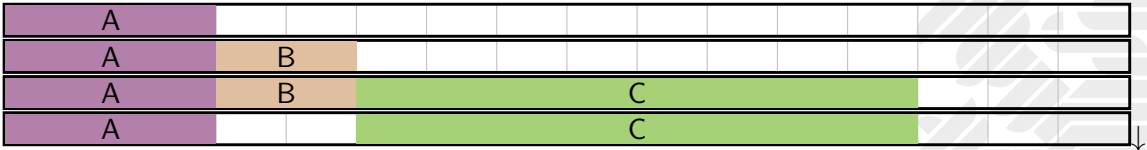
„Zufälliges“ Reservieren und Freigeben



Fragmentierung

Dynamische Speicherverwaltung:

„Zufälliges“ Reservieren und Freigeben



Fragmentierung

Defragmentierung

Dauer verteilungsabhängig

- Unvorhersehbar ⇒ keine Echtzeit

Statisch allokieren

- Speicherbereiche stehen zu Programmstart fest

\+ Sicher

\+ Einfach

– Zeitliche Dynamik nicht umsetzbar

Statisch allokieren

- Speicherbereiche stehen zu Programmstart fest

\+ Sicher

\+ Einfach

– Zeitliche Dynamik nicht umsetzbar

Memory Pool

- Allokationen gleicher Größe
- Größen-Dynamik fehlt

⇒ Kombination mehrerer Memory Pools

- Blockgrößen z.B. 1x, 2x, 4x, 8x,...

\+ Größendynamik (eingeschränkt)

- Aufteilen/Rekombinieren von Blöcken

– ⇒ Fragmentierung

Statisch allokieren

- Speicherbereiche stehen zu Programmstart fest

\+ Sicher

\+ Einfach

– Zeitliche Dynamik nicht umsetzbar

Memory Pool

- Allokationen gleicher Größe
- Größen-Dynamik fehlt

⇒ Kombination mehrerer Memory Pools

- Blockgrößen z.B. 1x, 2x, 4x, 8x,...

\+ Größendynamik (eingeschränkt)

- Aufteilen/Rekombinieren von Blöcken

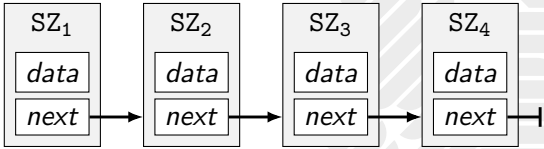
– ⇒ Fragmentierung

Allokation aufteilen in ...

- Reservierung (optional)
- Abholung einzelner Zellen

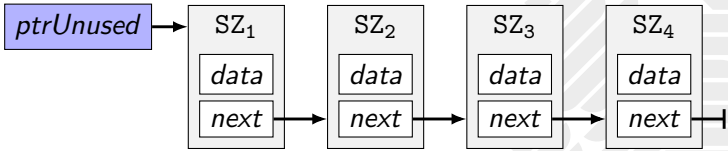
Speicherzellen

- *data*: Nutzdaten/„Payload“
  - Rohdaten
  - Metadaten, ...
- *next*: Pointer
  - Verwaltung der Speicherzellen



Speicherzellen

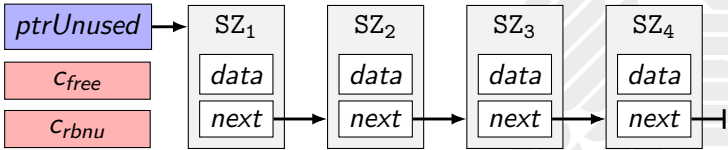
- *data*: Nutzdaten/„Payload“
  - Rohdaten
  - Metadaten, ...
- *next*: Pointer
  - Verwaltung der Speicherzellen



Pointer		Bedeutung
ptrUnused	„list of unused cells“	Liste der nicht verwendeten Speicherzellen Listenlänge $\#_{unused} = c_{free} + c_{rbnu}$

Speicherzellen

- *data*: Nutzdaten/„Payload“
  - Rohdaten
  - Metadaten, ...
- *next*: Pointer
  - Verwaltung der Speicherzellen



Pointer		Bedeutung
ptrUnused	„list of unused cells“	Liste der nicht verwendeten Speicherzellen Listenlänge $\#_{unused} = c_{free} + c_{rbnu}$

Zähler		Bedeutung
c_{free}	„free“	# freie Speicherzellen
c_{rbnu}	„reserved but not used“	# bereits reservierte, aber (noch) nicht verwendete Speicherzellen

Architektur – Operationen

- *PickUpFreeCell*
- *ReturnFreeCellByAddress*

} Memory Pool

- *RequestReservation*(n)
- *PickUpReservedCell*
- *ReturnReservationByAmount*(n)

} Separate Reservierung

- *ReturnFreeCellRingByAddress*

} Zurückgeben beliebiger Mengen

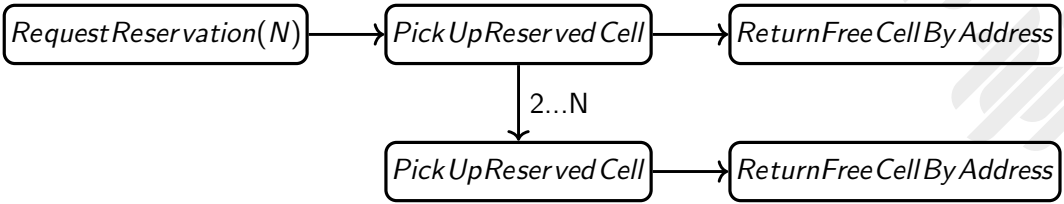
- *Idle*

} Vollständigkeit der Architektur

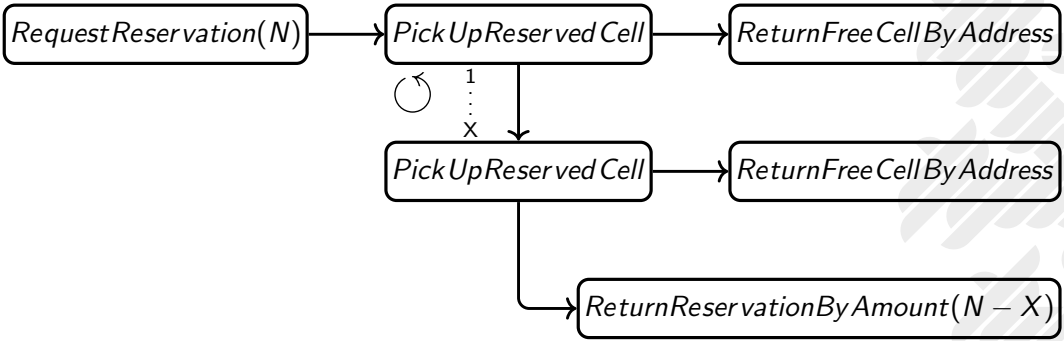
Zellen einzeln abholen & zurückgeben *(vgl. Blöcke im Memory Pool)*

Zellen einzeln abholen & zurückgeben *(vgl. Blöcke im Memory Pool)*

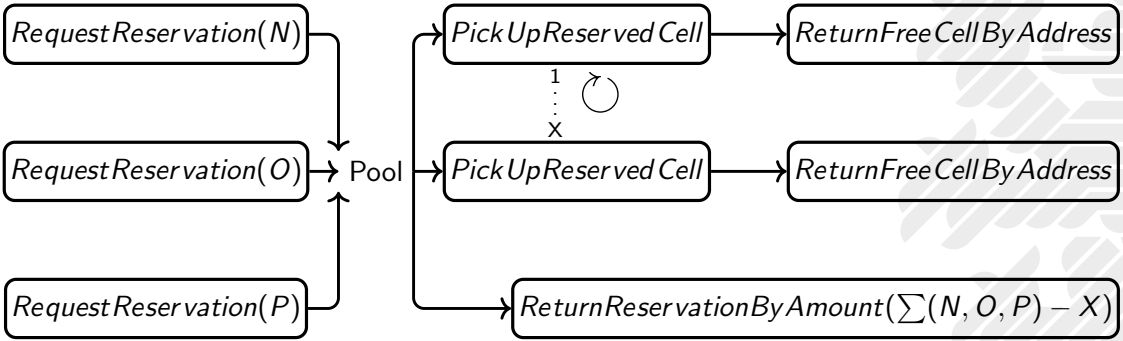
Reservieren, dann Zelle für Zelle abholen



Reservierung teilweise nutzen



Reservierungen zusammenführen



Zähler

Multiplexer + Addierer

$$c^{+}_{free} = c_{free} + f$$
$$c^{+}_{rbnu} = c_{rbnu} + r$$

Operation	f	r
Idle	0	0
PickUpFreeCell	-1	0
PickUpReservedCell	0	-1
RequestReservation	$-n$	n
ReturnFreeCellByAddress	1	0
ReturnFreeCellRingByAddress	$memory(addressIn).data$	0
ReturnReservationByAmount	n	$-n$

Zähler

Multiplexer + Addierer

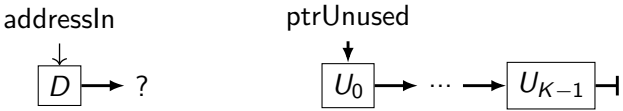
$$c^+_{free} = c_{free} + f$$
$$c^+_{rbnu} = c_{rbnu} + r$$

Multi-Input-Addierer, Koeffizienten ± 1

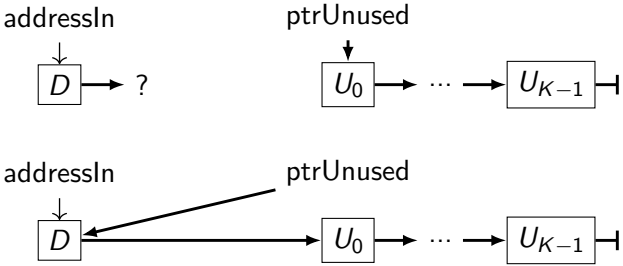
$$c^+_{free} = c_{free} + a_0 \cdot 1 + a_1 \cdot n + a_2 \cdot memory(addressIn).data$$
$$c^+_{rbnu} = c_{rbnu} + b_0 \cdot 1 + b_1 \cdot n$$

Operation	f	r	a_0	a_1	a_2	b_0	b_1
Idle	0	0	0	0	0	0	0
PickUpFreeCell	-1	0	-1	0	0	0	0
PickUpReservedCell	0	-1	0	0	0	-1	0
RequestReservation	$-n$	n	0	-1	0	0	$+1$
ReturnFreeCellByAddress	1	0	$+1$	0	0	0	0
ReturnFreeCellRingByAddress	$memory(addressIn).data$	0	0	0	$+1$	0	0
ReturnReservationByAmount	n	$-n$	0	$+1$	0	0	-1

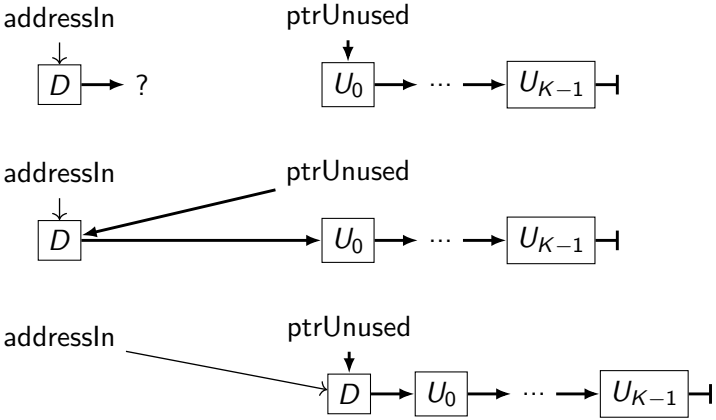
ReturnFreeCellByAddress



ReturnFreeCellByAddress



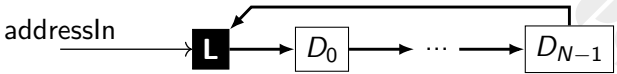
ReturnFreeCellByAddress



ReturnFreeCellRingByAddress

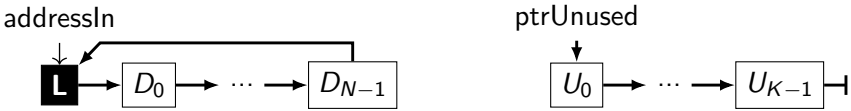
Voraussetzung

- Ringstruktur
- Länge bekannt

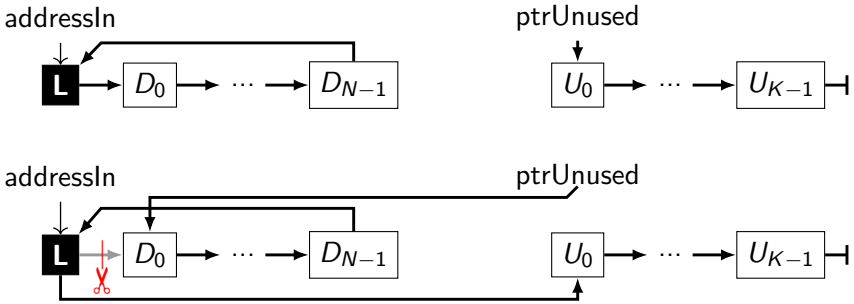


Knoten	Bedeutung	Inhalt
L	Länge	$.data = \#Zellen$ $.next = \&D_0$
D_i	Daten	$.data =$ Datenstück i $.next = \begin{cases} \&D_{i+1} & \text{für} \quad i < N-1 \\ \&L & \text{für} \quad i = N-1 \end{cases}$

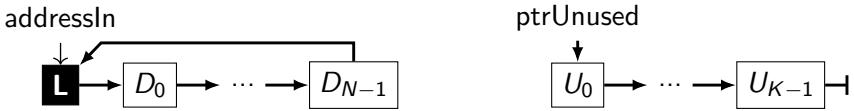
Ablauf

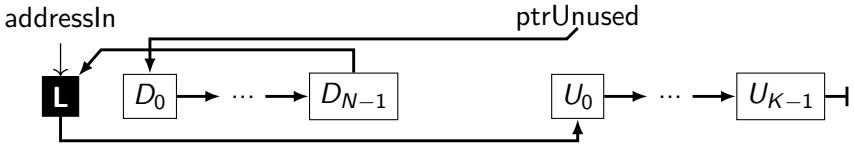


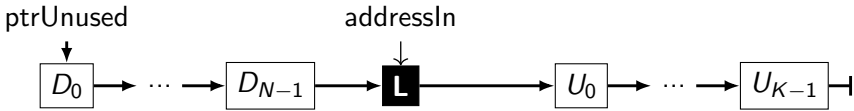
Ablauf


addressIn
ptrUnused
L
D_0
D_{N-1}
U_0
U_{K-1}
addressIn
ptrUnused
L
D_0
D_{N-1}
U_0
U_{K-1}

Ablauf


addressIn
ptrUnused
L
D_0
D_{N-1}
U_0
U_{K-1}


addressIn
ptrUnused
L
D_0
D_{N-1}
U_0
U_{K-1}


ptrUnused
addressIn
D_0
D_{N-1}
L
U_0
U_{K-1}

Komplexität

Element	#	Berechnungstyp	Komplexität
Zähler	2	Summe über endliche Menge	$\mathcal{O}(1)$
Zeiger	2	Auswahl eins aus zwei	$\mathcal{O}(1)$
Zellenzugriffe	2	Lesen & Schreiben	$\mathcal{O}(1)$

Komplexität

Element	#	Berechnungstyp	Komplexität
Zähler	2	Summe über endliche Menge	$\mathcal{O}(1)$
Zeiger	2	Auswahl eins aus zwei	$\mathcal{O}(1)$
Zellenzugriffe	2	Lesen & Schreiben	$\mathcal{O}(1)$

Parallelisierbarkeit

- Rechnungen $\rightarrow$ unabhängig
- Pointerzuweisungen $\rightarrow$ unabhängig
- Zellenzugriffe $\rightarrow$ unabhängig

$\Rightarrow$ Vollständig parallelisierbar
$\Rightarrow$ FPGA: Ausführung in einem Takt

Funktionalität

+ Reservieren/Zurückgeben beliebig großer Speichermengen in einem Schritt
+ Beliebige Aufteilung & Rekombination von Reservierungen

Funktionalität

+ Reservieren/Zurückgeben beliebig großer Speichermengen in einem Schritt
+ Beliebige Aufteilung & Rekombination von Reservierungen

– Kein zusammenhängender Speicher
  ⇒ Virtuelle Speicherverwaltung
  ⇒ Speicherzellen direkt nutzen
– Kein Wissen über aktive Reservierungen oder abgeholte Speicherzellen
  ⇒ Erfordert kooperative Nutzer

Funktionalität

\+ Reservieren/Zurückgeben beliebig großer Speichermengen in einem Schritt

\+ Beliebige Aufteilung & Rekombination von Reservierungen

– Kein zusammenhängender Speicher
⇒ Virtuelle Speicherverwaltung
⇒ Speicherzellen direkt nutzen

– Kein Wissen über aktive Reservierungen oder abgeholte Speicherzellen
⇒ Erfordert kooperative Nutzer

Performanz

– Großer Speicheraufwand für Pointer
⇒ Nachteil entfällt wenn *next*-Pointer vom Nutzer verwendet wird

Funktionalität

+ Reservieren/Zurückgeben beliebig großer Speichermengen in einem Schritt
+ Beliebige Aufteilung & Rekombination von Reservierungen

– Kein zusammenhängender Speicher
  ⇒ Virtuelle Speicherverwaltung
  ⇒ Speicherzellen direkt nutzen
– Kein Wissen über aktive Reservierungen oder abgeholte Speicherzellen
  ⇒ Erfordert kooperative Nutzer

Performanz

+ Hart echtzeitfähig
+ Geringer Rechenaufwand
  ⇒ kleiner FPGA Footprint
+ Teilschritte unabhängig
  ⇒ Perfekt parallelisierbar
    + FPGA: Ein Takt pro Befehl

– Großer Speicheraufwand für Pointer
  ⇒ Nachteil entfällt wenn *next*-Pointer vom Nutzer verwendet wird

Literatur

- Borg, A., Wellings, A., Gill, C. and Cytron, R.K.: Real-Time Memory Management: Life and Times. In Proceedings of the 18th Euromicro Conference on Real-Time Systems, ECRTS'06, pp. 237–247. Dresden (2006), https://doi.org/10.1109/ECRTS.2006.21
- AUTOSAR: Guidelines for the use of the C++14 language in critical and safety-related systems, AUTOSAR AP, Release 19-03. (2019)
- Kenwright, B.: Fast Efficient Fixed-Size Memory Pool – No Loops and No Overhead In: Ullrich, T., Lorenz, P. (eds.) The Third International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking, COMPUTATION TOOLS 2012, pp. 1–6. Nice, France (2012).
- Lindblad, J.: Handling memory fragmentation. In EDN EUROPE (2004)
- Puaut, I.: Real-Time Performance of Dynamic Memory Allocation Algorithms. In Proceedings of the 14th Euromicro Conference on Real-Time Systems, ECRTS'02 (2002), https://doi.org/10.1109/EMRTS.2002.1019184
- Hopcroft, J.E., Paul,W.J. and Valiant, L.G.: On time versus space and related problems. In Proceedings of the 16th Annual Symposium on Foundations of Computer Science, Berkeley, California, USA, pp. 57–64. IEEE Computer Society (1975), https://doi.org/10.1109/SFCS.1975.23

Vielen Dank für Ihre Aufmerksamkeit...

Fragen?