# Hochleistungsrechnen auf dem PC

Steffen Börm

Christian-Albrechts-Universität zu Kiel

Ringvorlesung Informatik, 26. Juni 2014

# Übersicht

# Übersicht

# Computergrafik: Rendering

Aufgabe: Möglichst realistische Darstellung eines dreidimensionalen Computermodells in Echtzeit.

Quelle: Wikipedia

Algorithmus: Gegenstände dargestellt durch viele tausend Dreiecke, die auf eine imaginäre Leinwand projiziert werden müssen.

# Computergrafik: Raytracing

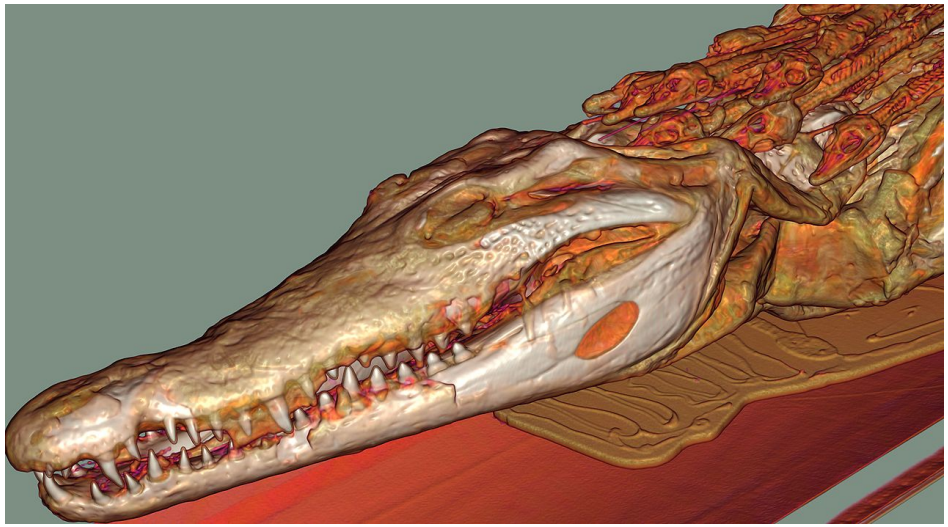
Aufgabe: Fotorealistische Darstellung eines Computermodells.

Quelle: Wikipedia

Algorithmus: Von einem imaginären Auge ausgehende *Sehstrahlen* werden in die Szene entsandt. Bei Kontakt mit Oberflächen entscheiden physikalische Modelle, was diese Strahlen „sehen“.

# Computergrafik: Medizinische Bildgebung

Aufgabe: Darstellung einer Tomographie.



Quelle: Wikipedia

Algorithmus: Tomographie führt zu *Voxel-Modell* eines Objekts. Zu dessen Darstellung werden die Beiträge einzelner Voxel entlang von Sehstrahlen gesammelt.

# Computersimulation: Wellenausbreitung

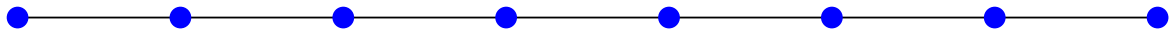
Aufgabe: Simulation der Ausbreitung einer Welle.

Algorithmus:

- Modell: Ersetze kontinuierliche Saite durch mit Federn gekoppelte Punktmassen.

# Computersimulation: Wellenausbreitung

Aufgabe: Simulation der Ausbreitung einer Welle.

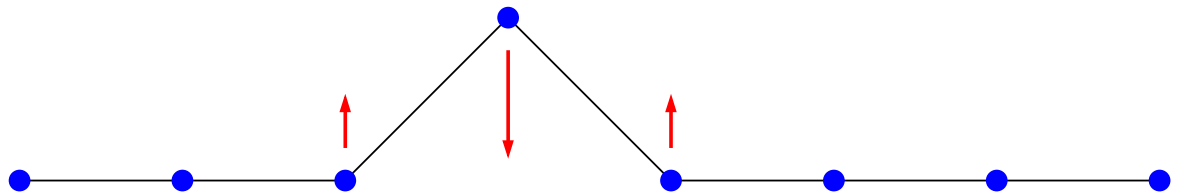


Algorithmus:

- Modell: Ersetze kontinuierliche Saite durch mit Federn gekoppelte Punktmassen.

# Computersimulation: Wellenausbreitung

Aufgabe: Simulation der Ausbreitung einer Welle.

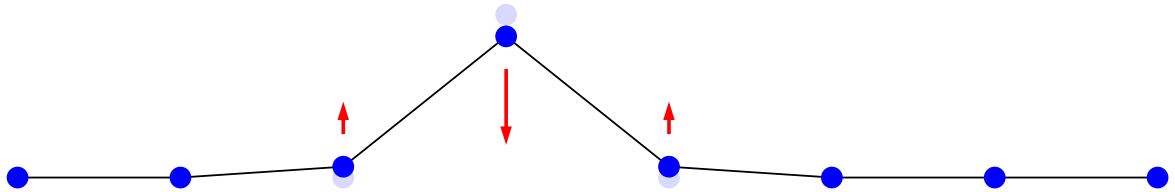


Algorithmus:

- Modell: Ersetze kontinuierliche Saite durch mit Federn gekoppelte Punktmassen.
- Physik: Bestimme Kräfte, Geschwindigkeiten und Positionen.

# Computersimulation: Wellenausbreitung

Aufgabe: Simulation der Ausbreitung einer Welle.



Algorithmus:

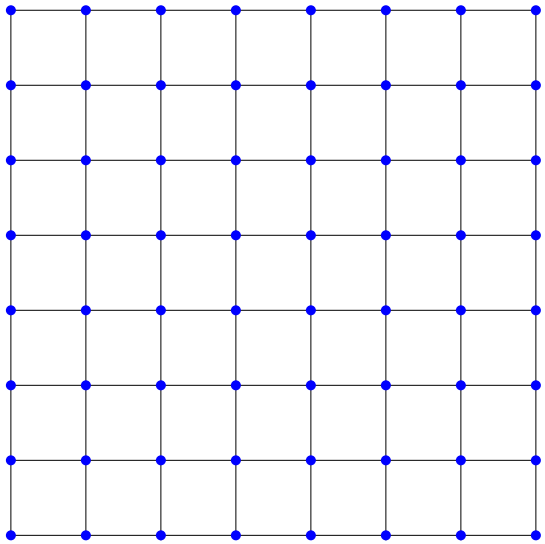
- Modell: Ersetze kontinuierliche Saite durch mit Federn gekoppelte Punktmassen.
- Physik: Bestimme Kräfte, Geschwindigkeiten und Positionen.
- Zeitschrittverfahren: Setze Kräfte und Geschwindigkeiten als für kurze Zeiten konstant.

# Computersimulation: Mehrdimensionale Wellen

Aufgabe: Simulation der Ausbreitung einer Welle

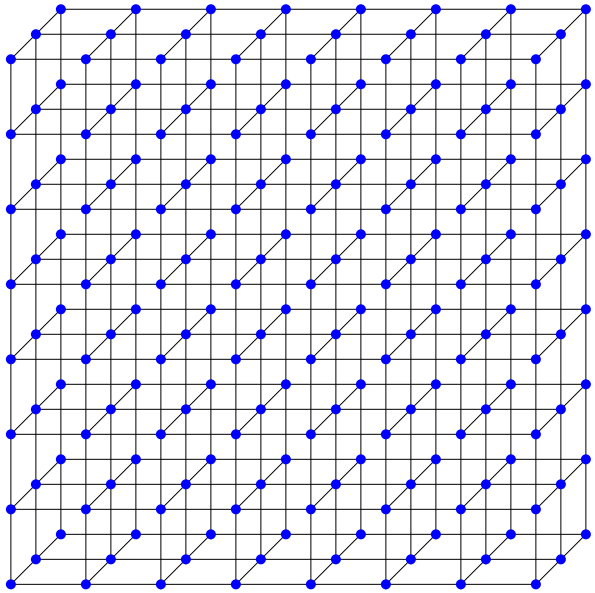
# Computersimulation: Mehrdimensionale Wellen

Aufgabe: Simulation der Ausbreitung einer Welle im zwei-



# Computersimulation: Mehrdimensionale Wellen

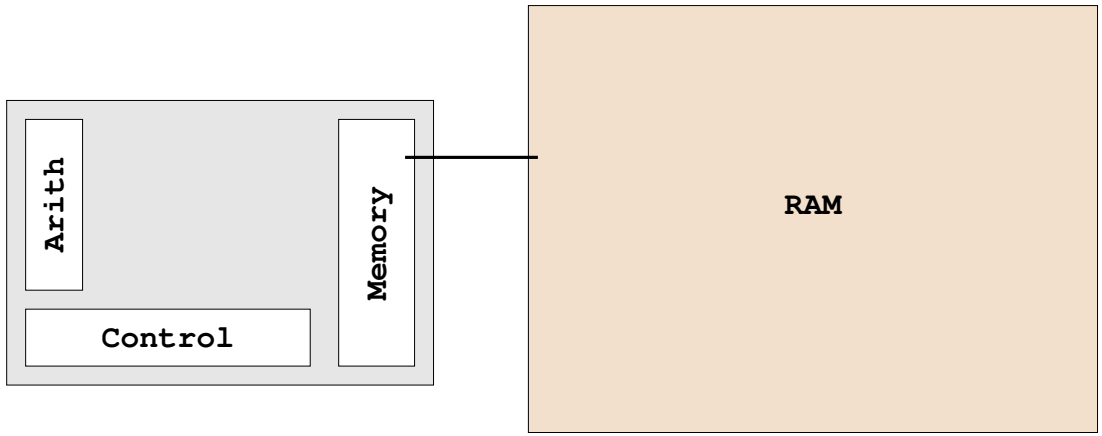
Aufgabe: Simulation der Ausbreitung einer Welle im zwei- oder dreidimensionalen Medium.



Schwierigkeit: Brauchbare Genauigkeit erfordert sehr viele Punkte.
$\rightarrow$ Sehr hoher Rechenaufwand.

# Übersicht

# Prozessor




Bestandteile:

- Steuerwerk regelt Ablauf eines Programms.
- Rechenwerk führt Berechnungen aus.
- Speichercontroller regelt Zugriff auf den Speicher.

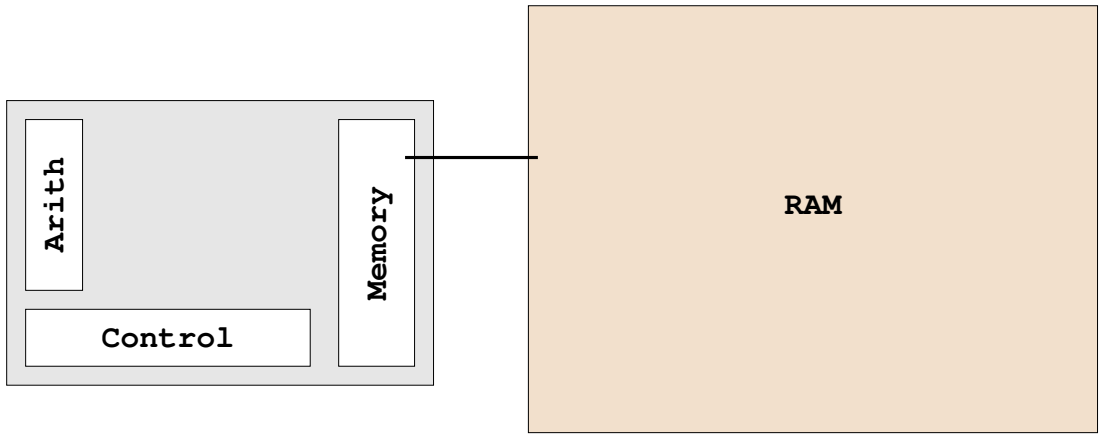
# Implementierung Wellengleichung

Beispiel: Leapfrog-Verfahren für die Simulation der Wellenausbreitung.

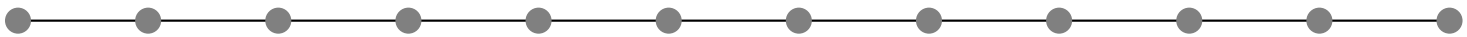
Daten: Auslenkungen in Array `x`, Geschwindigkeiten in Array `v`.

```
for(i=1; i<=N; i++)
  x[i] += xdelta * v[i];

for(i=1; i<=N; i++)
  v[i] += vdelta * (x[i+1] + x[i-1] - 2*x[i]);
```

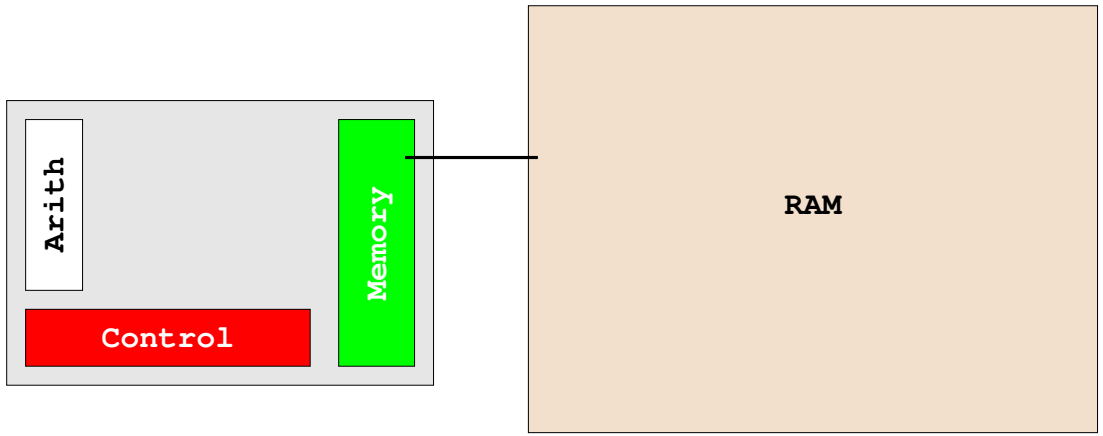
# Einfacher Prozessor im Beispiel



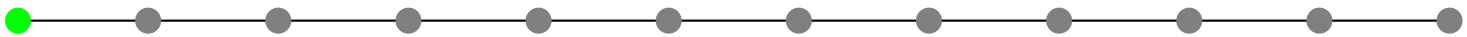

## Anwendung auf Wellengleichung:



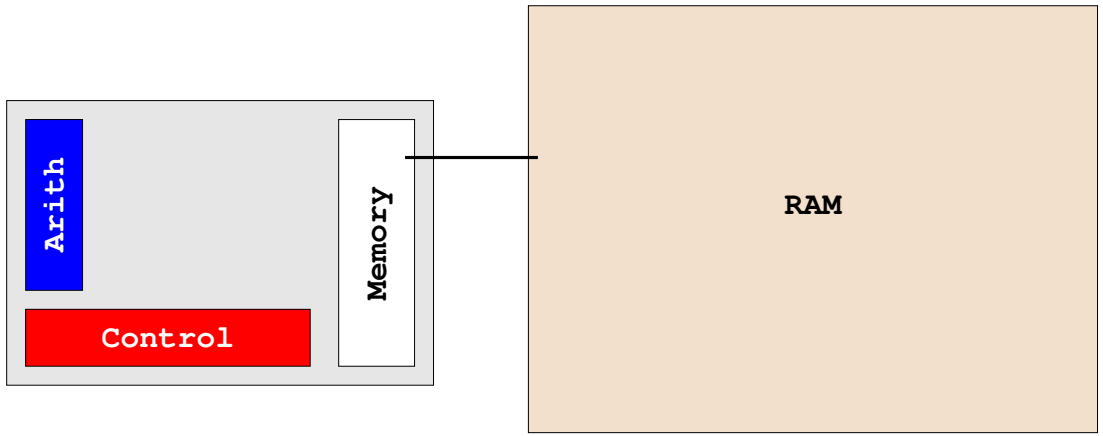
# Einfacher Prozessor im Beispiel



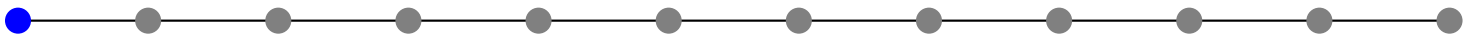

Anwendung auf Wellengleichung:



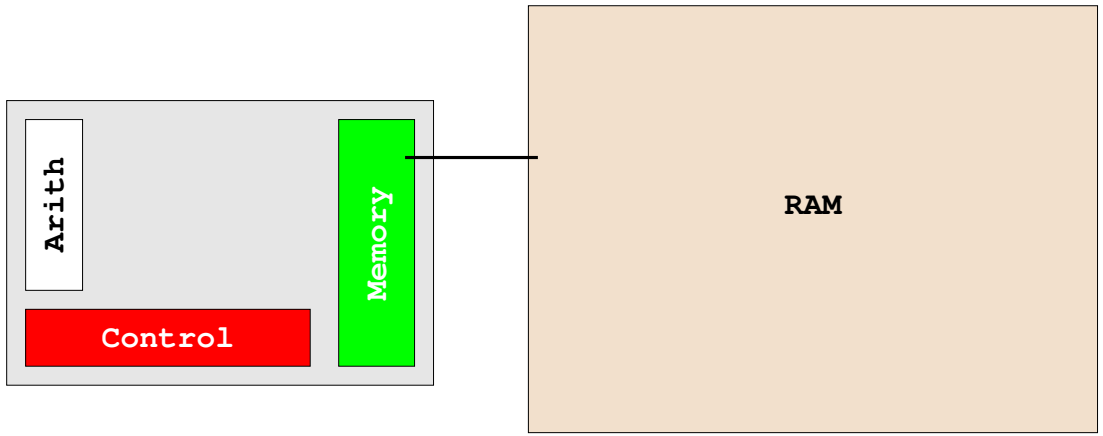
# Einfacher Prozessor im Beispiel



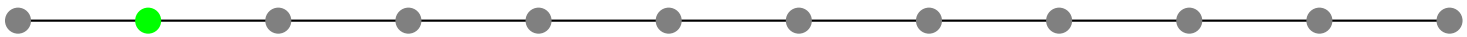

Anwendung auf Wellengleichung:



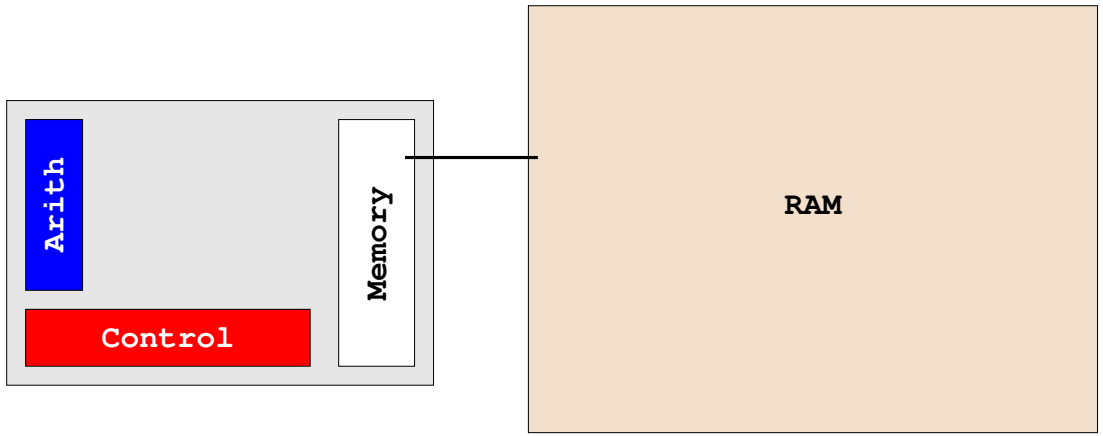
# Einfacher Prozessor im Beispiel



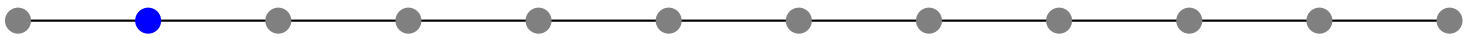

## Anwendung auf Wellengleichung:



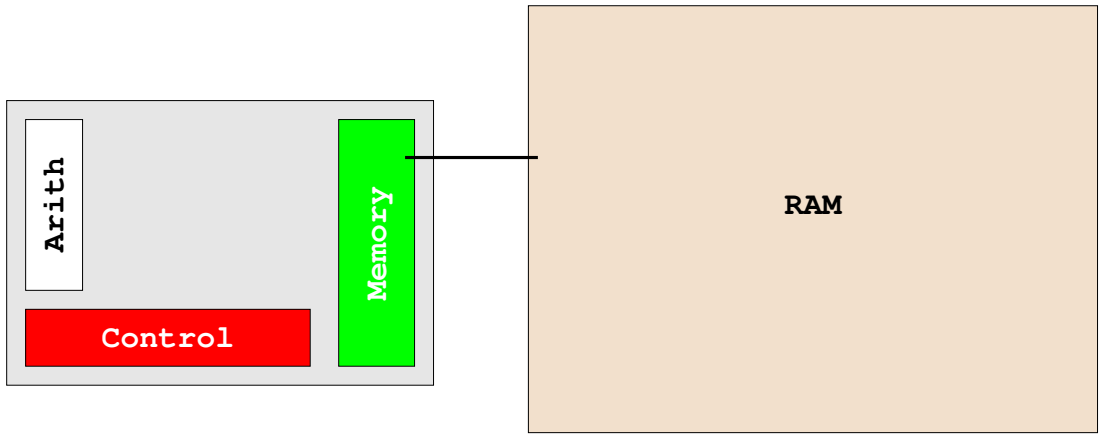
# Einfacher Prozessor im Beispiel



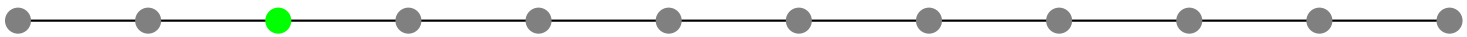

Anwendung auf Wellengleichung:



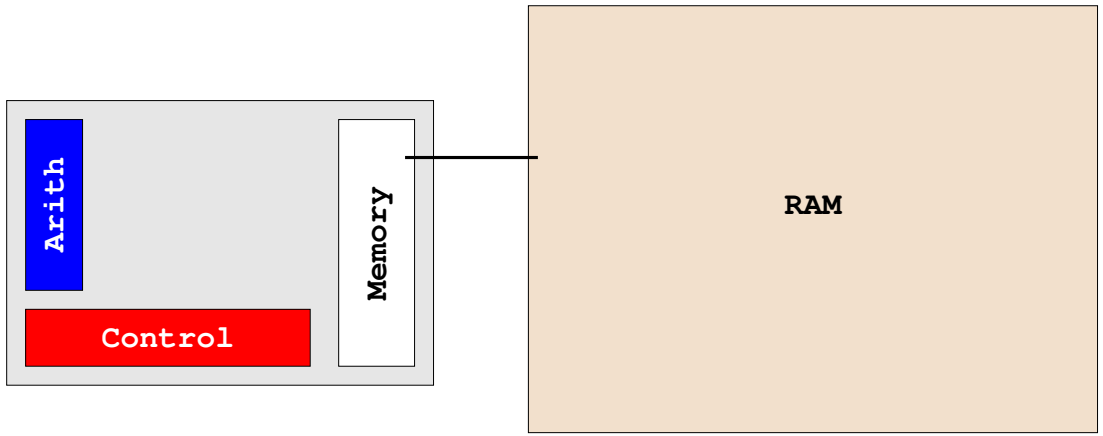
# Einfacher Prozessor im Beispiel



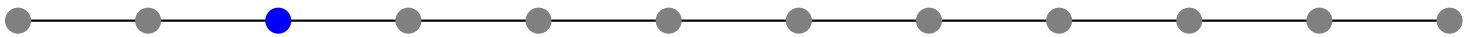

Anwendung auf Wellengleichung:



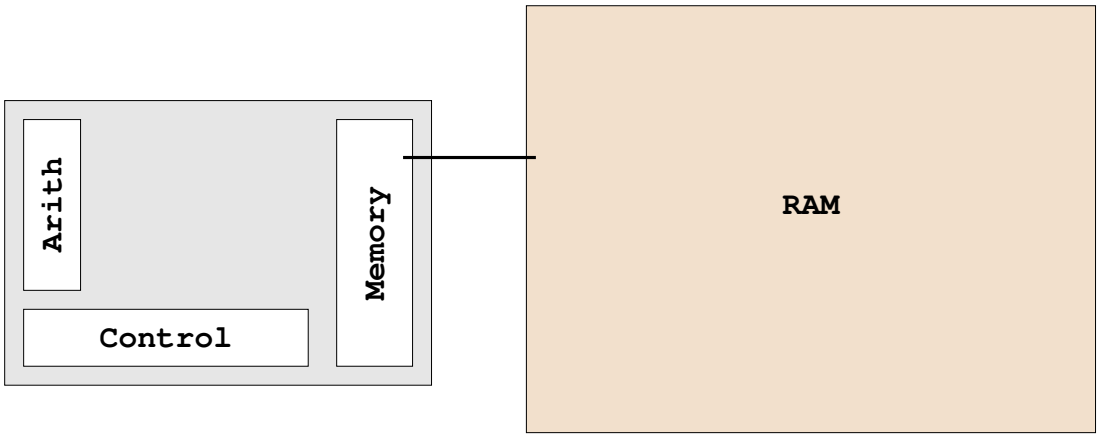
# Einfacher Prozessor im Beispiel




## Anwendung auf Wellengleichung:

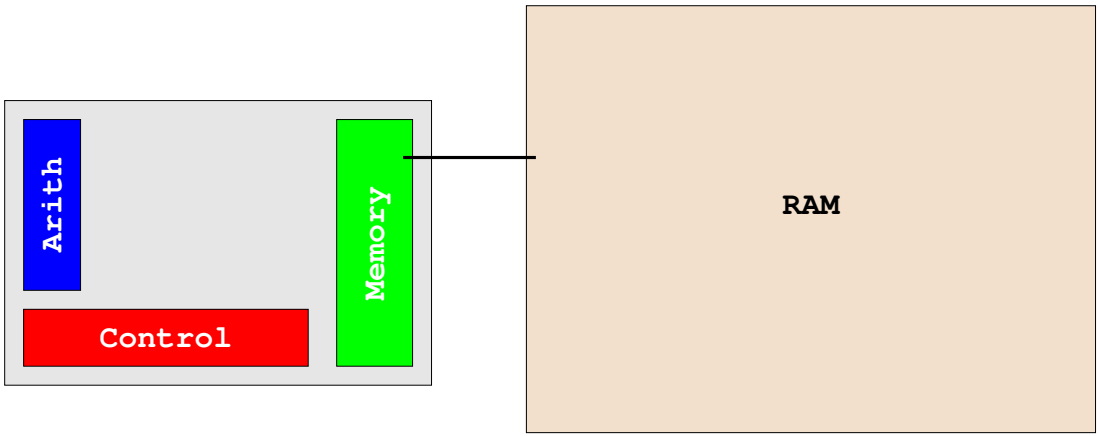


# Einfacher Prozessor im Beispiel




Nachteil: In jedem Takt ist neben dem Steuerwerk nur eine Funktionseinheit aktiv.

# Superskalarer Prozessor

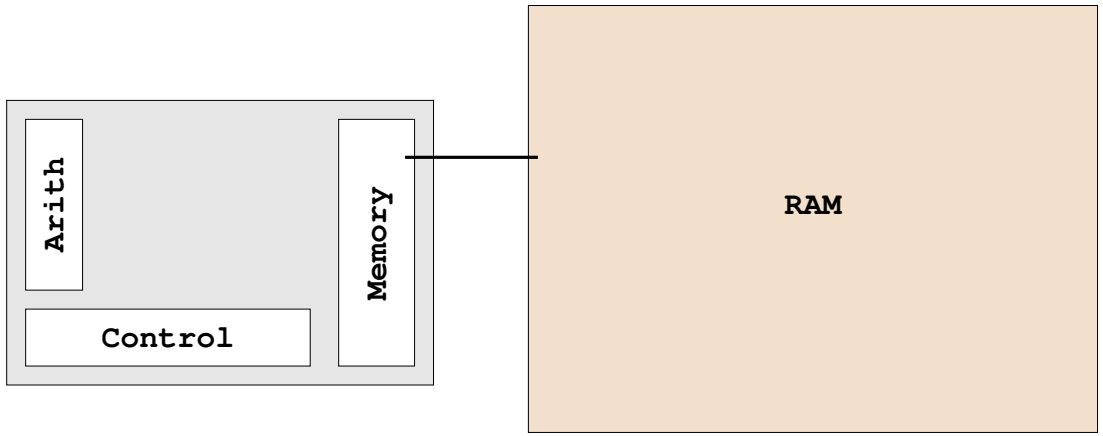



Idee:

- Führe mehrere Befehle gleichzeitig aus, sofern sie unabhängig voneinander sind.
- Sortiere eventuell Befehle um, um möglichst gute Auslastung aller Funktionseinheiten zu erreichen.

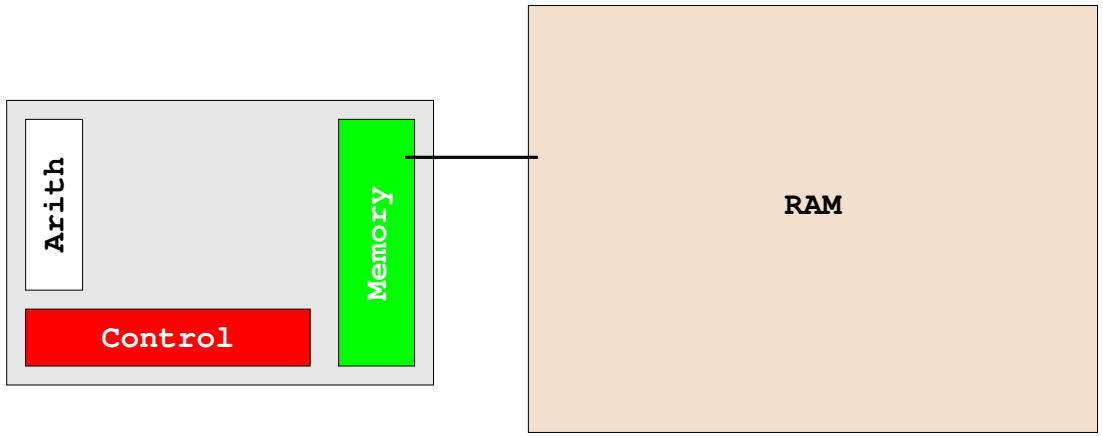
1965 CDC 6600 Supercomputer, 1993 Intel Pentium Prozessor.

# Superskalarer Prozessor im Beispiel



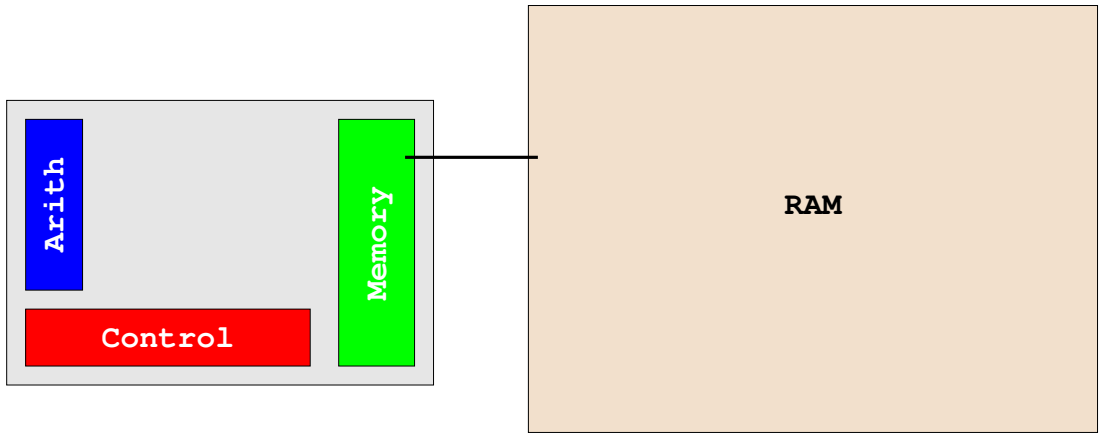

## Anwendung auf Wellengleichung:

# Superskalarer Prozessor im Beispiel



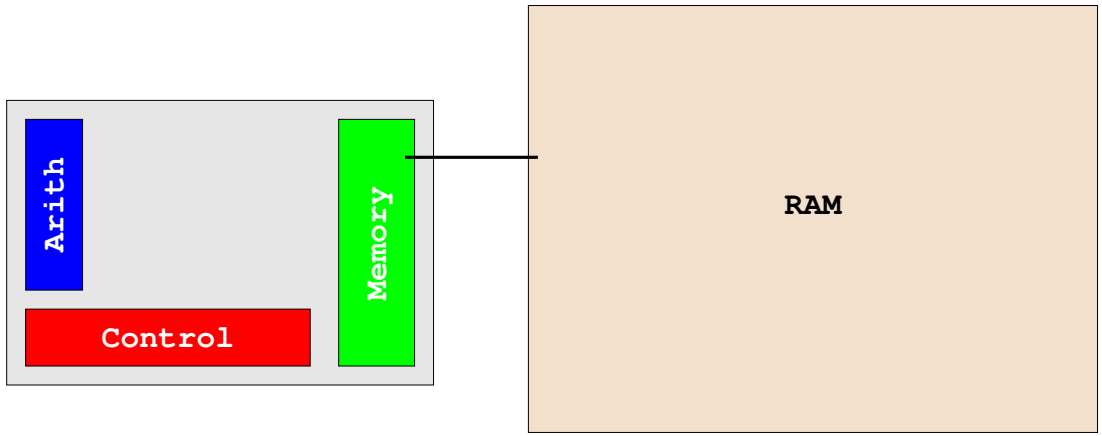

## Anwendung auf Wellengleichung:

# Superskalarer Prozessor im Beispiel



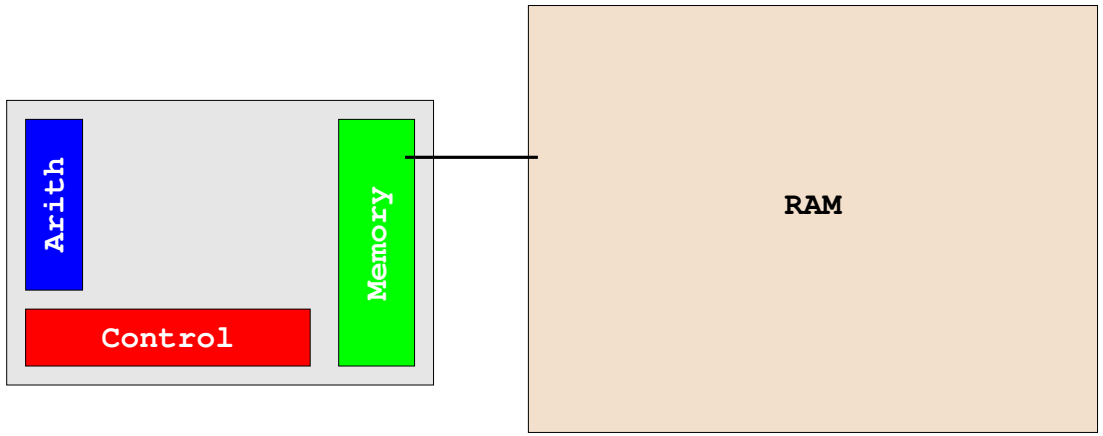

Anwendung auf Wellengleichung:

# Superskalarer Prozessor im Beispiel



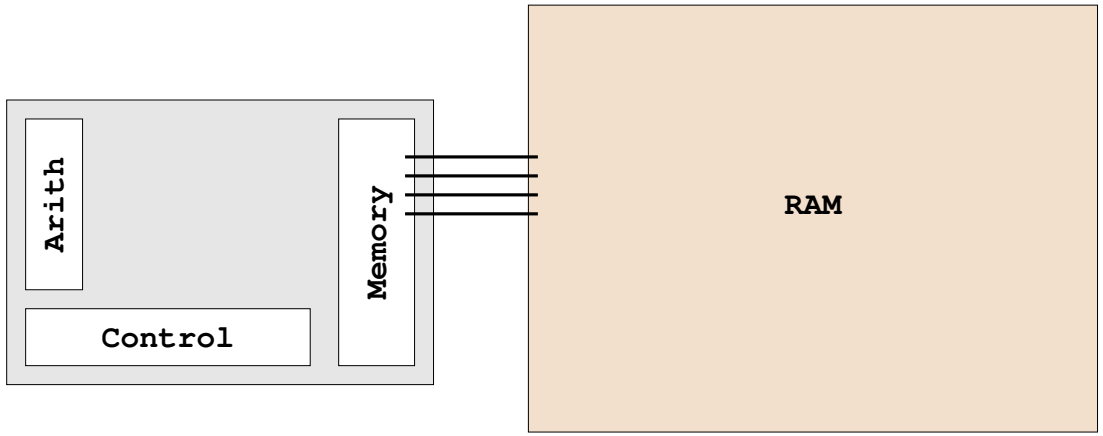

Anwendung auf Wellengleichung:

# Superskalarer Prozessor im Beispiel




Anwendung auf Wellengleichung:

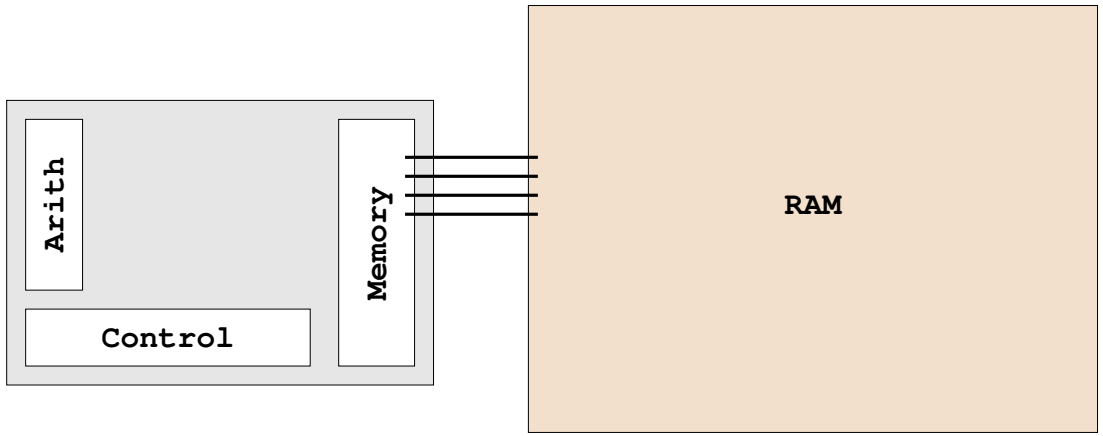
# Verbesserter Speicherzugriff




Ideen:

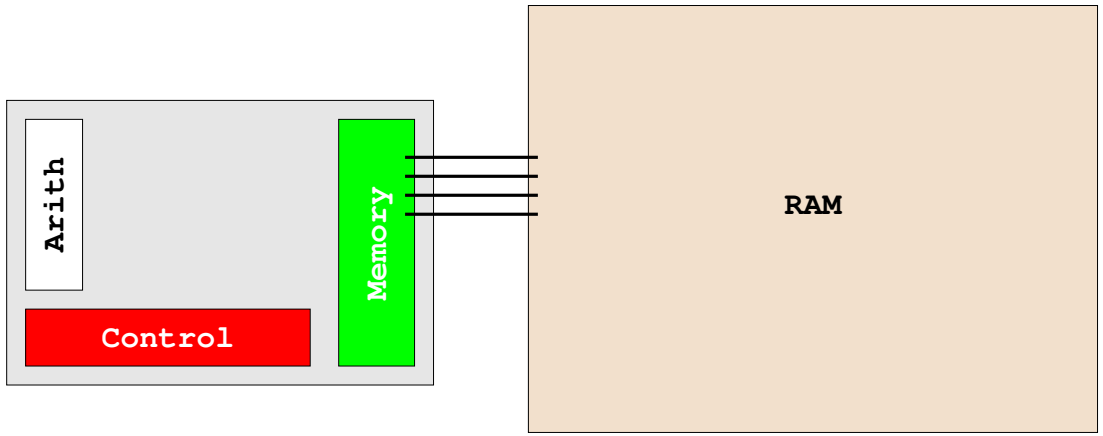
- Burst-Betrieb für aufeinander folgende Adressen.
- Caches
  (1968 IBM System/360 Model 85, 1989 Intel 80486).
- Mehrkanaliger Speicherzugriff
  (1968 CDC 6600, 2004 AMD Athlon 64).

# Burst-Zugriffe im Beispiel



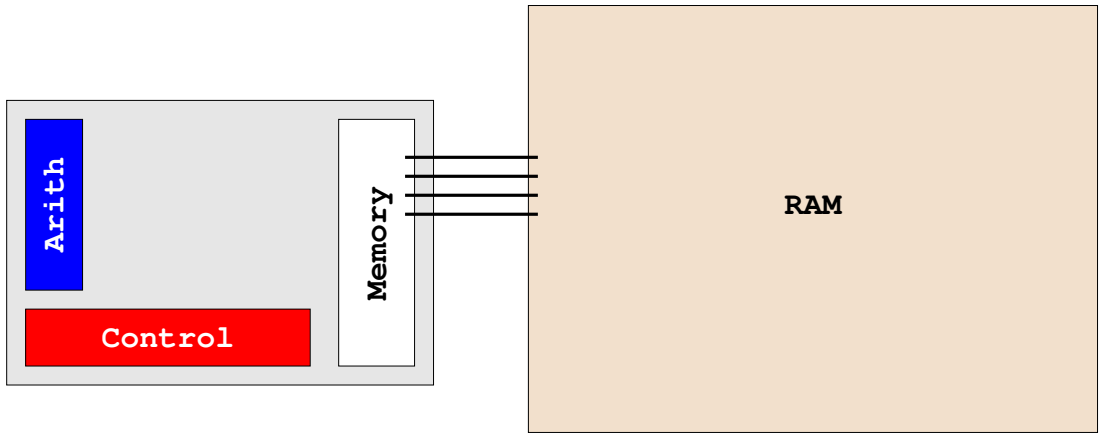

## Anwendung auf Wellengleichung:

# Burst-Zugriffe im Beispiel



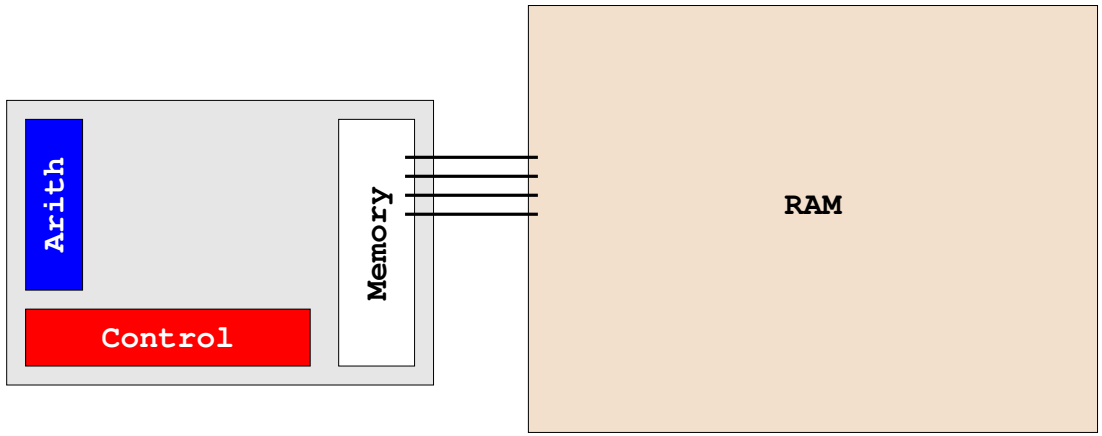

Anwendung auf Wellengleichung:

# Burst-Zugriffe im Beispiel



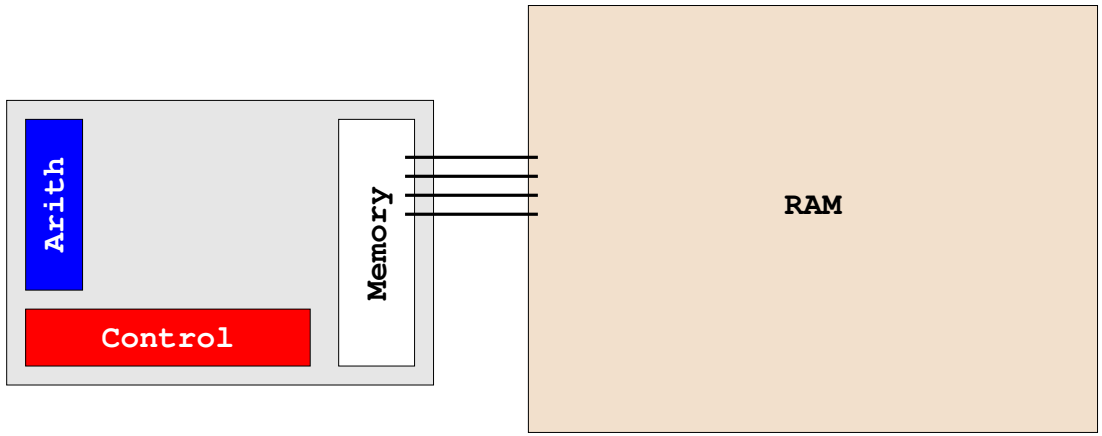

Anwendung auf Wellengleichung:

# Burst-Zugriffe im Beispiel



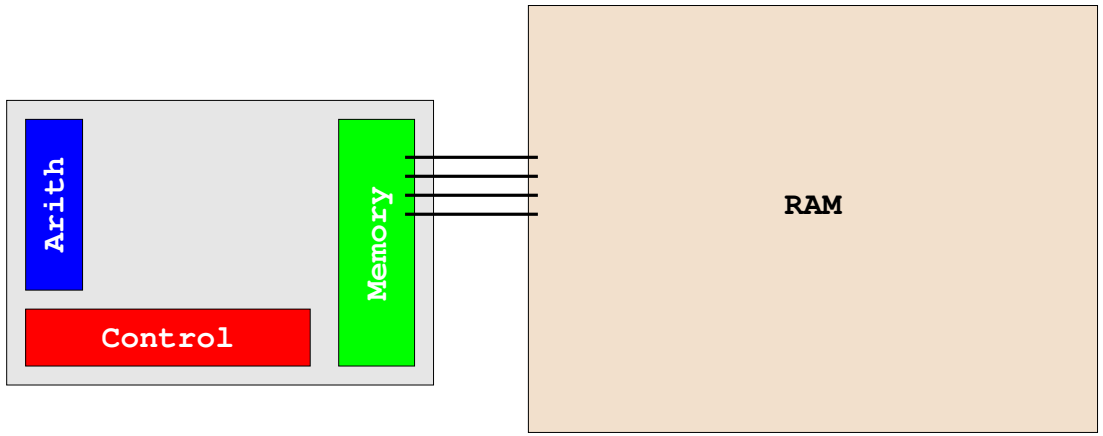

Anwendung auf Wellengleichung:

# Burst-Zugriffe im Beispiel



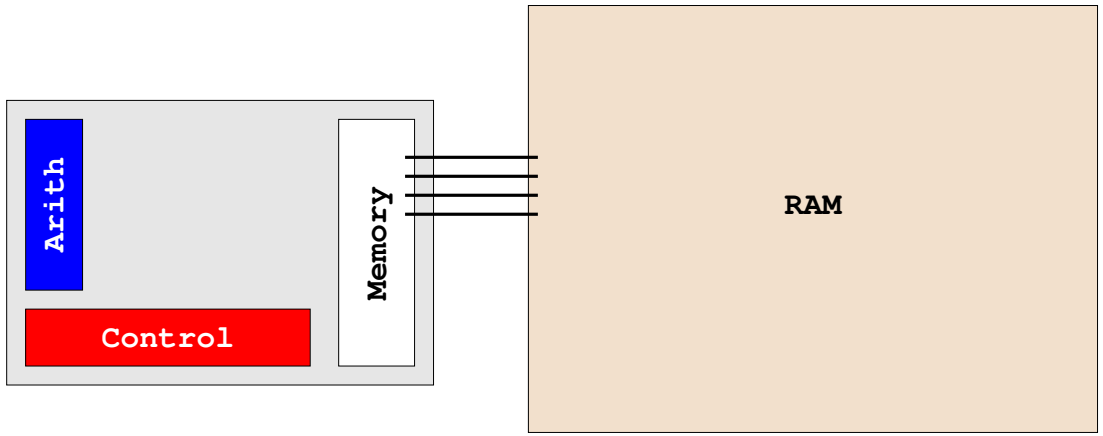

Anwendung auf Wellengleichung:

# Burst-Zugriffe im Beispiel




Anwendung auf Wellengleichung:

# Burst-Zugriffe im Beispiel




Anwendung auf Wellengleichung:

# Zweidimensionale Wellengleichung

## Erste Fassung:

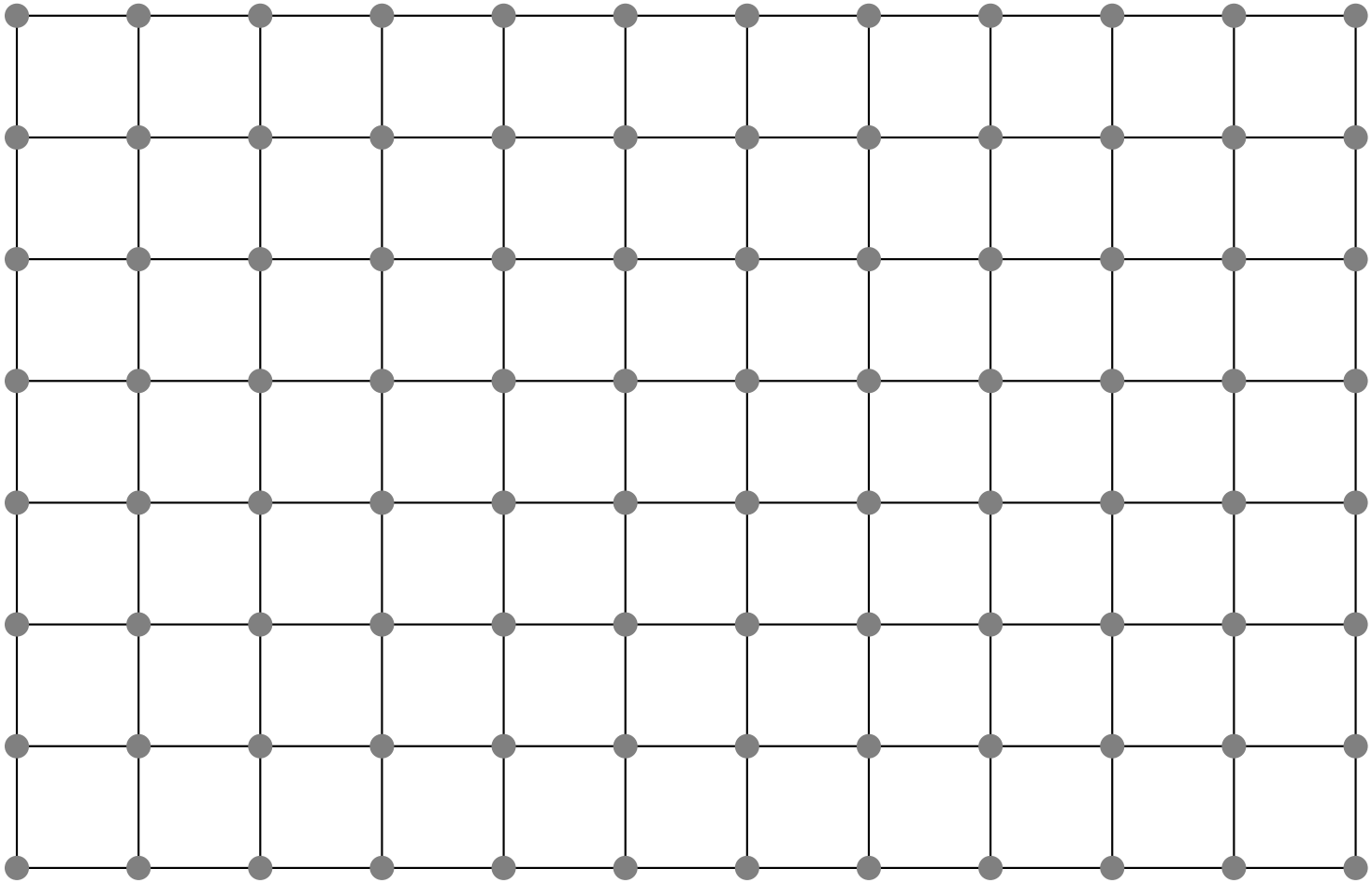
```
for(i=1; i<=N; i++)
  for(j=1; j<=N; j++)
    x[i][j] += xdelta * v[i][j];

for(i=1; i<=N; i++)
  for(j=1; j<=N; j++)
    v[i][j] += vdelta * (...);
```

## Zweite Fassung:

```
for(j=1; j<=N; j++)
  for(i=1; i<=N; i++)
    x[i][j] += xdelta * v[i][j];

for(j=1; j<=N; j++)
  for(i=1; i<=N; i++)
    v[i][j] += vdelta * (...);
```

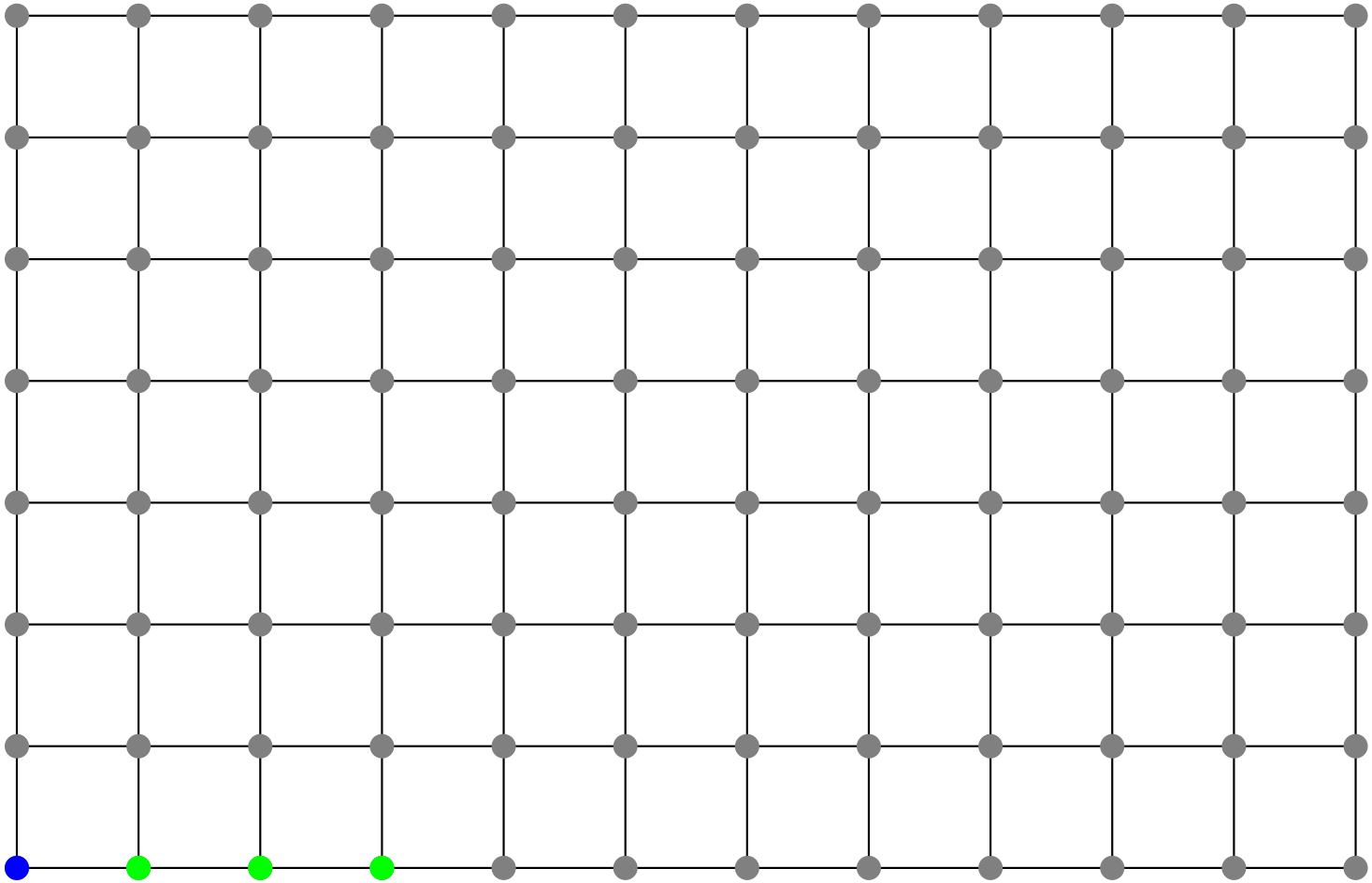
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



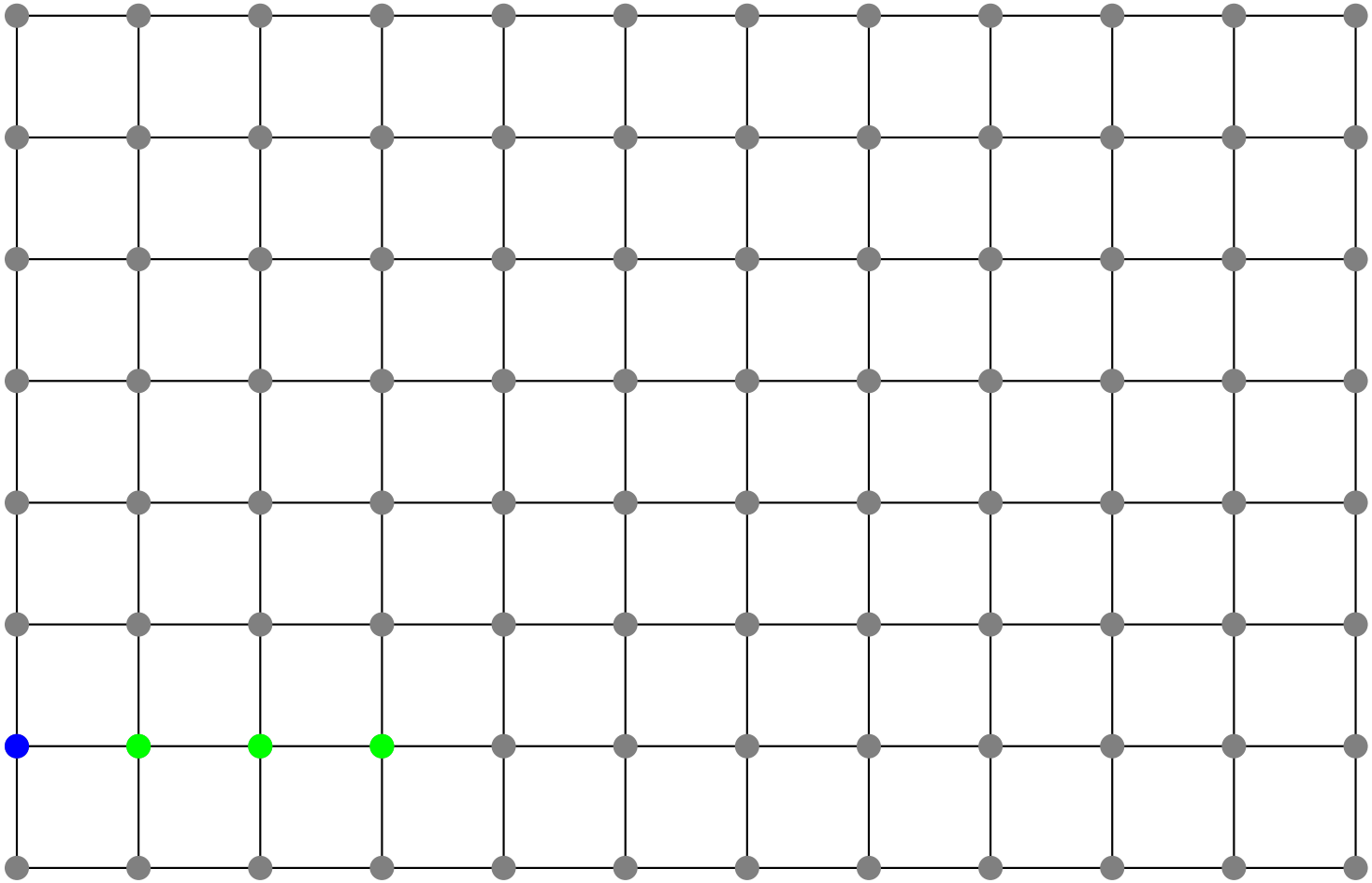
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



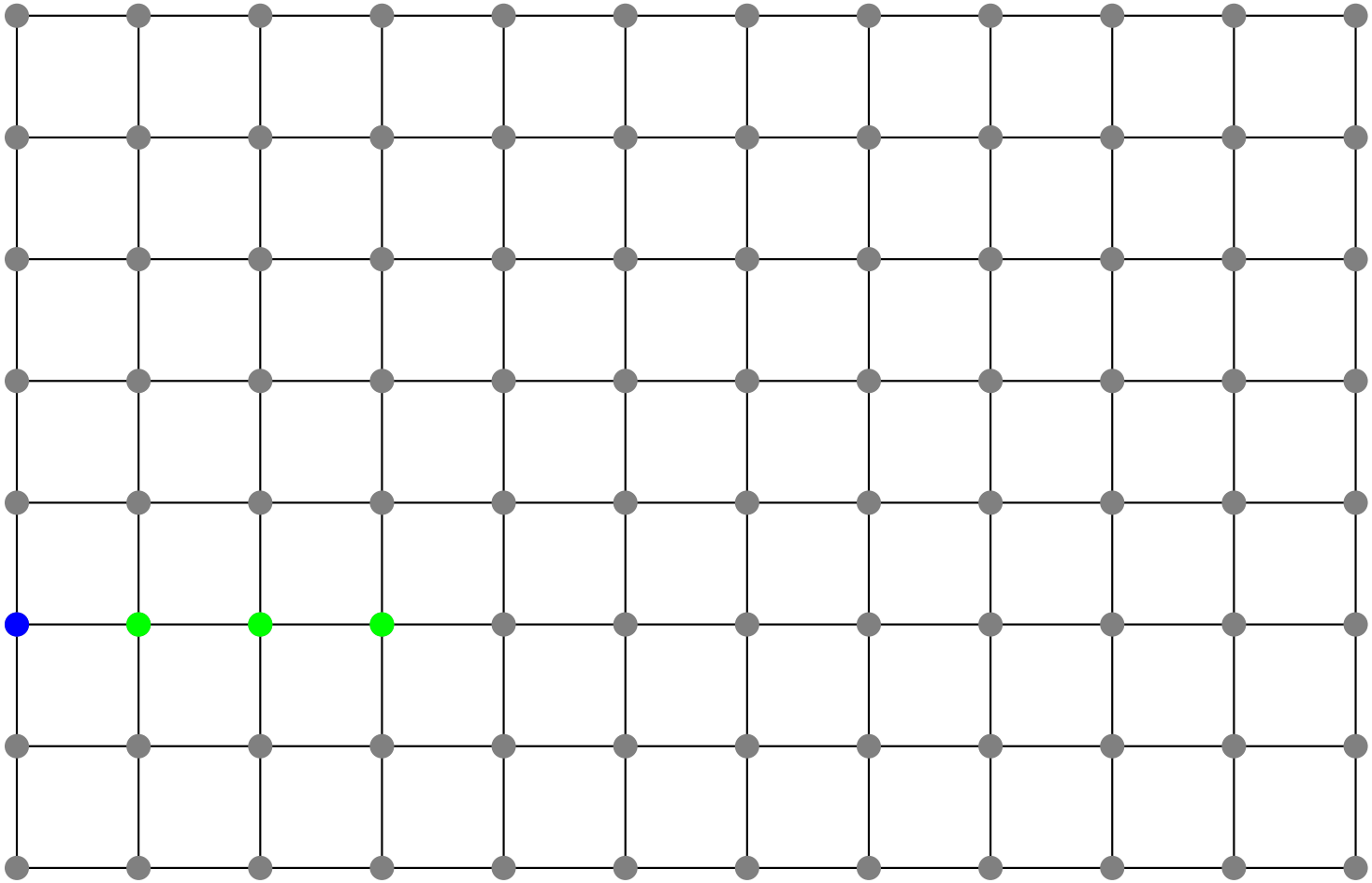
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



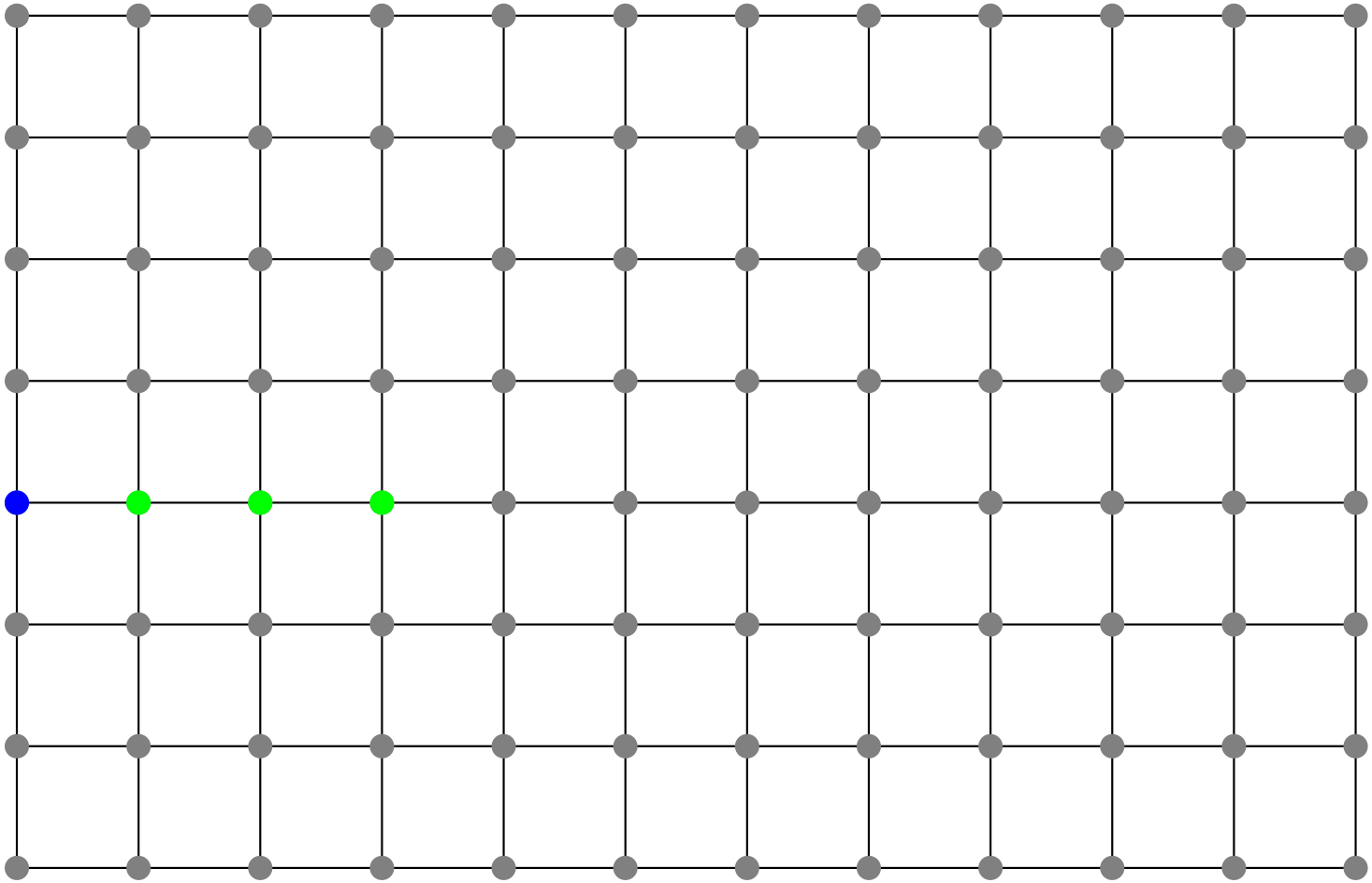
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



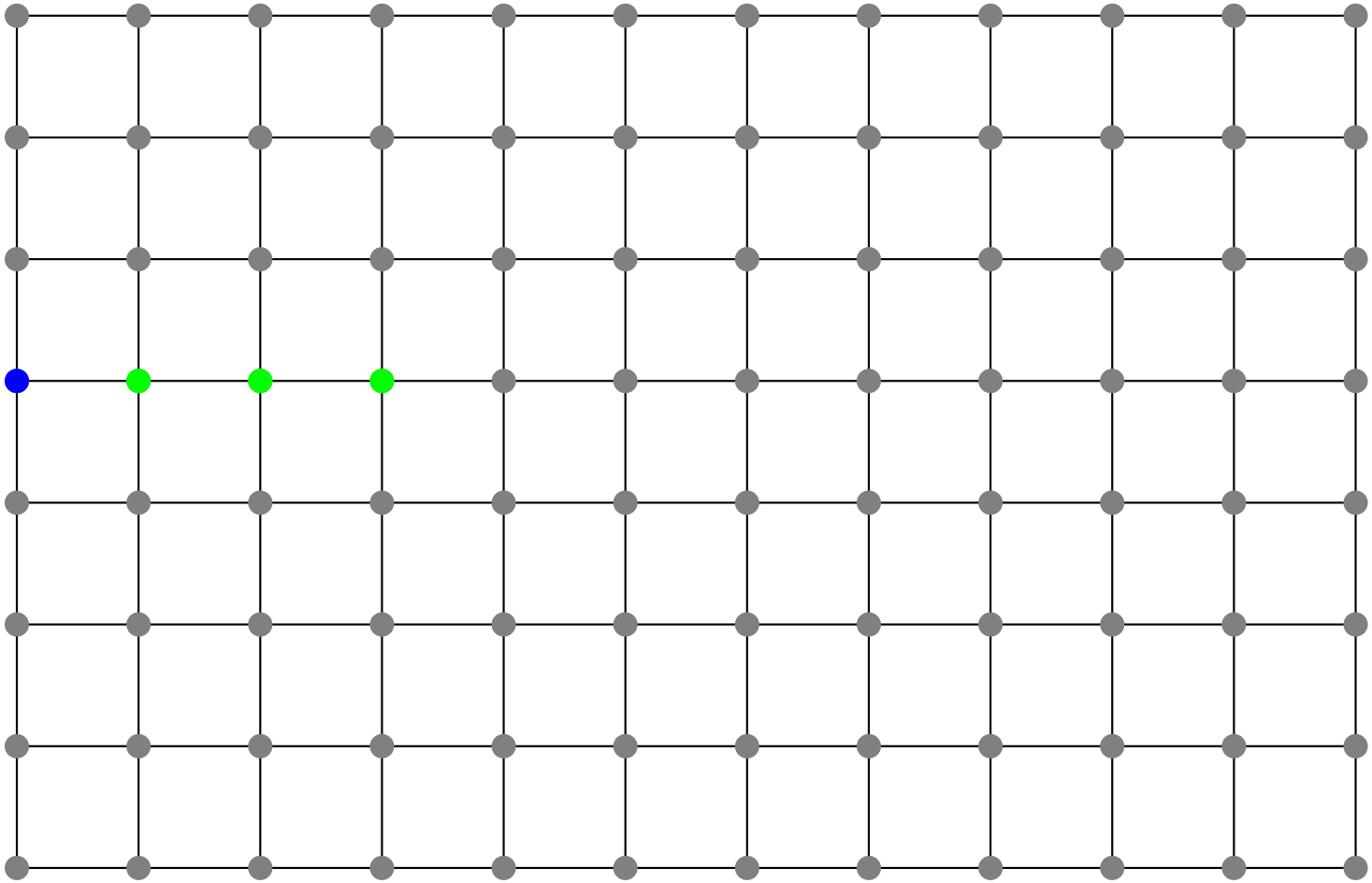
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



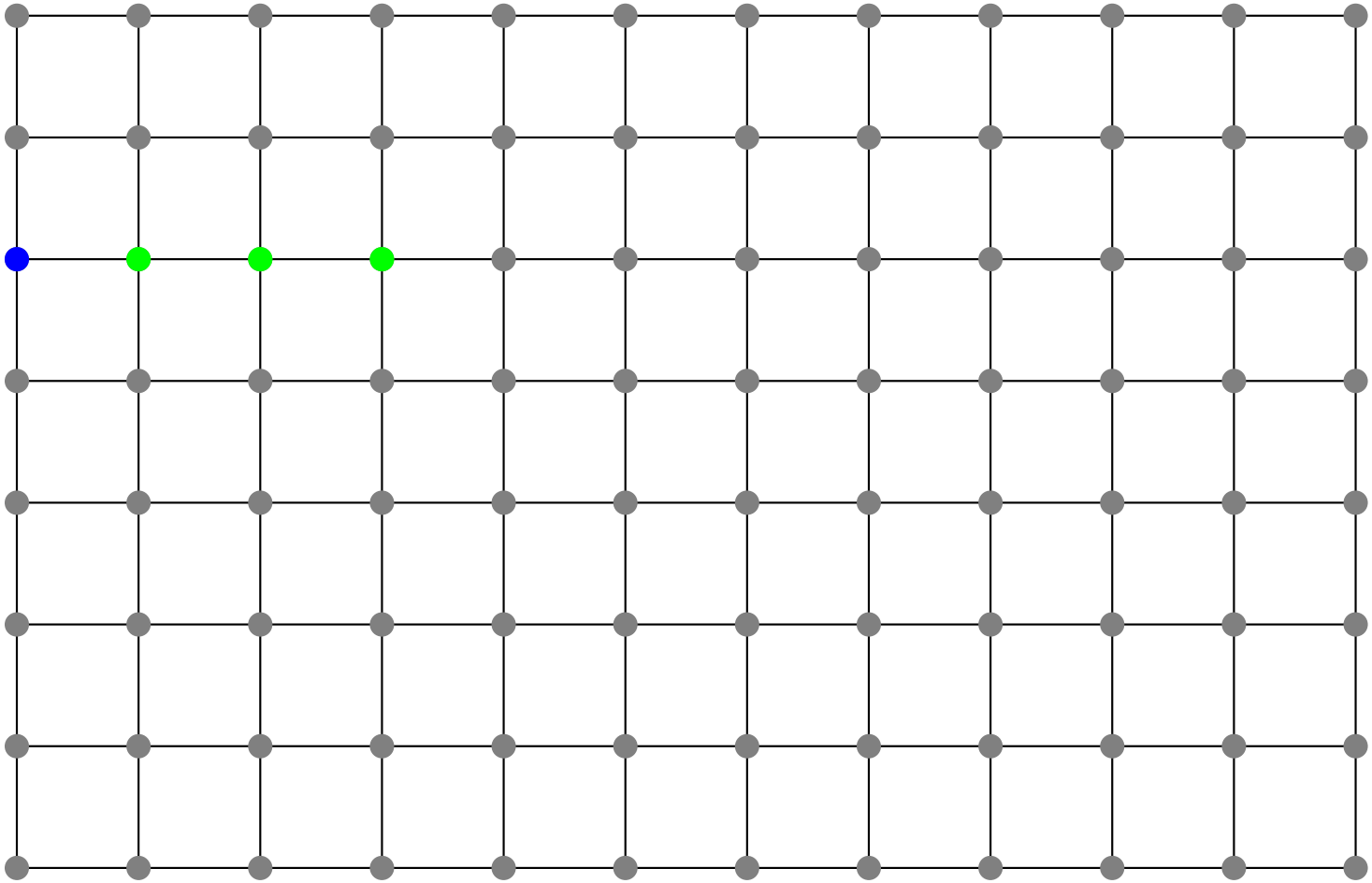
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



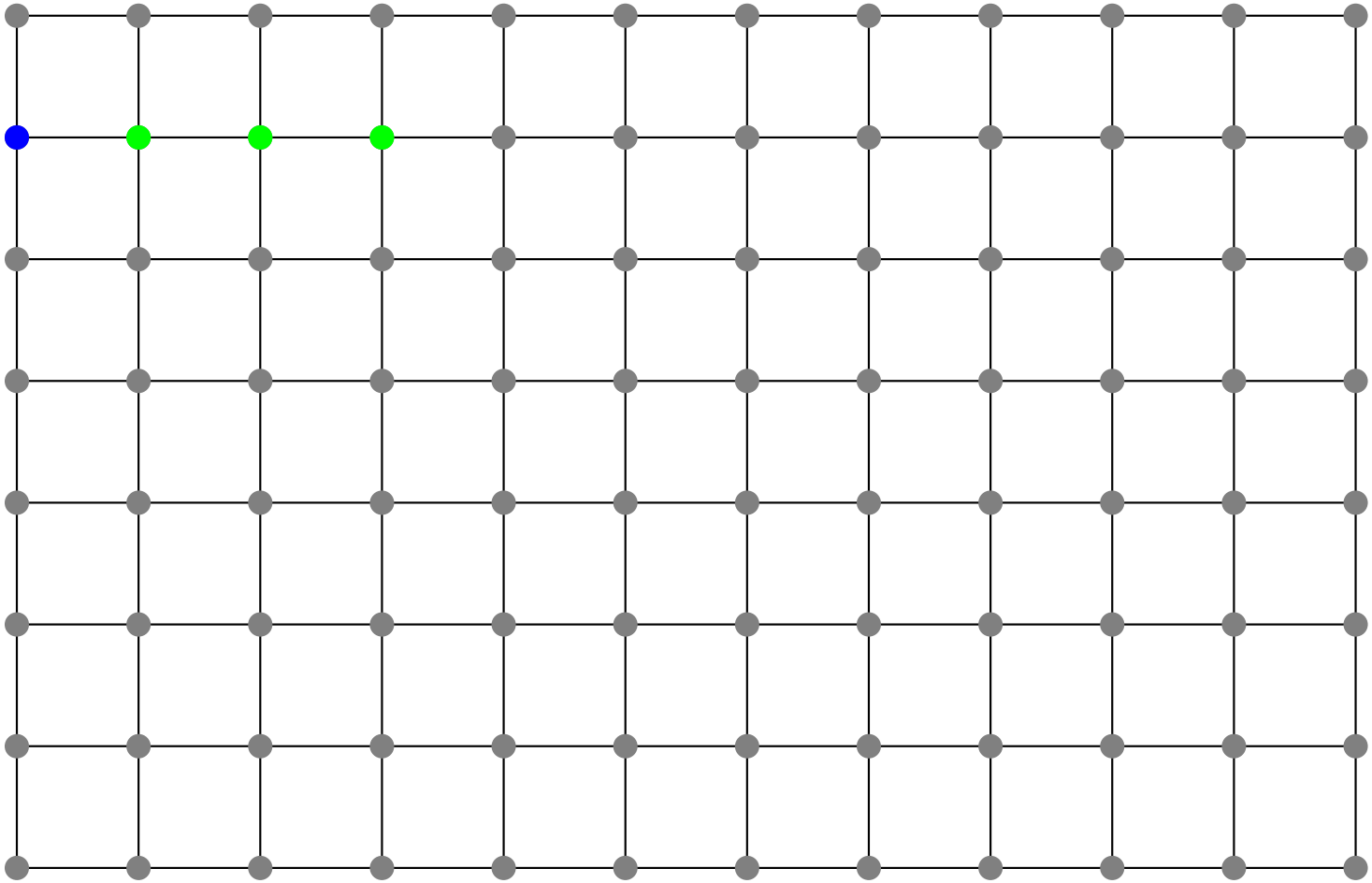
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



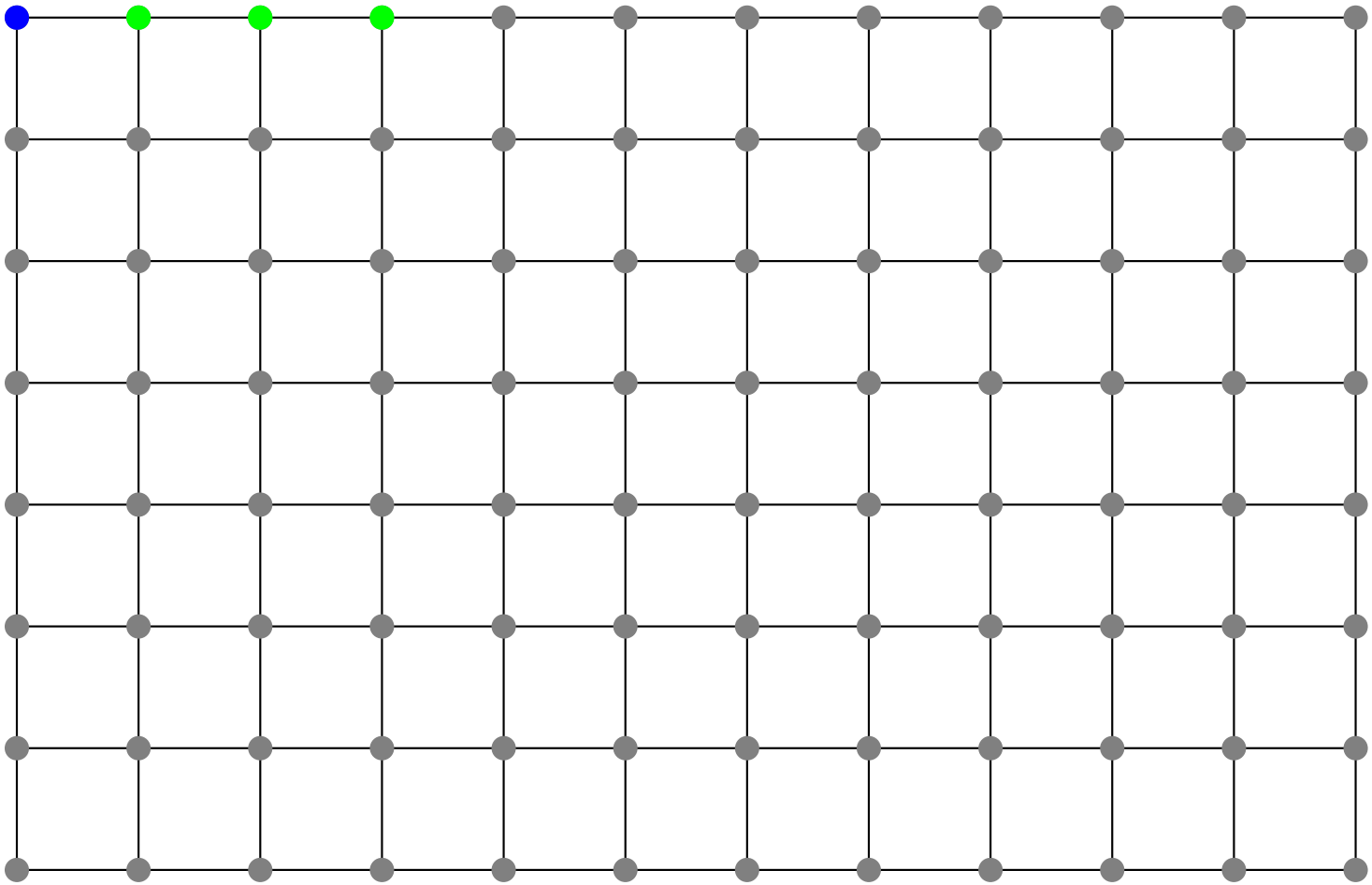
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



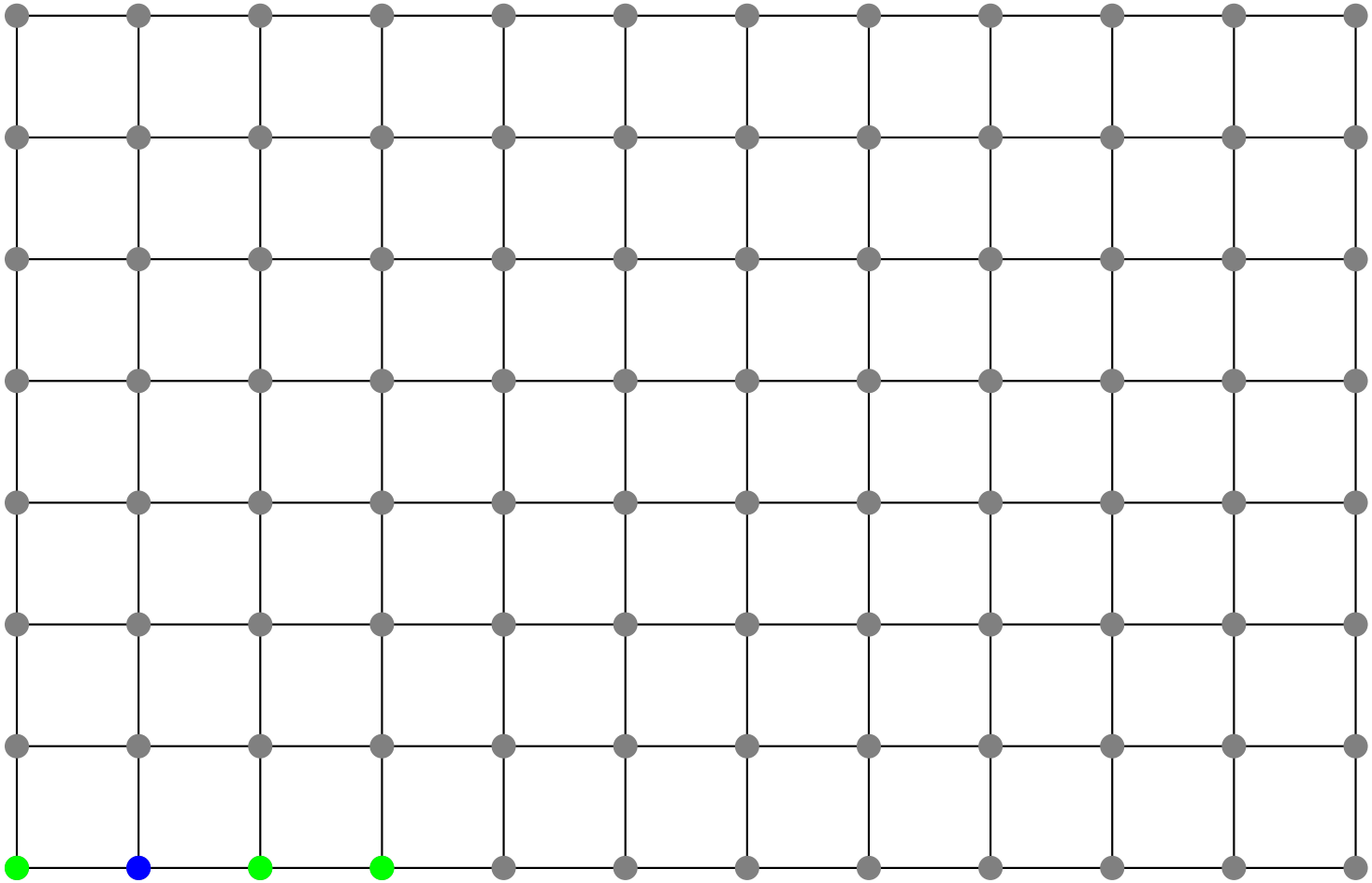
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



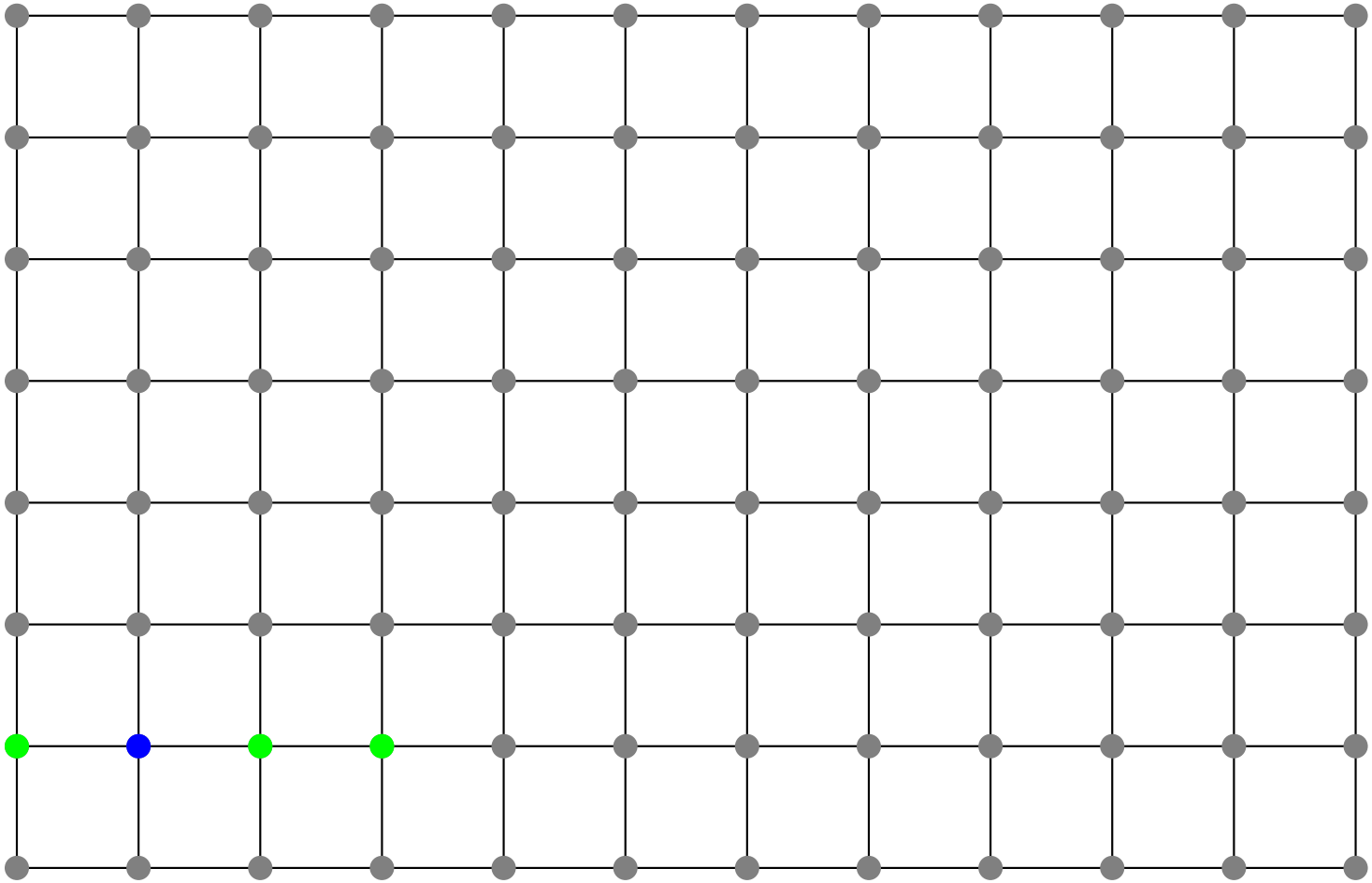
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



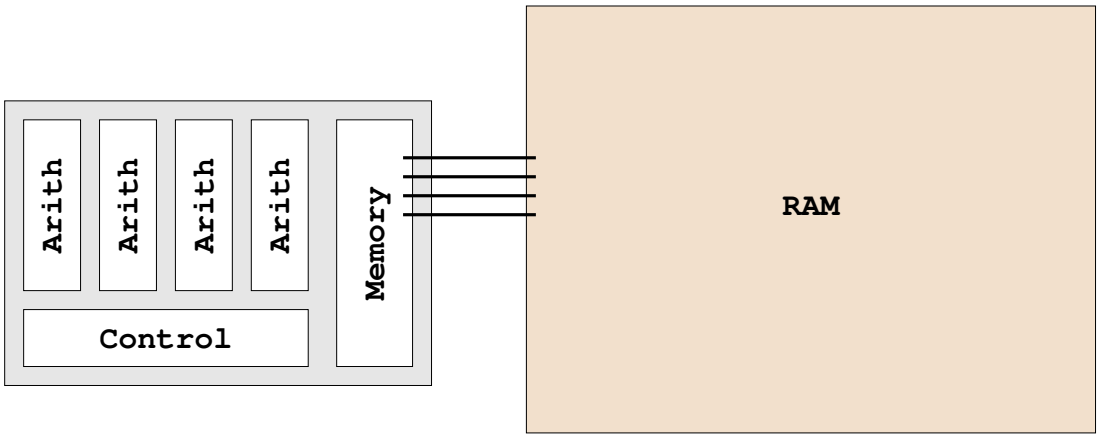
# Burst-Zugriffe im zweidimensionalen Beispiel

Zweite Fassung: Spaltenweise Verarbeitung.



# Übersicht

# Vektorisierung




Idee: Mehrere parallel geschaltete Rechenwerke.

1971 ILLIAC IV, 1997 Intel Pentium (MMX),
1999 Motorola PowerPC 7400 (AltiVec), 1999 Intel Pentium III (SSE)

# Vektorisierte Wellengleichung

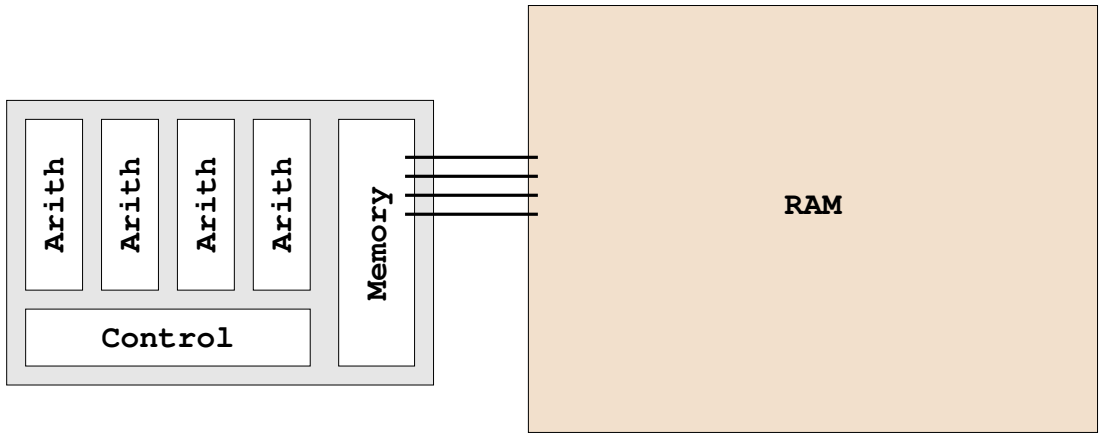
## Original:

```
for(i=1; i<=N; i++)
  x[i] += xdelta * v[i];

for(i=1; i<=N; i++)
  v[i] += vdelta * (x[i+1] + x[i-1] - 2*x[i]);
```

## Vektorrechner:

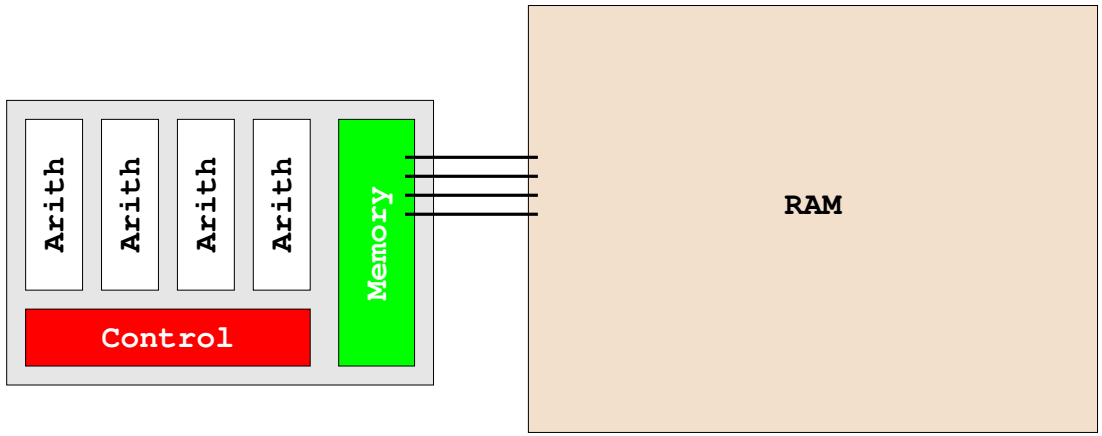
```
x4  = (float4 *) (x+1); v4  = (float4 *) (v+1);
x4l = (float4 *) x;     x4r = (float4 *) (x+2);

for(i=0; i<N/4; i++)
  x4[i] += xdelta * v4[i];

for(i=0; i<N/4; i++)
  v4[i] += vdelta * (x4r[i] + x4l[i] - 2*x4[i]);
```

# Vektorisierung im Beispiel



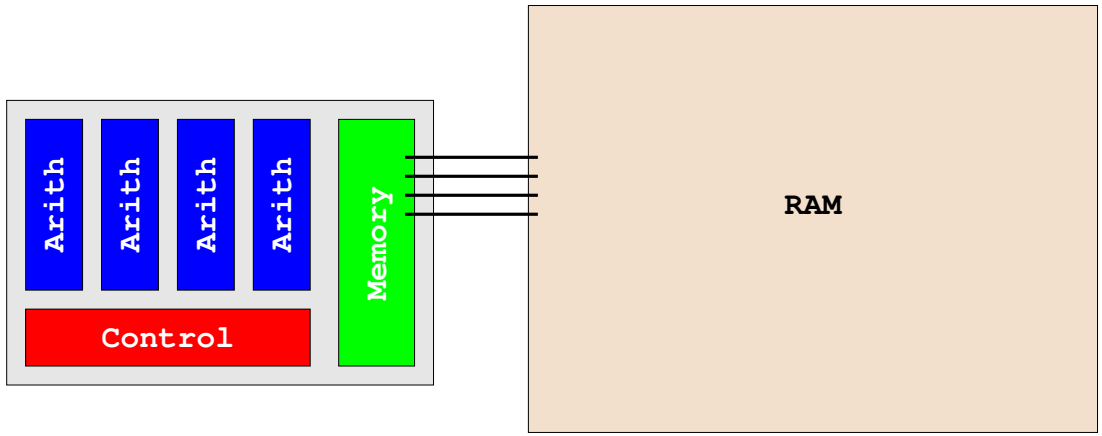

## Anwendung auf Wellengleichung:

# Vektorisierung im Beispiel



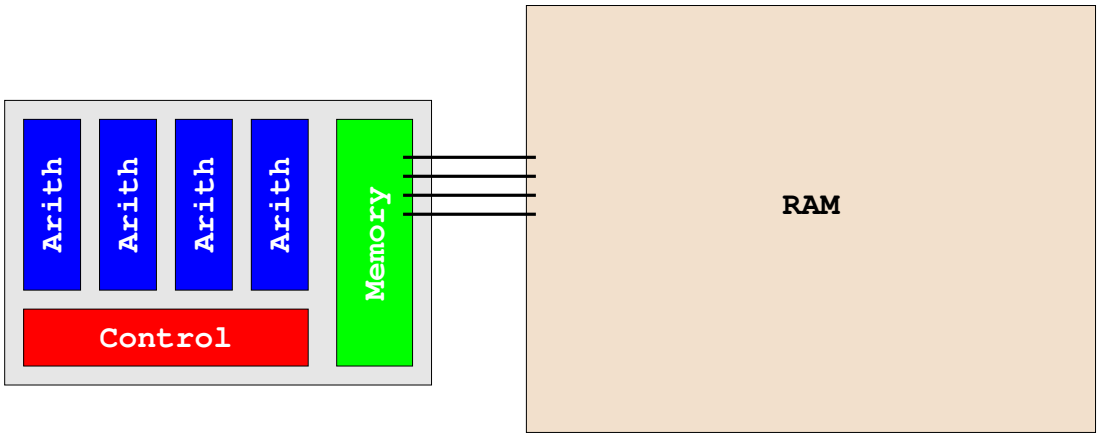

Anwendung auf Wellengleichung:

# Vektorisierung im Beispiel



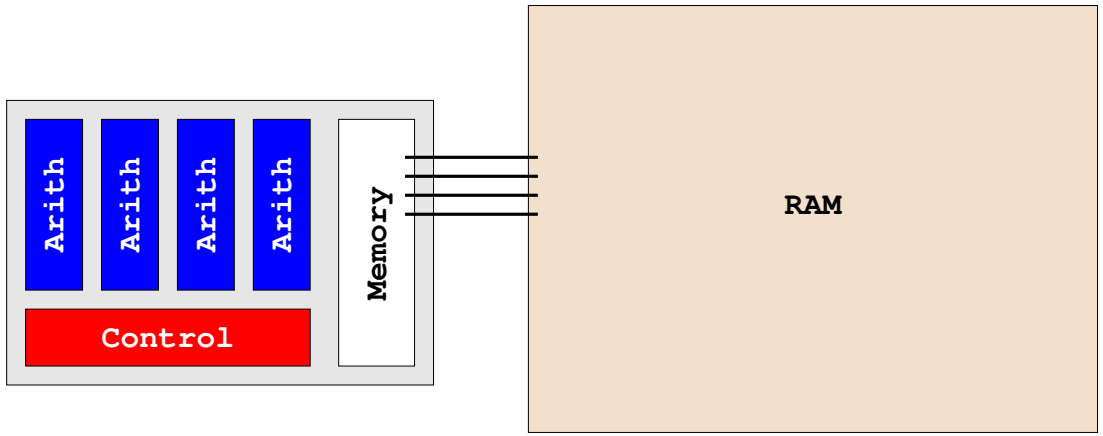

Anwendung auf Wellengleichung:

# Vektorisierung im Beispiel



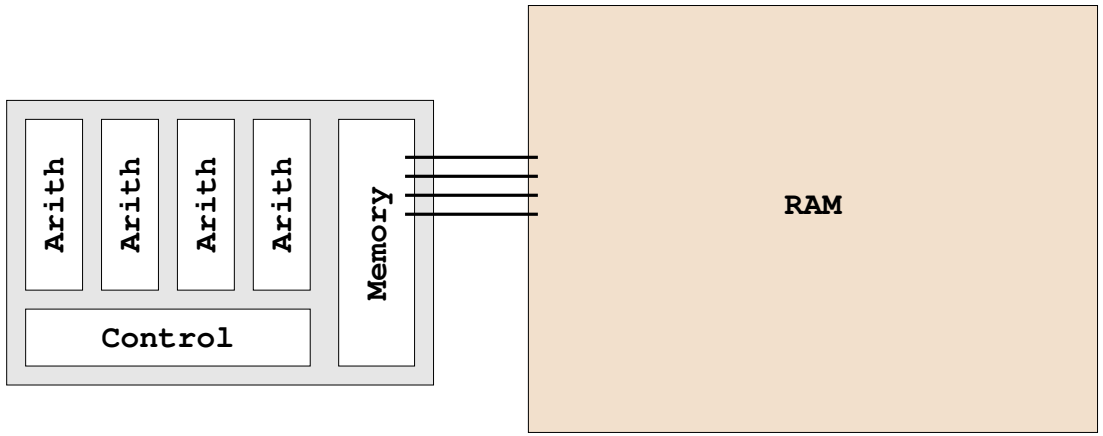

Anwendung auf Wellengleichung:

# Vektorisierung im Beispiel




Anwendung auf Wellengleichung:

# Vektorisierung im Beispiel

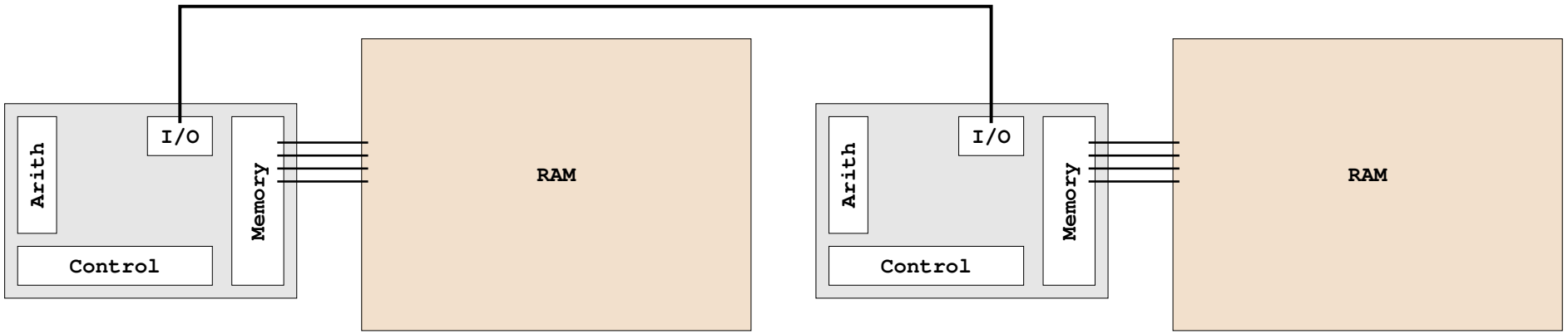



## Anwendung auf Wellengleichung:

# Vor- und Nachteile

+ Bei geeigneten Aufgaben einfach anwendbar.
+ Hoher Geschwindigkeitsgewinn (SSE $4\times$, AVX $8\times$, AVX512 $16\times$).
+ Manche Compiler vektorisieren automatisch.
- Nur benutzbar, falls identische Folgen von Operationen auf große Datenmengen angewendet werden müssen.

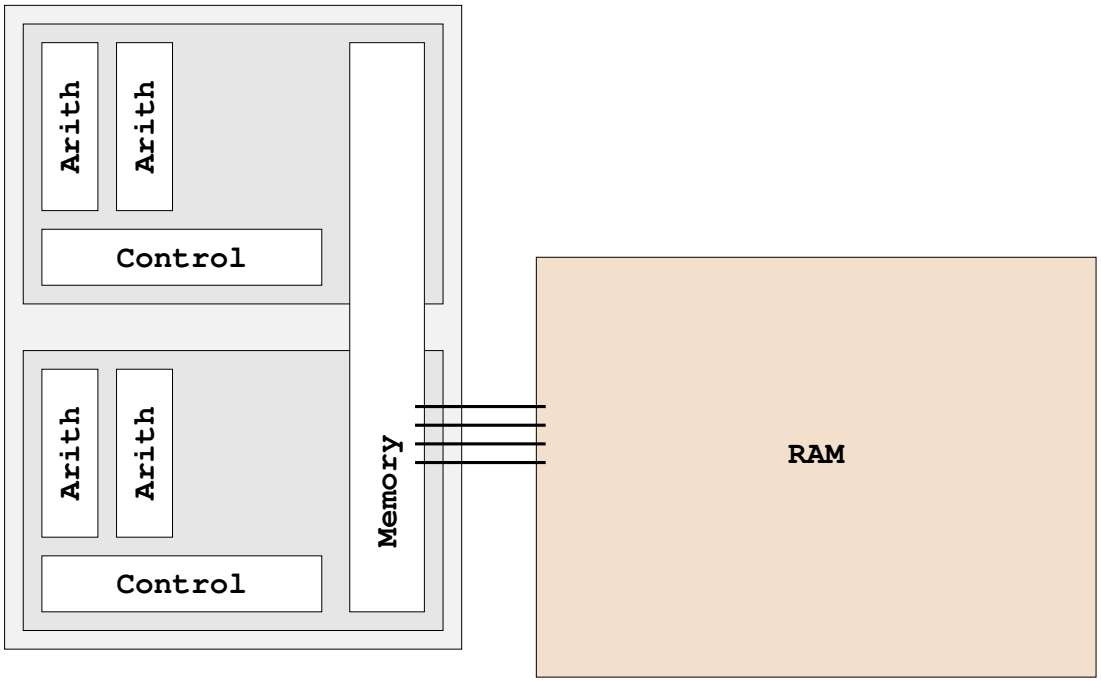
# Parallelisierung




Idee: Verteile Arbeit auf mehrere Computer, die über ein Netzwerk miteinander Daten austauschen.

Im Kontext der Informatik angeblich schon 1842 von F. L. Menabrea in Verbindung mit der *Analytical Engine* erwähnt.

# Mehrkernprozessor




Idee: Mehrere Prozessorkerne teilen eine Verbindung zur „Außenwelt“.

2005 AMD Athlon X2, 2006 Intel Core Duo

# OpenMP

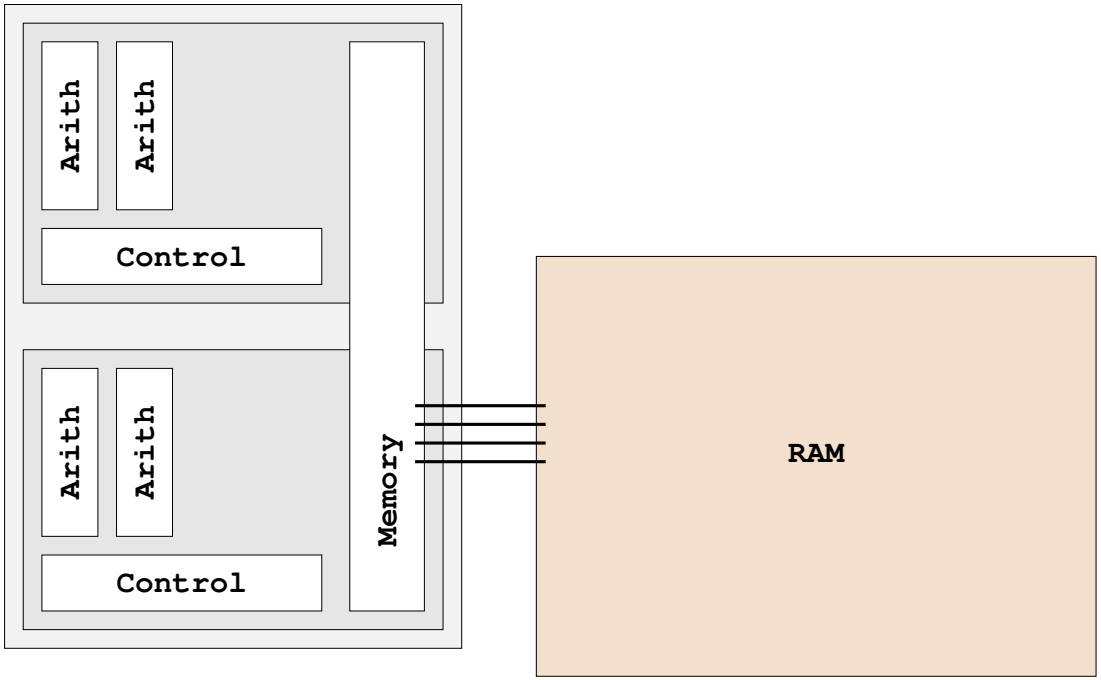
## Original:

```
for(i=1; i<=N; i++)
  x[i] += xdelta * v[i];

for(i=1; i<=N; i++)
  v[i] += vdelta * (x[i+1] + x[i-1] - 2*x[i]);
```

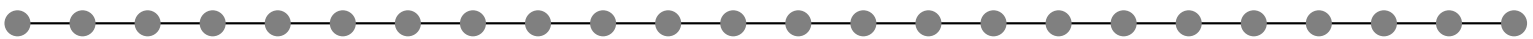
## Mit OpenMP für Mehrkernprozessoren:

```
#pragma omp parallel for
for(i=1; i<=N; i++)
  x[i] += xdelta * v[i];

#pragma omp parallel for
for(i=1; i<=N; i++)
  v[i] += vdelta * (x[i+1] + x[i-1] - 2*x[i]);
```

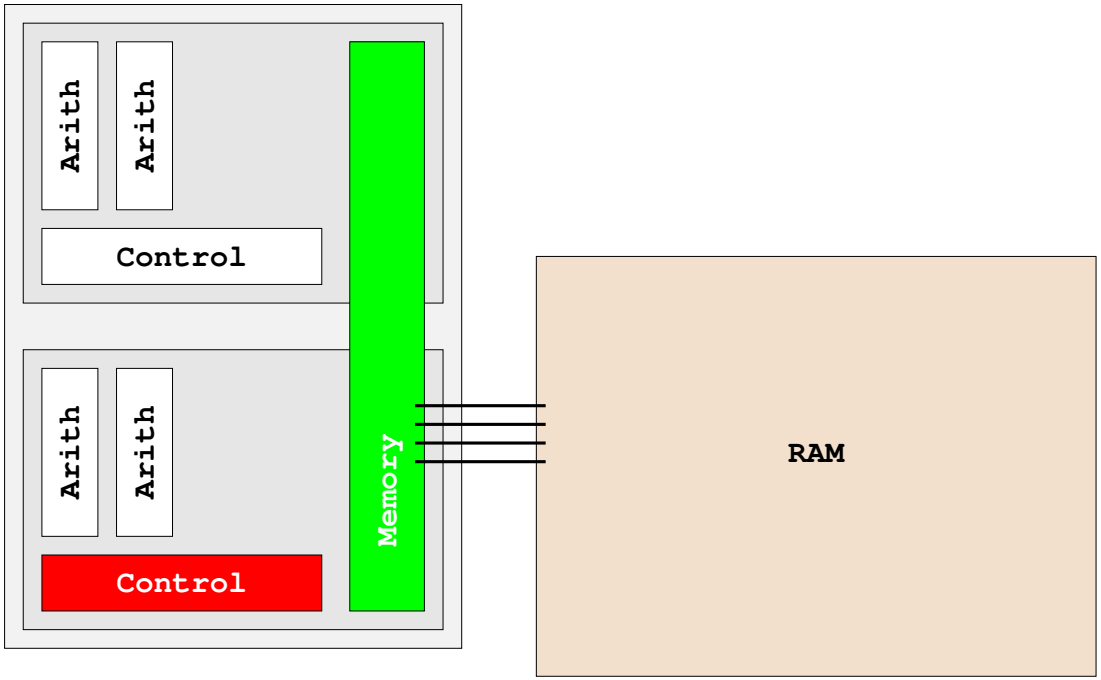
# Mehrkernprozessor im Beispiel



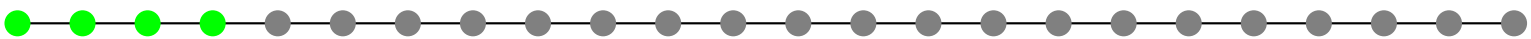

## Anwendung auf Wellengleichung:



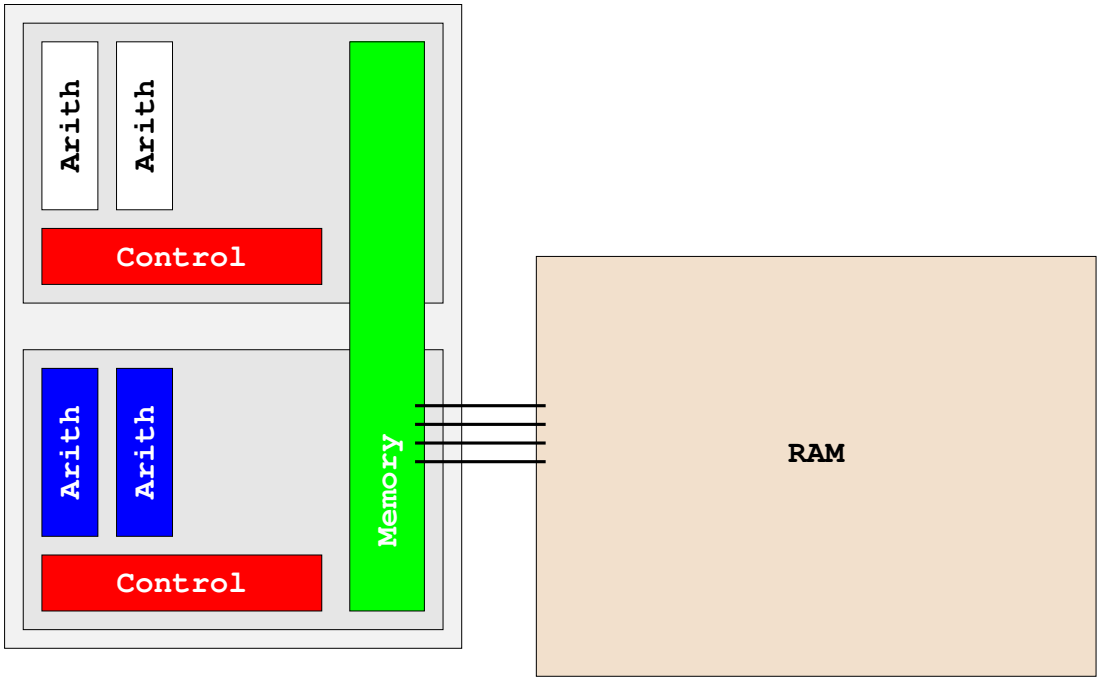
# Mehrkernprozessor im Beispiel



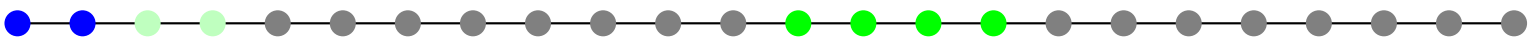

Anwendung auf Wellengleichung:



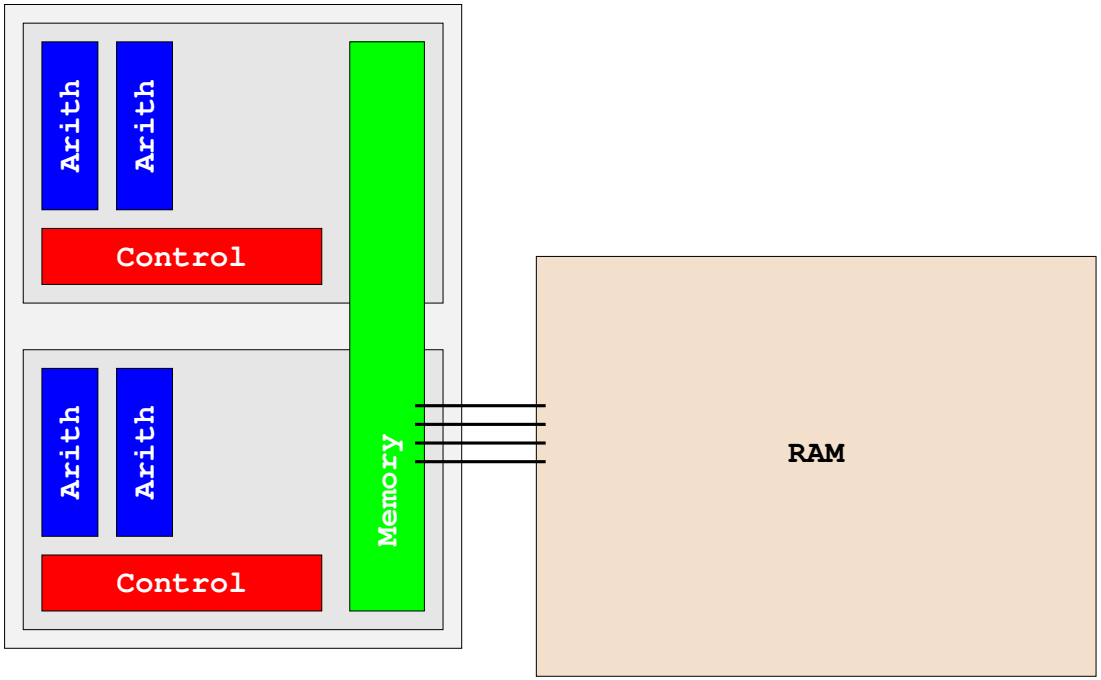
# Mehrkernprozessor im Beispiel



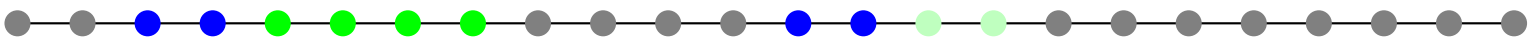

Anwendung auf Wellengleichung:



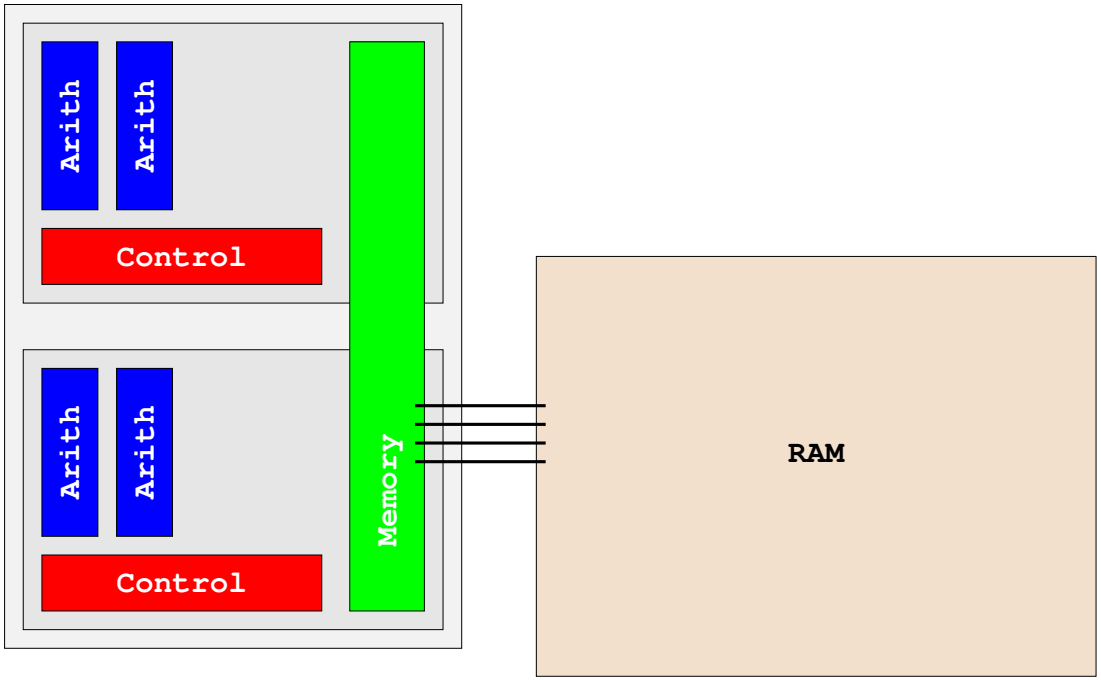
# Mehrkernprozessor im Beispiel



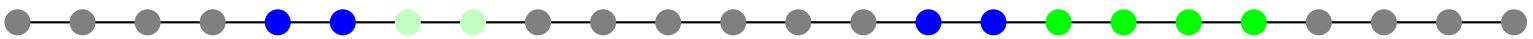

Anwendung auf Wellengleichung:



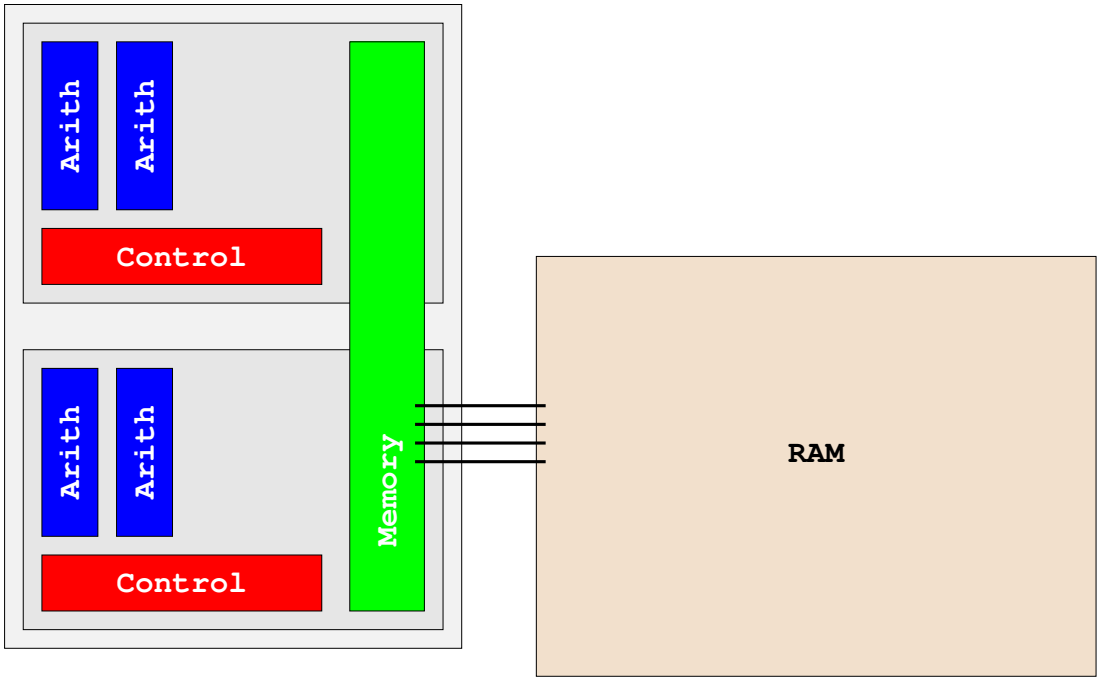
# Mehrkernprozessor im Beispiel



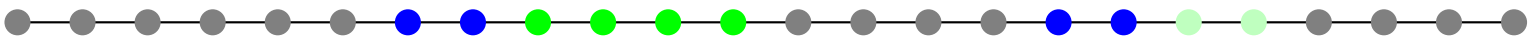

Anwendung auf Wellengleichung:



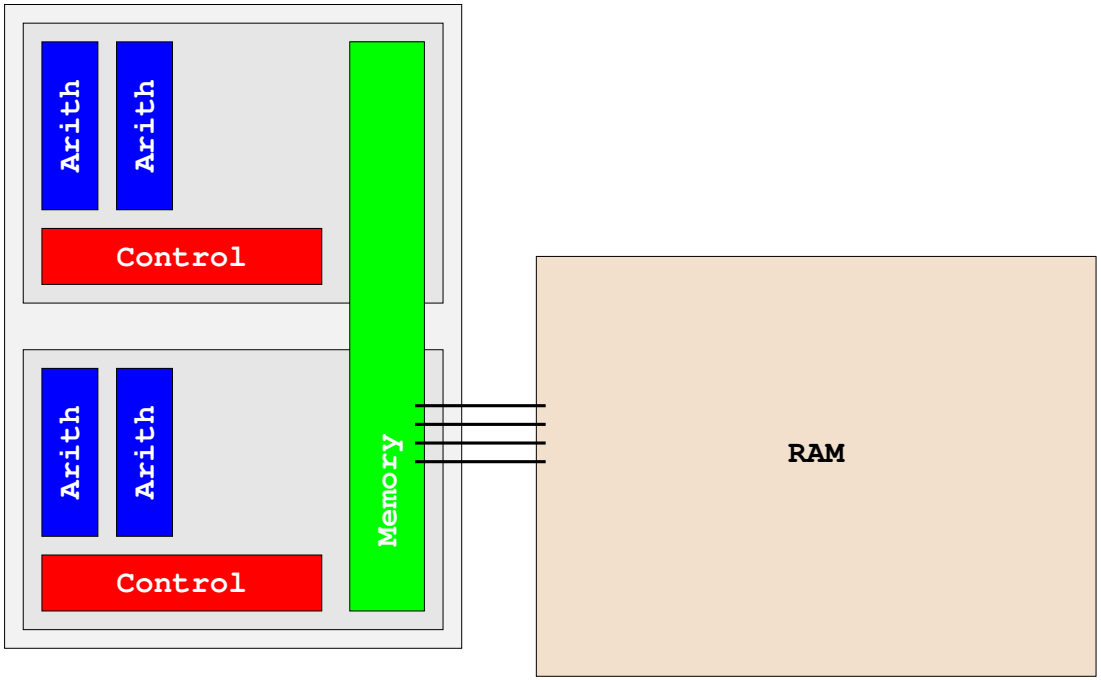
# Mehrkernprozessor im Beispiel



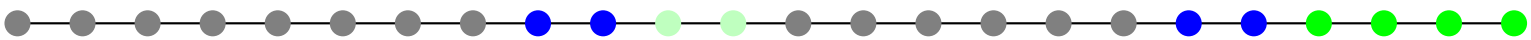

Anwendung auf Wellengleichung:



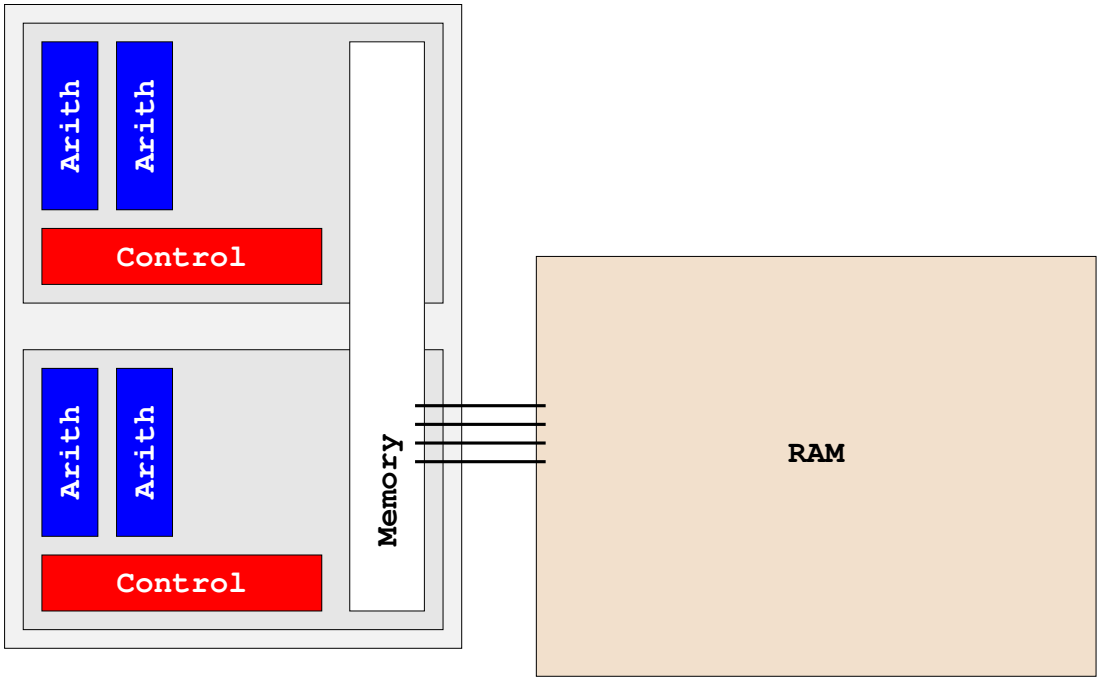
# Mehrkernprozessor im Beispiel



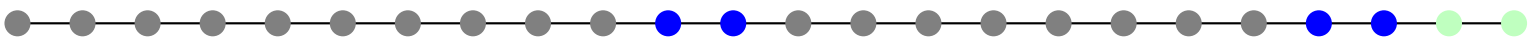

Anwendung auf Wellengleichung:



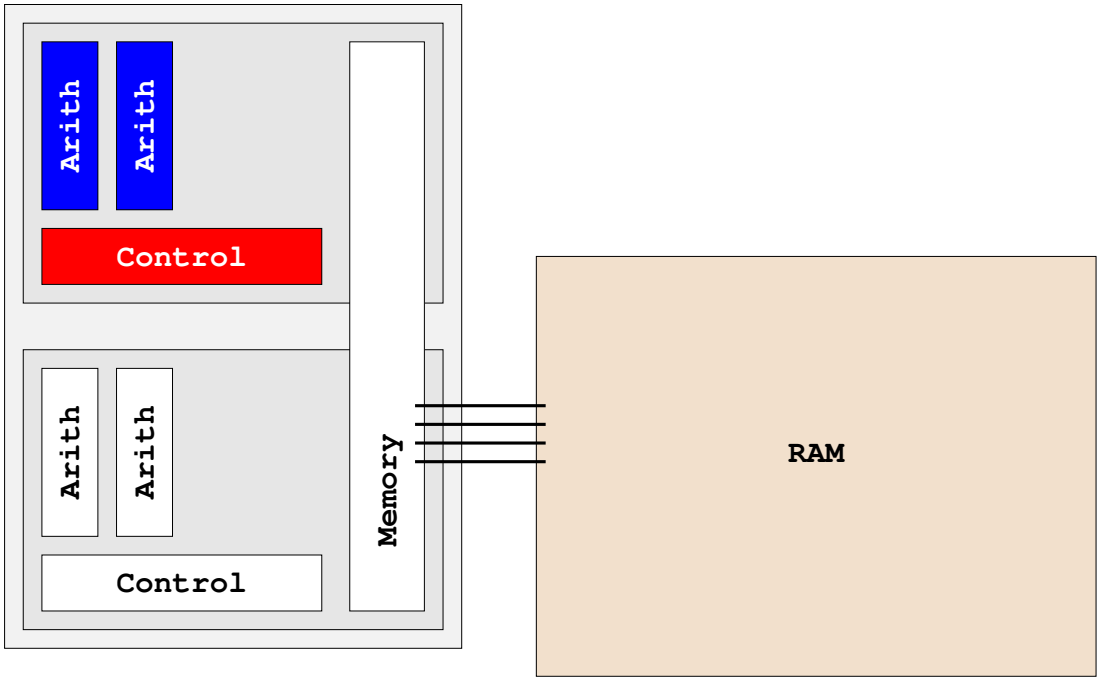
# Mehrkernprozessor im Beispiel



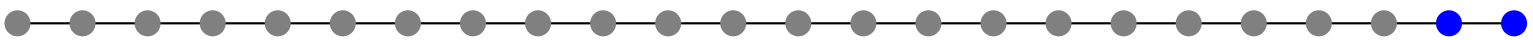

Anwendung auf Wellengleichung:



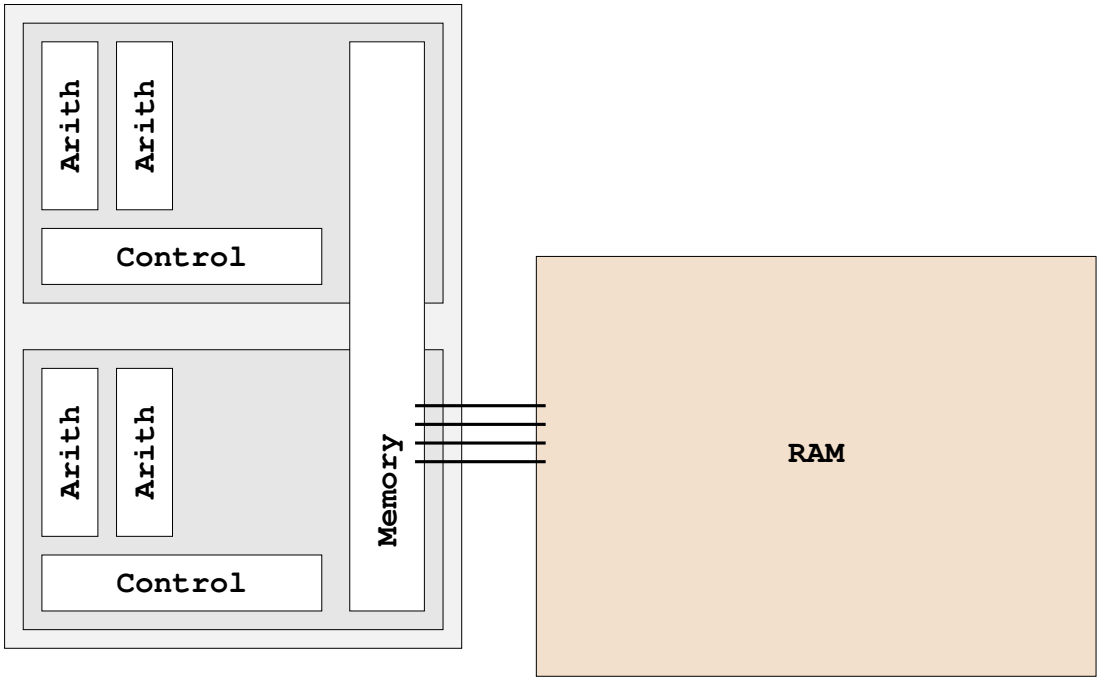
# Mehrkernprozessor im Beispiel



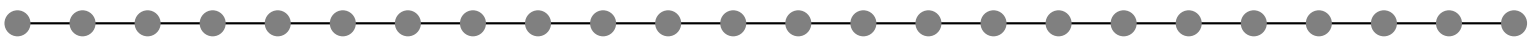

Anwendung auf Wellengleichung:



# Mehrkernprozessor im Beispiel



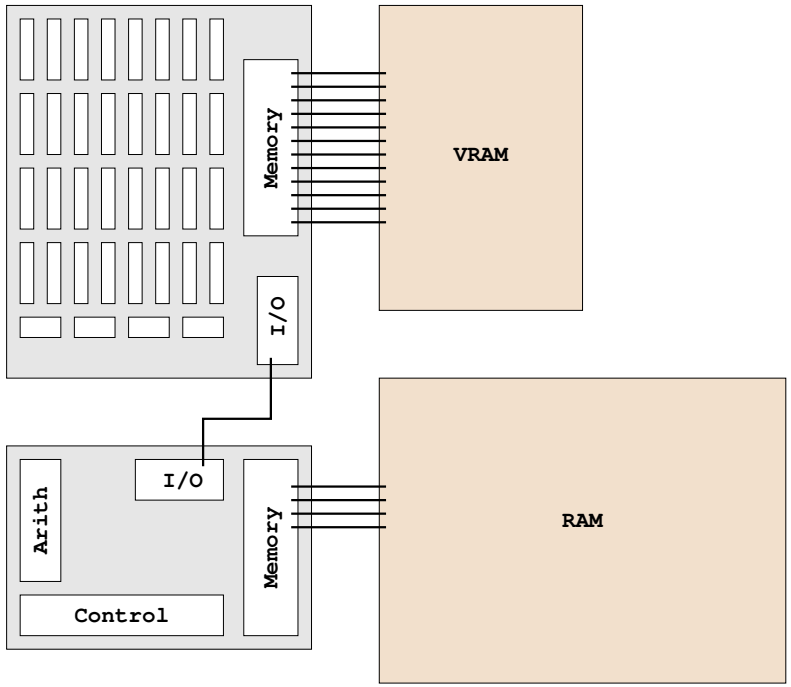

## Anwendung auf Wellengleichung:



# Vor- und Nachteile

+ Sehr flexibel.
+ Häufig leicht anwendbar.
+ Gute Beschleunigung bei rechenintensiven Anwendungen.
- Bei speicherintensiven Anwendungen kaum Vorteile.
- Synchronisation potentiell schwierig.

# Grafikkarten



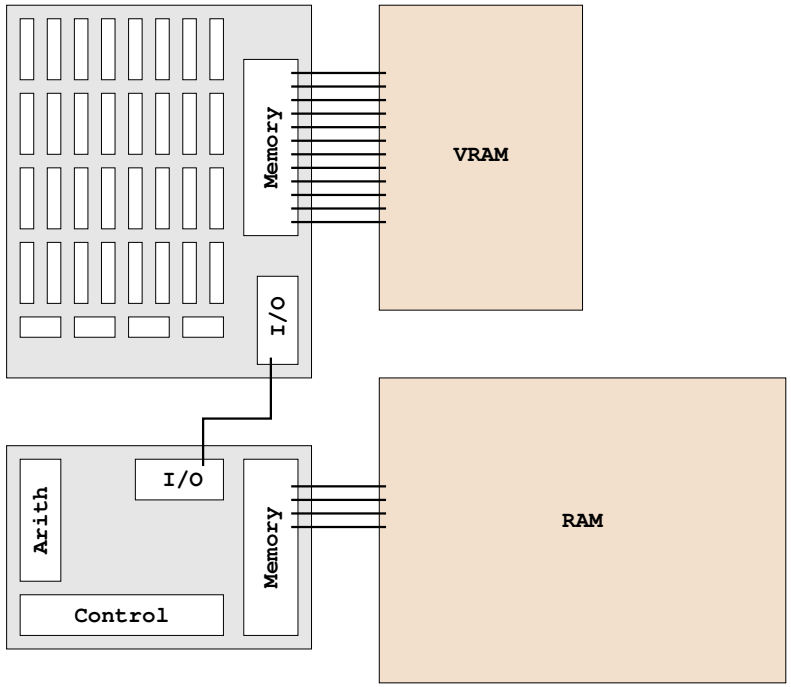

Idee: Nutze hohe Rechenleistung moderner Grafikkarten für „ernsthafte“ Anwendungen.

# OpenCL

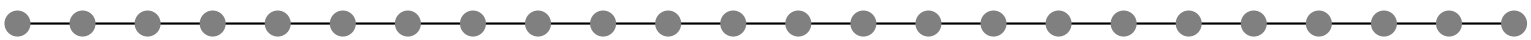
```
__kernel void update_x(__global float *x,
                       __global const float *v)
{
  int i = get_global_id(0) + 1;
  x[i] += xdelta * v[i];
}
__kernel void update_v(__global const float *x,
                       __global float *v)
{
  int i = get_global_id(0) + 1;
  v[i] += vdelta * (x[i+1] + x[i-1] - 2*x[i]);
}
```

```
clEnqueueNDRangeKernel(queue, update_x, 1, 0, &n,
                       &blksize, 0, 0, 0);
clEnqueueNDRangeKernel(queue, update_v, 1, 0, &n,
                       &blksize, 0, 0, 0);
```

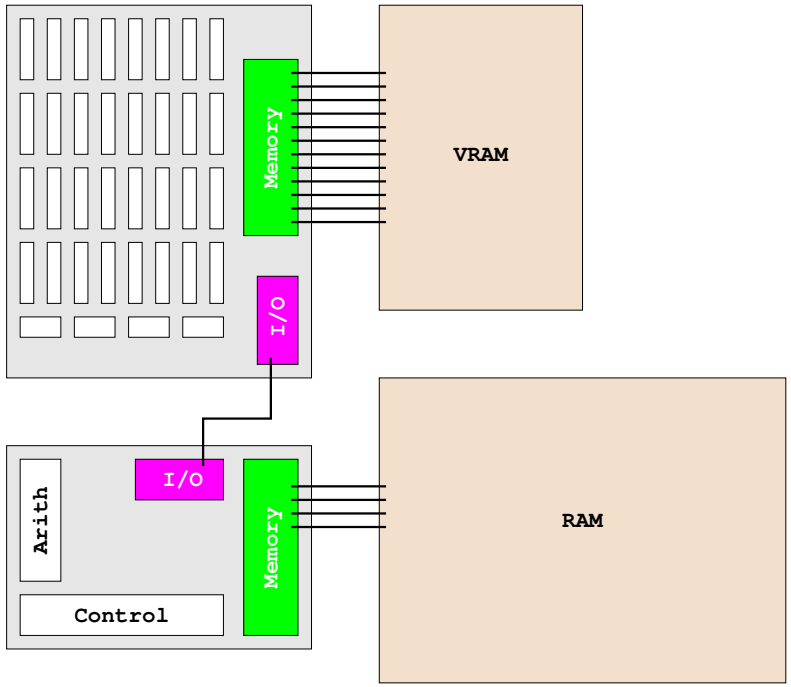
# Grafikkarte im Beispiel



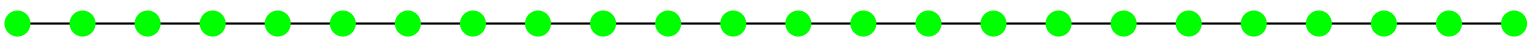

## Anwendung auf Wellengleichung:



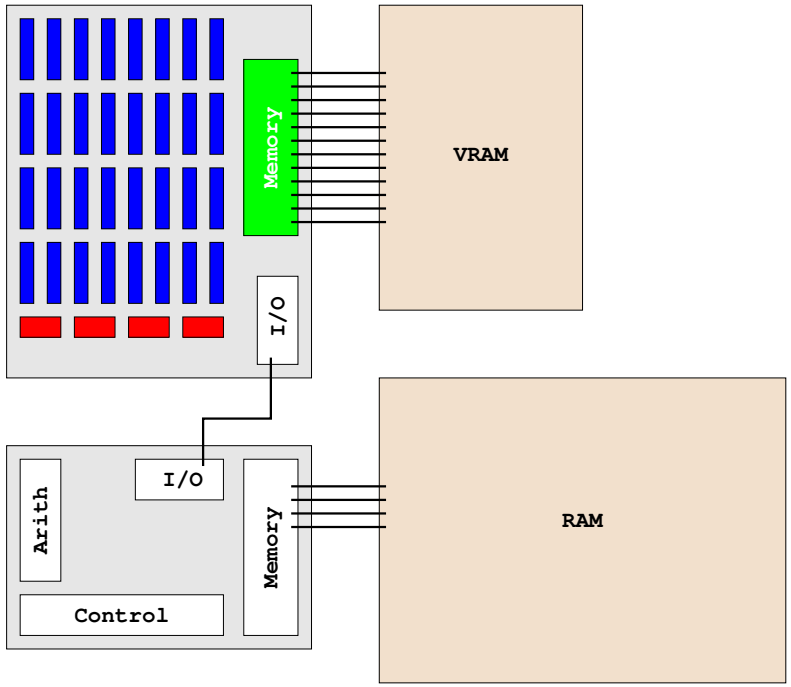
# Grafikkarte im Beispiel



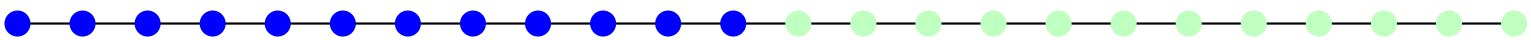

## Anwendung auf Wellengleichung:



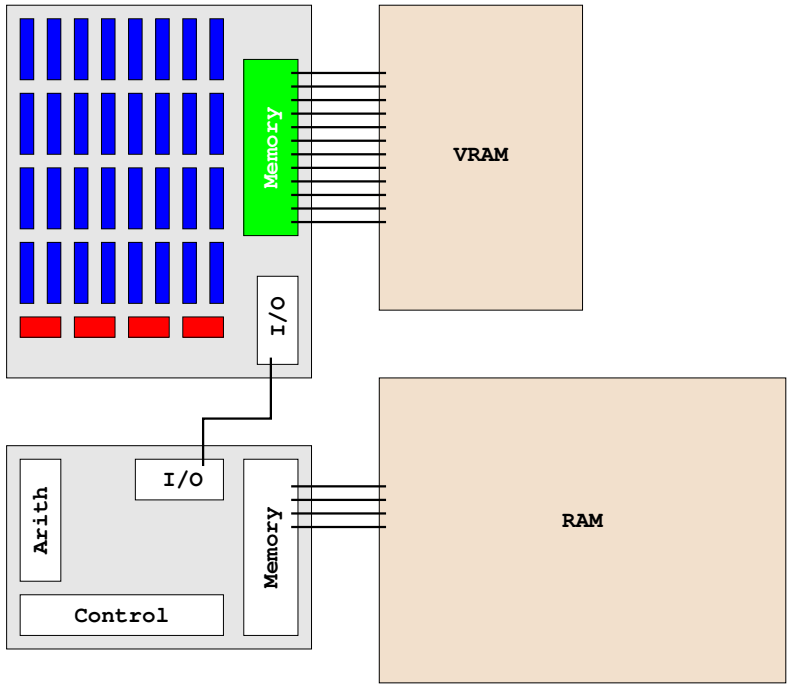
# Grafikkarte im Beispiel



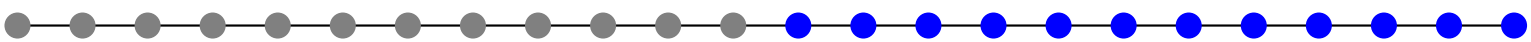

Anwendung auf Wellengleichung:



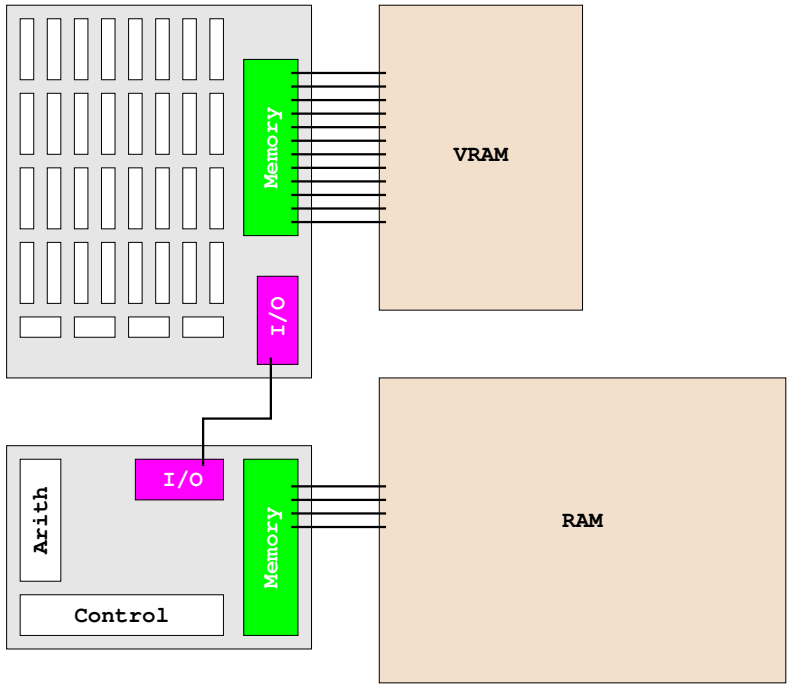
# Grafikkarte im Beispiel



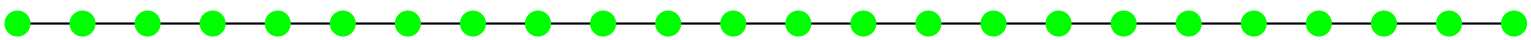

## Anwendung auf Wellengleichung:



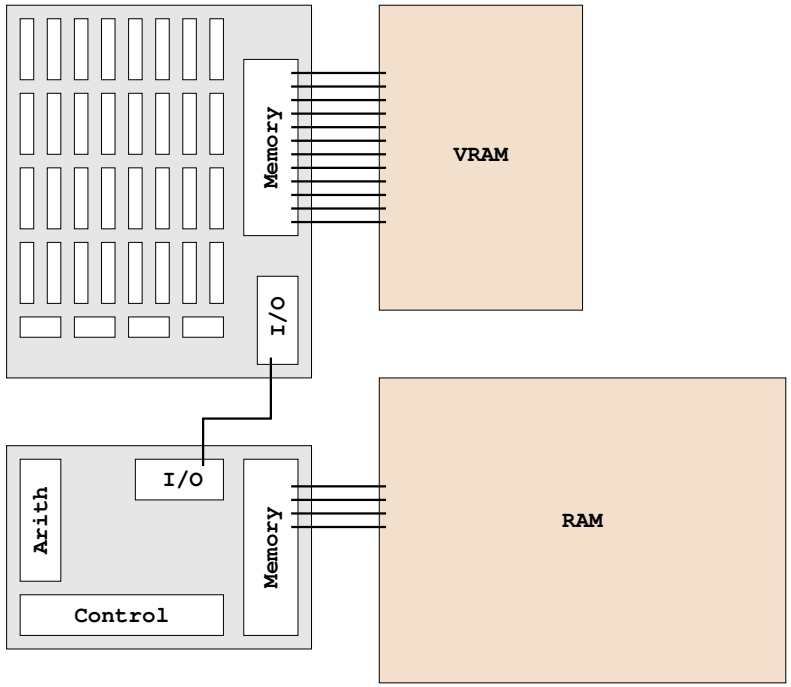
# Grafikkarte im Beispiel



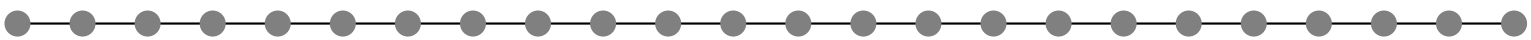

## Anwendung auf Wellengleichung:



# Grafikkarte im Beispiel




## Anwendung auf Wellengleichung:



# Vor- und Nachteile

+ Sehr hohe Rechenleistung.
+ Sehr schneller Speicherzugriff.
- Speicherzugriff trotzdem oft nicht schnell genug.
- Transfer zwischen Grafikkarte und Prozessor langsam.
- Konkurrierende Standards für die Programmierung.

# Zusammenfassung

Bei geschickter Programmierung bietet ein moderner PC eine Rechenleistung, die noch vor wenigen Jahren Supercomputern vorbehalten war.

Nutzbar wird sie durch

- Vermeidung von Datenabhängigkeiten und Sprüngen (superskalare Prozessoren),
- Geeignete Organisation von Speicherzugriffen (Caches),
- Einsatz von Vektor-Rechenwerken,
- Parallelisierung (Mehrkernprozessoren),
- Einsatz von Grafikkarten.