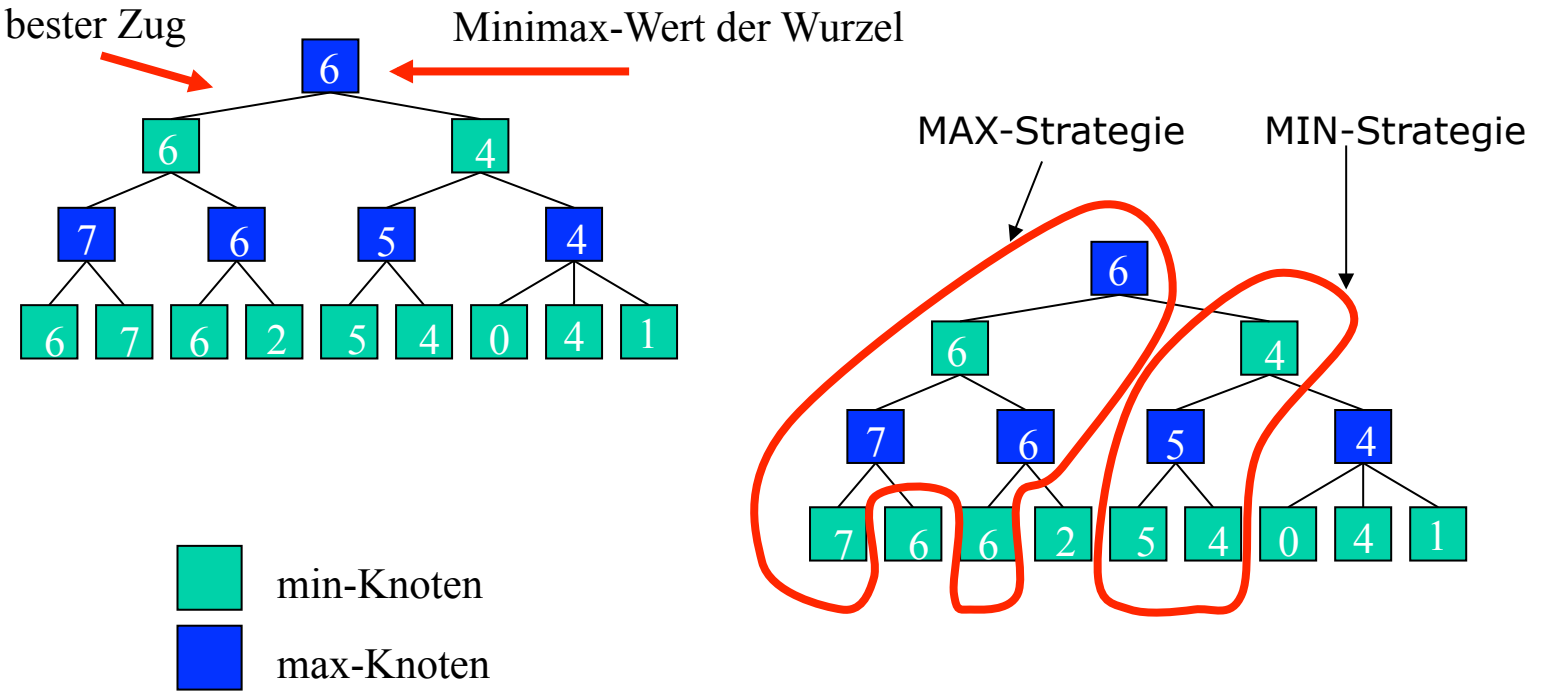
TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computerschach

Grundlagen I

Was macht ein Schachprogramm eigentlich?

(II) Vorausschau



Minimax-Prinzip

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition: Im folgenden ist G = (T,h) ein **Spielbaum**, wenn T = (V,E) ein Baum ist und h : L(G) → {0,1} eine Funktion ist, die Blattknoten des Spielbaums G auf Zahlen abbildet. L(G) bezeichnet hier die Menge der Blattknoten von G.

Definition: Es gebe 2 Spieler MIN und MAX. MAX besitzt das Zugrecht auf den geraden Ebenen des Spielbaums, MIN auf den anderen. Hierdurch wird eine Spielerfunktion definiert: **p : V → {MAX, MIN}**

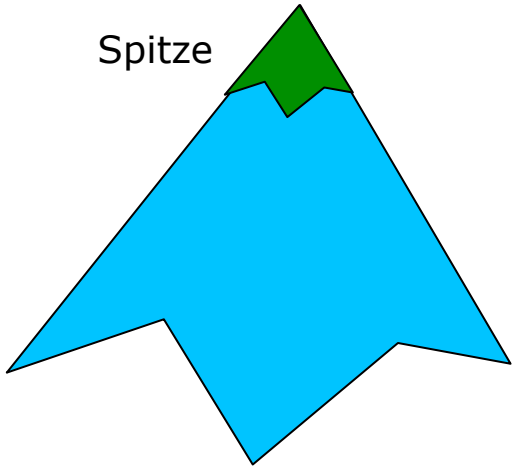
Definition: Sei G = ((V,E),h) ein Spielbaum. Sei $v \in V$ ein Knoten des Spielbaums G. Die Funktion **minimax: V → {0,1}** ist induktiv wie folgt definiert:

$$\text{minimax}(v) := \begin{cases} h(v), & \text{falls } v \in L(G) \\ \max\{\text{minimax}(v' \mid (v,v') \in E\}, & \text{falls } p(v) = \text{MAX} \\ \min\{\text{minimax}(v') \mid (v,v') \in E\}, & \text{falls } p(v) = \text{MIN} \end{cases}$$

Der Minimaxwert des Knotens v ist minimax(v). Der Minimaxwert der Wurzel eines Spielbaums G wird mit minimax(G) bezeichnet.

Schnell, schnell, schnell!

Der kritische Punkt ist die intelligente Vorausschau.



- **Man wähle einen geeigneten Teilbaum, der die aktuelle Stellung enthält. So groß wie möglich.**
- **Weise den künstlichen Endstellungen heuristische Werte zu!**
- **Werte aus!**

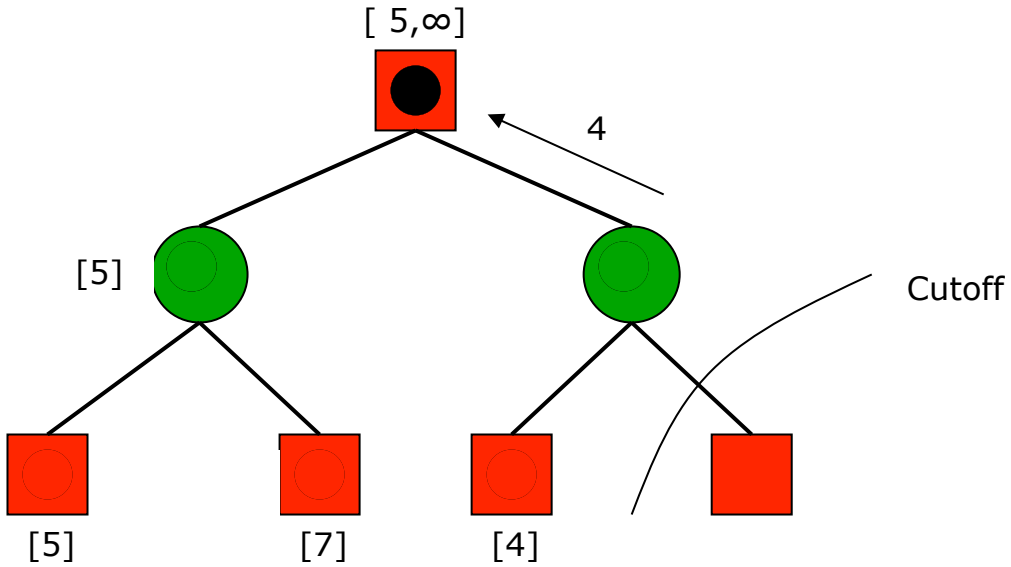
50 Jahre währende Erfahrung zeigt:

Der Spielbaum verhält sich wie ein Fehlerfilter!

*Je größer der Baum, desto besser der Filter. Der **Alphabeta-Algorithmus** spielt dabei eine wichtige Rolle.*

Alphabeta-Algorithmus

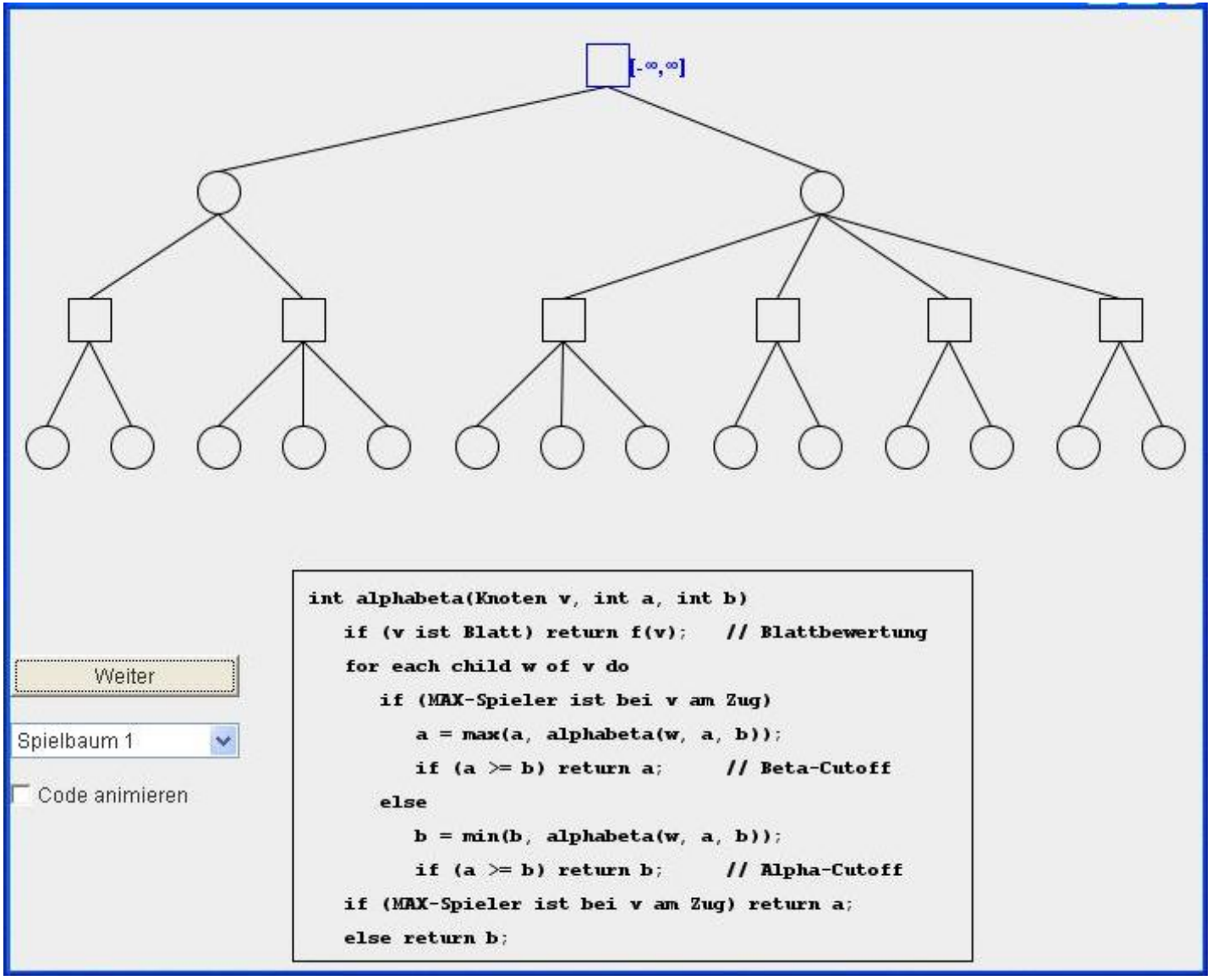
TECHNISCHE UNIVERSITÄT DARMSTADT



Miniaturbeispiel

Alphabeta-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT



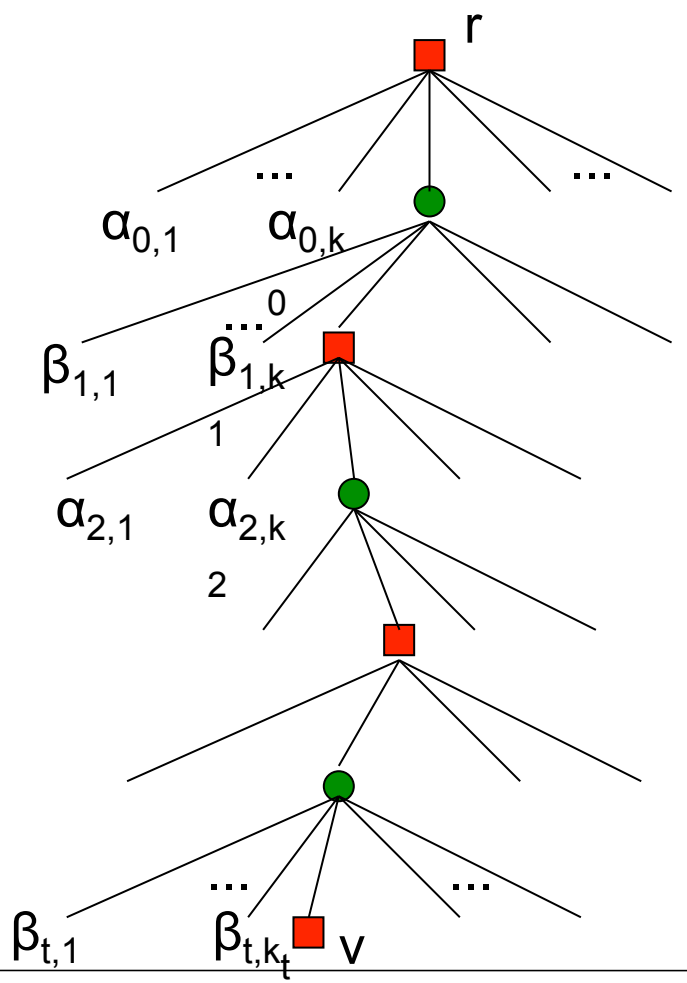
Alphabeta-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

Der Alphabetaalgorithmus:

•Tiefensuche

• wenn die Suche an Knoten v angelangt, können wir uns vorstellen, dass
  - der durchsuchte Teil des Suchbaums „links“ von dem Weg von der Wurzel r zu v liegt, und
  - der noch nicht durchsuchte Teil „rechts“ des Weges liegt.



Alphabeta-Algorithmus

Es seien $\alpha_{i,j}$ ($\beta_{i,j}$) die Werte der linken Nachbarknoten von MIN- (bzw. MAX)- Knoten auf dem Weg von r nach v.

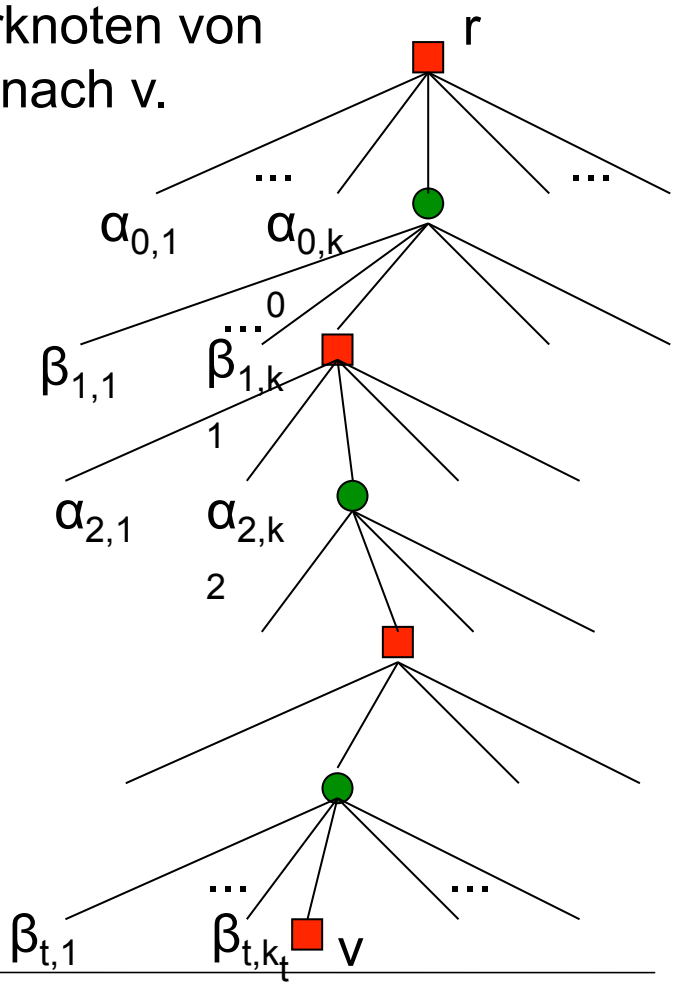
Sei

$\alpha(v) := \max_{i=0,...,t;\ i \text{ gerade}} \max_{j=1,...,k_i} \alpha_{i,j}$

und

$\beta(v) := \min_{i=0,...,t;\ i \text{ ungerade}} \min_{j=1,...,k_i} \beta_{i,j}$

Dann kann MAX bereits $\alpha(v)$ erreichen, MIN bereits $\beta(v)$. F(v) ist also nur noch von Interesse, wenn $\alpha(v) < F(v) < \beta(v)$. Wurde Alphabeta(r,$-\infty$,$\infty$) aufgerufen, wird der Alphabeta-Algorithmus für v mit Alphabeta(v,$\alpha(v)$,$\beta(v)$) aufgerufen.



Alphabeta-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

Beobachtung: α(v) und β(v) sind abhängig von dem bereits durchsuchten Teil des Suchbaums und damit von der Reihenfolger von Nachfolge-knoten.

Satz: Es sei G=(V,E,f) ein Spielbaum, v ein Knoten aus V, α(v) und β(v) die Schranken für v. Der Alphabeta-Algorithmus untersucht v genau dann, wenn α(v) < β(v).

Beweis: klar mit Überlegung von vorher und Cutoff-Bedingung im Algorithmus.

Alphabeta-Algorithmus, Negamax-Variante

TECHNISCHE UNIVERSITÄT DARMSTADT

```
int negamax(node v, int a, b)
{
  if (v ist Blatt) return f(v);      // f bewerte v jetzt aus Sicht des Ziehenden!
  erzeuge alle Nachfolger v_0 ... v_{b-1} von v
  for (i = 0; i < b; i++) {
        a = max(a, -negamax(v_i, -b, -a))
        if (a ≥ b) return a;        // cutoff
  }
  return a;
}
```

Lemma: Sei x = negamax(v,α,β) und F beschreibe den Minimax-Wert aus Sicht des Spielers, der bei v am Zug ist. Dann gilt:

$x \leq \alpha$	falls $F(v) \leq \alpha$
$x = F(v)$	falls $\alpha < F(v) < \beta$
$x \geq \beta$	falls $F(v) \geq \beta$

Beweis: Induktion über Höhe h des Spielbaums → Übung

Alphabeta-Algorithmus, Aufwand

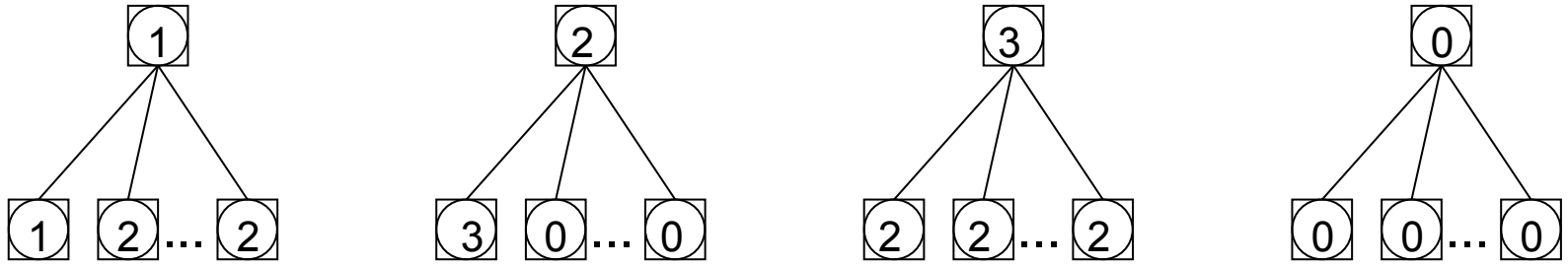
TECHNISCHE UNIVERSITÄT DARMSTADT

Sei G = (V,E,f) ein b/t-uniformer Spielbaum, d.h., alle inneren Knoten besitzen b Nachfolger und alle Blätter liegen in Tiefe t. Sei r seine Wurzel.

Beobachtung: Im schlimsten Fall besucht der Alphabeta-Algorithmus alle Knoten des Spielbaums.

Definition (Knotentyp, kritischer Knoten)

a) Die Abbildung *Typ* : V → {0,1,2,3} ist definiert durch Typ(r) = 1 und



b) ein Knoten v heißt *kritisch*, wenn Typ(v) ≠ 0.

Alphabeta-Algorithmus, Aufwand

TECHNISCHE UNIVERSITÄT DARMSTADT

Satz: Es sei A ein beliebiger Algorithmus, der zu einem Spielbaum G = (V,E,f) mit Wurzel r dem Minimaxwert F(r) berechnet.

Dann kann man G durch Permutationen der Nachfolger der inneren Knoten so zu G‘ verändern, dass jeder Knoten, den der Alphabeta-Algorithmus auf G‘ besucht auch von A auf G besucht wird.

Der Alphabeta- Algorithmus besucht bei richtiger Sortierung gerade die kritischen Knoten. Und jeder Algorithmus muss diese Knoten untersuchen.

Alphabeta-Algorithmus, Aufwand

TECHNISCHE UNIVERSITÄT DARMSTADT

Beweisskizze:

Für Knoten vom Typ 1,2,3 gelten folgende Aussagen, welche durch eine gemeinsame Induktion über die Tiefe von G bewiesen werden können:

A1: Sei v ein Knoten mit Typ(v) = 1. Dann gilt:

- A berechnet für v den Minimaxwert F(v).
- ist v ein Blatt, so wird von A f(v) berechnet,
- sonst:
  - muss A einen Nachfolger v_i von v gefunden haben, dessen untere Schranke den Wert F(v) besitzt. Ausserdem müssen für alle Nachfolger obere Schranken ≤ F(v) nachgewiesen worden sein. Insbesondere muss der Minimaxwert von v_i bestimmt worden sein. Permutiere die Nachfolger von v in G‘ nun so, dass v_i in G‘ der erste Nachfolger ist. In G weisen wir dem Knoten v_i den Typ 1 zu, allem anderen Nachfolgern von v den Typ 2.
- v wird mit negamax(v,-∞,∞) untersucht
- Typ(v_i) = 1 in G‘ Typ(v_j) = 2 für j ≠ i in G‘ und v_i wird in G‘ mit negamax(v_i,-∞,∞) untersucht, und alle anderen Nachfolger von v mit negamax(v_j,-∞,-F(v_i)).
  An den v_j kommt es zu Cutoffs, weil der erste Nachfolger von v in G‘ den größten Wert besitzt.

Alphabeta-Algorithmus, Aufwand

TECHNISCHE UNIVERSITÄT DARMSTADT

A2: Sei v ein Knoten mit Typ(v) = 2. Dann gilt:

- A berechnet für v zumindest eine untere Schranke $\beta > -\infty$.
- ist v ein Blatt, so wird von A f(v) berechnet,
- sonst:
  - muss A einen Nachfolger v_i von v gefunden haben, dessen untere Schranke $\beta > -\infty$ ist. Wir weisen v_i den Typ 3 zu, allen anderen Nachfolgern von v den Typ 0. Permutiere die Nachfolger von v in G‘ nun so, dass v_i in G‘ der erste Nachfolger ist.
- v wird mit negamax(v,$-\infty$,β) untersucht
- Typ(v_i) = 3 in G‘ Typ(v_j) = 0 für $j \neq i$ in G‘ und v_i wird in G‘ mit negamax(v_i,$-\beta$,∞) untersucht. negamax(v_i,$-\beta$,∞) liefert einen Wert $x \leq -\beta$ und es erfolgt ein Cutoff an v.

A3: Sei v ein Knoten mit Typ(v) = 3. Dann gilt:

- A berechnet für v eine obere Schranke für F(v). Er muss also alle Nachfolger betrachten. Alle Nachfolger bekommen den Typ 2.
- ist v ein Blatt, so wird von A f(v) berechnet,
- sonst: v wird in G‘ mit negamax(v,α,∞) untersucht, und Typ(v_j) = 2 für alle Nachfolger v_j von v. Die Nachfolger werden gesucht mit dem Aufruf negamax(v_j,$-\infty$,α).

Alphabeta-Algorithmus, Aufwand

TECHNISCHE UNIVERSITÄT DARMSTADT

Folgerung: Es sei G = (V,E,f) ein b/t-uniformer Spielbaum mit Wurzel r. Jeder Algorithmus zur Berechnung von F(r) besucht mindestens $b^{\lfloor t/2 \rfloor} + b^{\lceil t/2 \rceil} - 1$ Blätter von G.

Beweis:
Es sei $k_i(j)$ die Anzahl der Knoten v in Ebene j von G mit Typ(v) = i. Dann ist $k_1(0) = 1$, $k_2(0) = 0$ und $k_3(0) = 0$. Weiter ist

$k_1(t) = k_1(t-1)$
$k_2(t) = (b-1) \cdot k_1(t-1) + b \cdot k_3(t-1)$
$k_3(t) = k_2(t-1)$

Lösung des Systems ergibt:

$k_1(t) = 1$
$k_2(t) = b^{\lceil t/2 \rceil} - 1$
$k_3(t) = b^{\lfloor t/2 \rfloor} - 1$

TECHNISCHE UNIVERSITÄT DARMSTADT

Computerschach

Die letzte Runde im Mensch-Maschine Schachwettkampf ist beendet

Anmerkung: Dieses Kapitel enthält Aussagen, die man zu Recht als „nicht-wissenschaftlich“ bezeichnen sollte. Es handelt sich dabei um reine Spekulation, und sie wurden deshalb als solche gekennzeichnet. Dies gilt insbesondere für die Folien 134 und 135.

Computerschach hat eine atemberaubende Geschichte

TECHNISCHE UNIVERSITÄT DARMSTADT

1940-1970: Versuche, menschliches Schachspiel nachzuahmen, versanden

1. 1970s: Chess 4.5 ist das erste 'starke' Programm. Es legt Wert auf Sucheffizienz. Erster Computersieg eines Menschenturniers, Minnesota Open 1977.
2. 1983: Belle wurde 'National Master', 2100 ELO.
3. 1988: Hitec erringt einen ersten Sieg gegen einen Großmeister
4. 1988: Deep Thought auf Großmeisterniveau
5. 1992: Die ChessMachine gewinnt die offene Computerschachweltmeisterschaft, ein konventionelles PC-Programm von Ed Schröder.
6. 1997: Deep Blue schlägt Kasparov in einem 6-Spiele Wettkampf.
7. Von der Zeit an, dominieren PC-Programme die Welt, mit einer jährliche Spielstärke-steigerung von ca. 30 ELO. Im Internet gibt es virtuelle Räume, in denen Turniere abgehalten werde, in denen man jederzeit gegen Großmeister oder starke Maschinen spielen kann.

ELO: statistisches Maß; 100 Pkte Differenz entsprechen einer 64% Gewinnchance

Anfänger 1000ELO
Internationaler Meister > 2400
Großmeister > 2500
Menschlicher Weltmeister > 2800

Das Ziel

Nur ein BIG POINT fehlte noch:
stärker spielen als die besten Menschen.

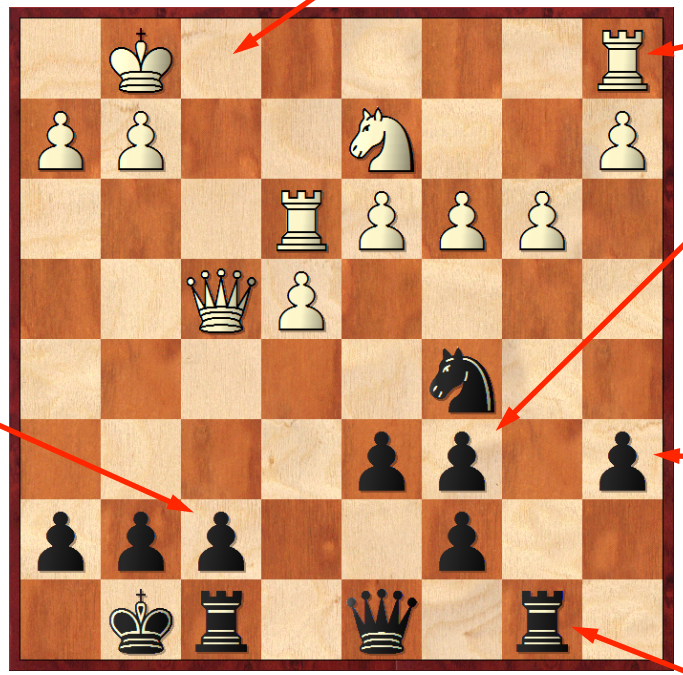
4 Programme lieferten sich bis 2005 ein Rennen:

- **Shredder** von Stefan Meyer-Kahlen,
- **Fritz** von Frans Morsch,
- **Junior** von Amir Ban und Shay Bushinsky
- **Brutus/Hydra** von Chrilly Donninger, Ulf Lorenz (2003-2006), Christopher Lutz, Nasir Ali
- Rybka, Fruit seit 2005

Die oberen vier Programme holten z.B. auf der Computerschach-Weltmeisterschaft in Graz, 2003, mehr als 95% der Punkte gegen das restliche Feld.

Was macht ein Schachprogramm eigentlich?

(I) Stellungsbewertung



London: Die Kontrahenten

TECHNISCHE UNIVERSITÄT DARMSTADT

Adams:

geboren 17.11.1971, lebt in London.

- Wurde 1989 Großmeister, mit dem Gewinn der Britischen Meisterschaften, im Alter von 17 Jahren.
- 1997 britischer Meister.
- 1990 und 2005 zum Spieler des Jahres gekürt und
- gewann zwischen 1993 und 2002 rekordverdächtig viele Turniere.

Zu der Zeit Nr. 1 in England, Nr. 7 der Welt, mit einer Spielstärke von 2741 Elo.

London: Die Kontrahenten

TECHNISCHE UNIVERSITÄT DARMSTADT

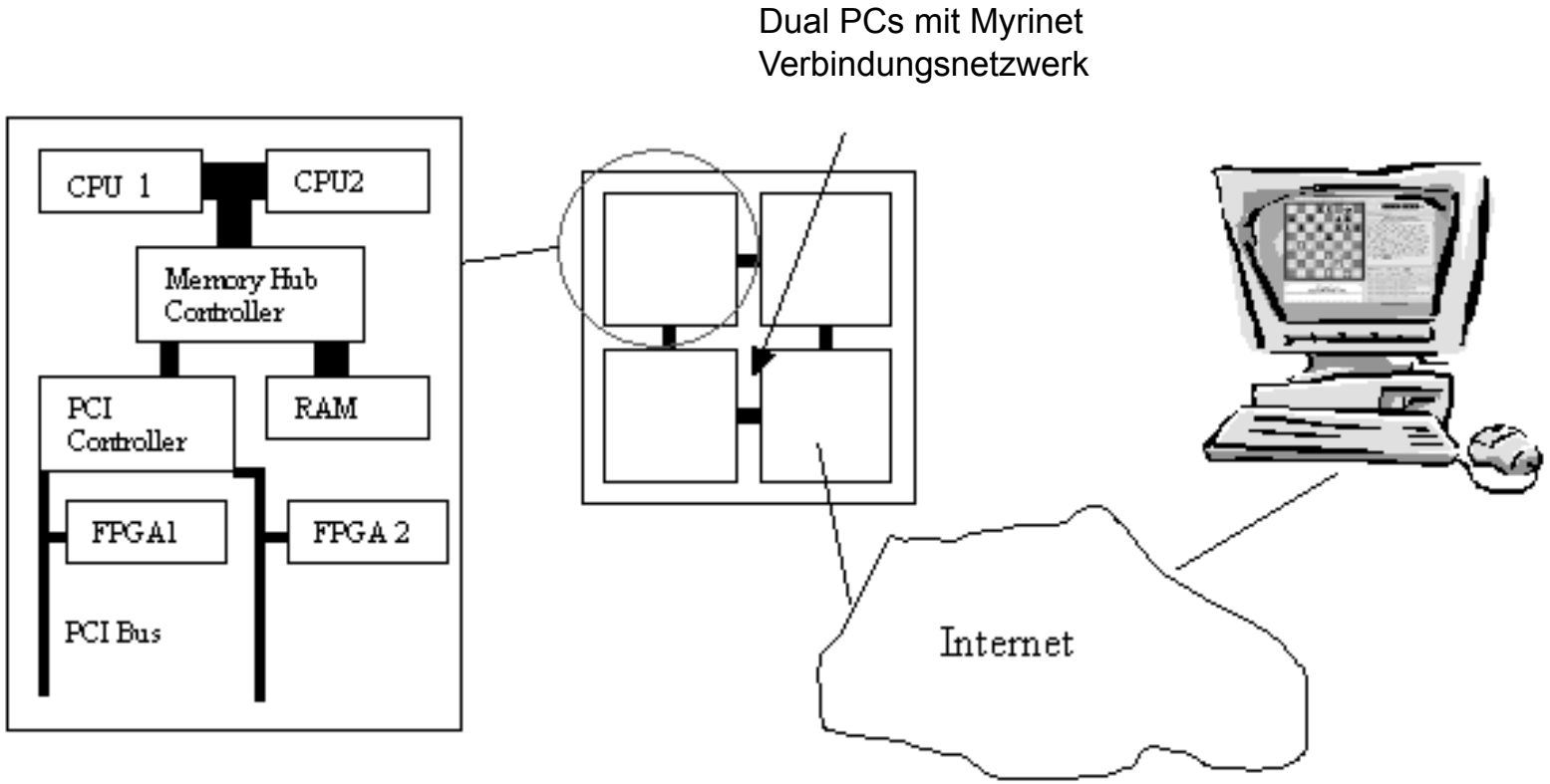
Hydra:

- Die einzige 'Schacheinheit', die jemals (bis 2009) deutlich über 3000 Elo spielt.
- 150 Millionen Schachstellungen pro Sekunde
- typische Rechentiefe nach der Eröffnung: 19 bis 20 Halbzüge
- Cluster mit 32 Pentium Prozessoren, jeweils mit einem XilinX Virtex Pro II 7000 als Schach-Koprozessor

(finanziert und durchgeführt von PAL Computer Systems, VAE, Abu Dhabi)

- Gewinner des 13. IPCCC 2004 mit 6.5 aus 7
- Gewinner gegen Shredder in Abu Dhabi mit 5.5 : 2.5
- Gewinner gegen GM Vladimirov (2630 Elo) mit 3.5 : 0.5
- Mensch-Maschine Mannschaftsweltmeister mit 3.5 : 0.5 gegen Schnitt von 2690 ELO
- Gewinner des 14. IPCCC 2005 mit 8 aus 9

Hydras globale Architektur

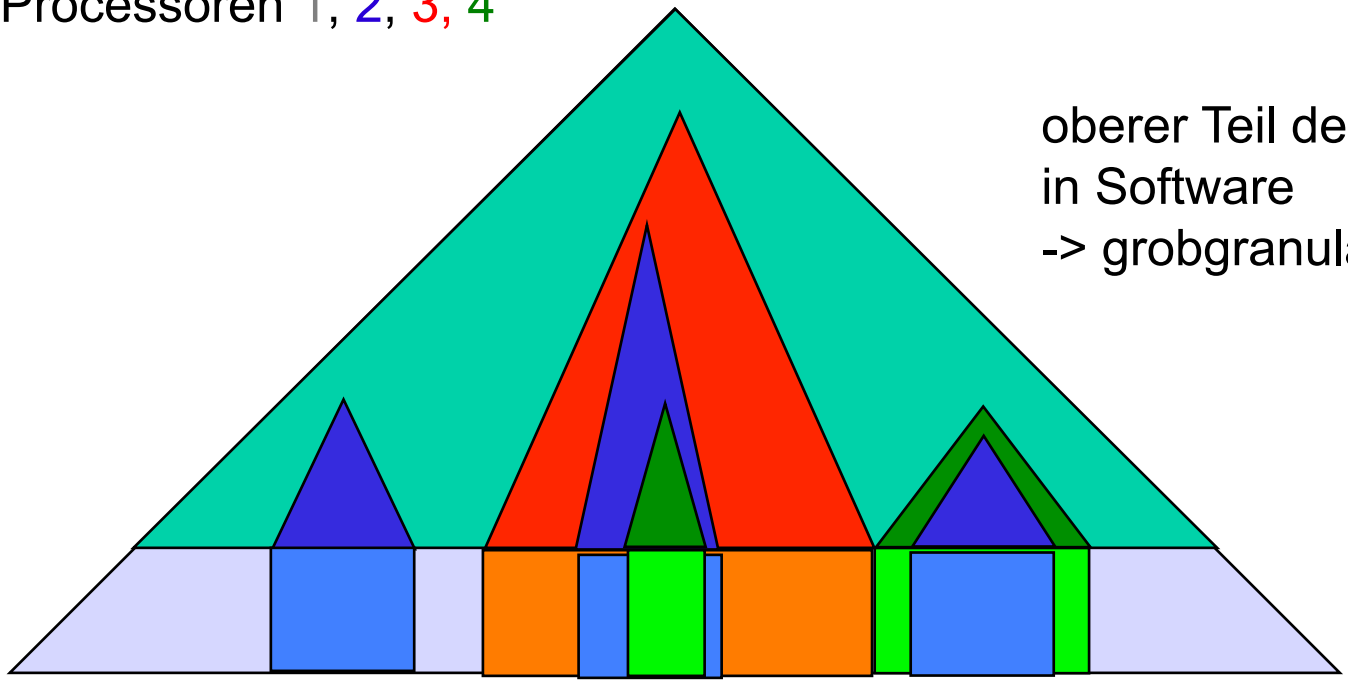


FPGA: Virtex II: VP70 -> ca. 5,4 Millionen Suchknoten pro Sekunde pro Prozessor
bis zu 32 CPUs: 3.0 GHz -> Speedup 17 und somit 100 Millionen Suchknoten pro Sekunde

Zerlegung des Suchbaums

TECHNISCHE UNIVERSITÄT DARMSTADT

Processoren 1, 2, 3, 4



oberer Teil des Baums
in Software
-> grobgranulare Parallelität

die letzten 4 Ebenen in FPGA -> feingranulare Parallelität,
FPGA Karte wird ca. 100000 mal pro Sekunde angestoßen

Paralleler Suchalgorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Zerlegung des Suchbaums
- ‘Work Stealing‘: Prozessoren ohne Arbeit senden randomisiert Anfragen
  in den Cluster;
  ein spezieller Prozessor beginnt die Berechnungen wie im sequentiellen Fall
- ein Prozessor, der Arbeit hat und eine Arbeitsanfrage einfängt, gibt ein Teilproblem
  an den Sender der Anfrage ab,
  der Anfragende wird 'worker‘, der andere ‘master‘
- master/worker Beziehungen sind dynamisch, oft geknüpft und wieder gelöst

- Nachrichten: REQUEST, WORK, NO-WORK, WINDOW_UPDATE,
  CUTOFF, RESULT

- Problemauswahl für Abgaben: nah an der Wurzel, YBWC

Speedups

	Zeit(s)	SPE	SO %
1	24213	1	0
2	12139	1.99	0
4	6888	3.5	3.6
8	3488	6.5	-1
16	2011	12	9.3
32	1424 *	17	80 *

* geschätzt, da andere Versionen von Hydra zugrunde liegen

FPGA (Field Programmable Gate Array)

TECHNISCHE UNIVERSITÄT DARMSTADT

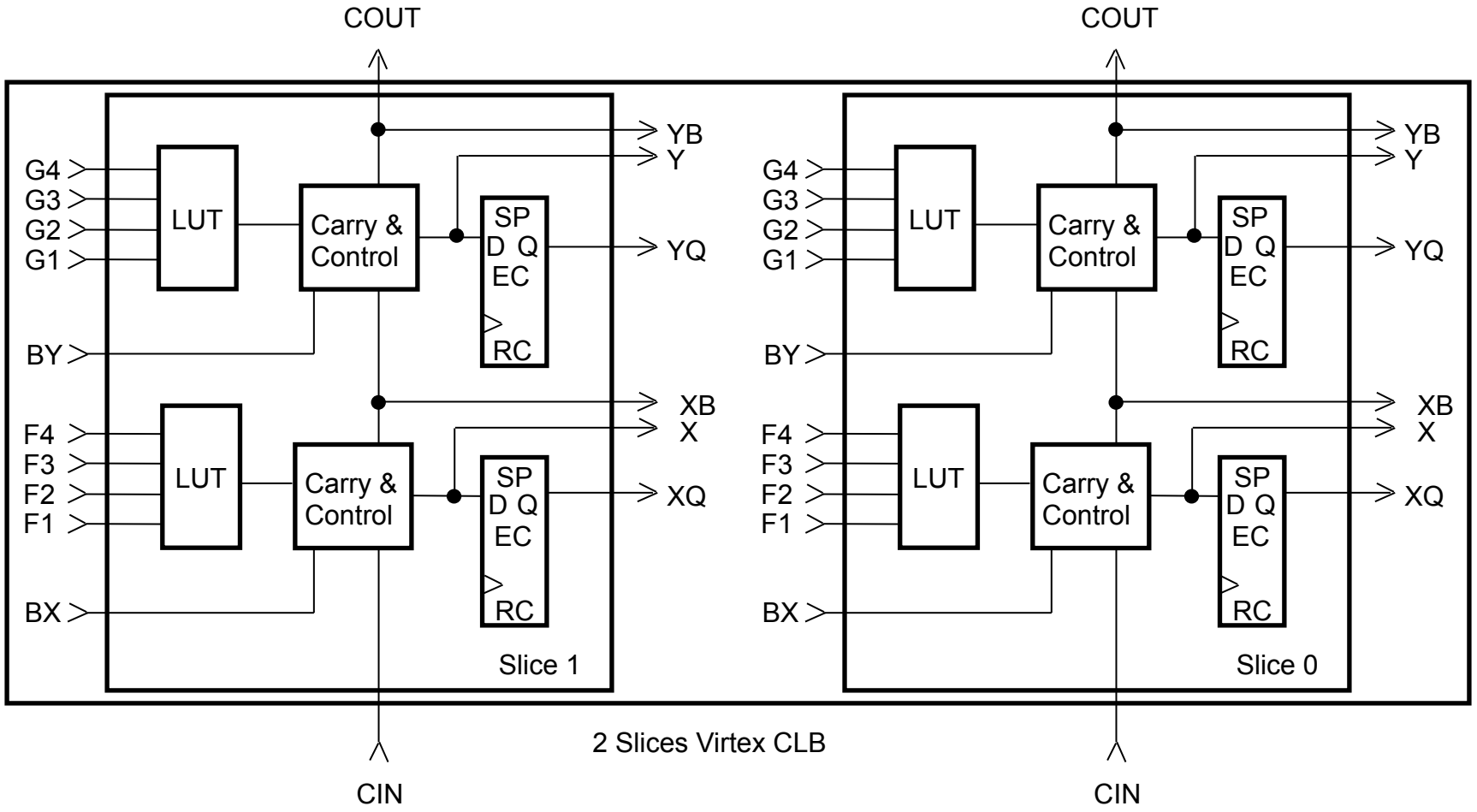
- **ist rekonfigurierbare Logic für rapid prototyping und für die Implementierung digitaler Systeme**
- **ist eine Art in RAM simulierter Hardware, die fein-granulare Parallelität erhaltend**
- **ist eine Möglichkeit Logik zu realisieren**

Hauptvorteile (für Hydra):

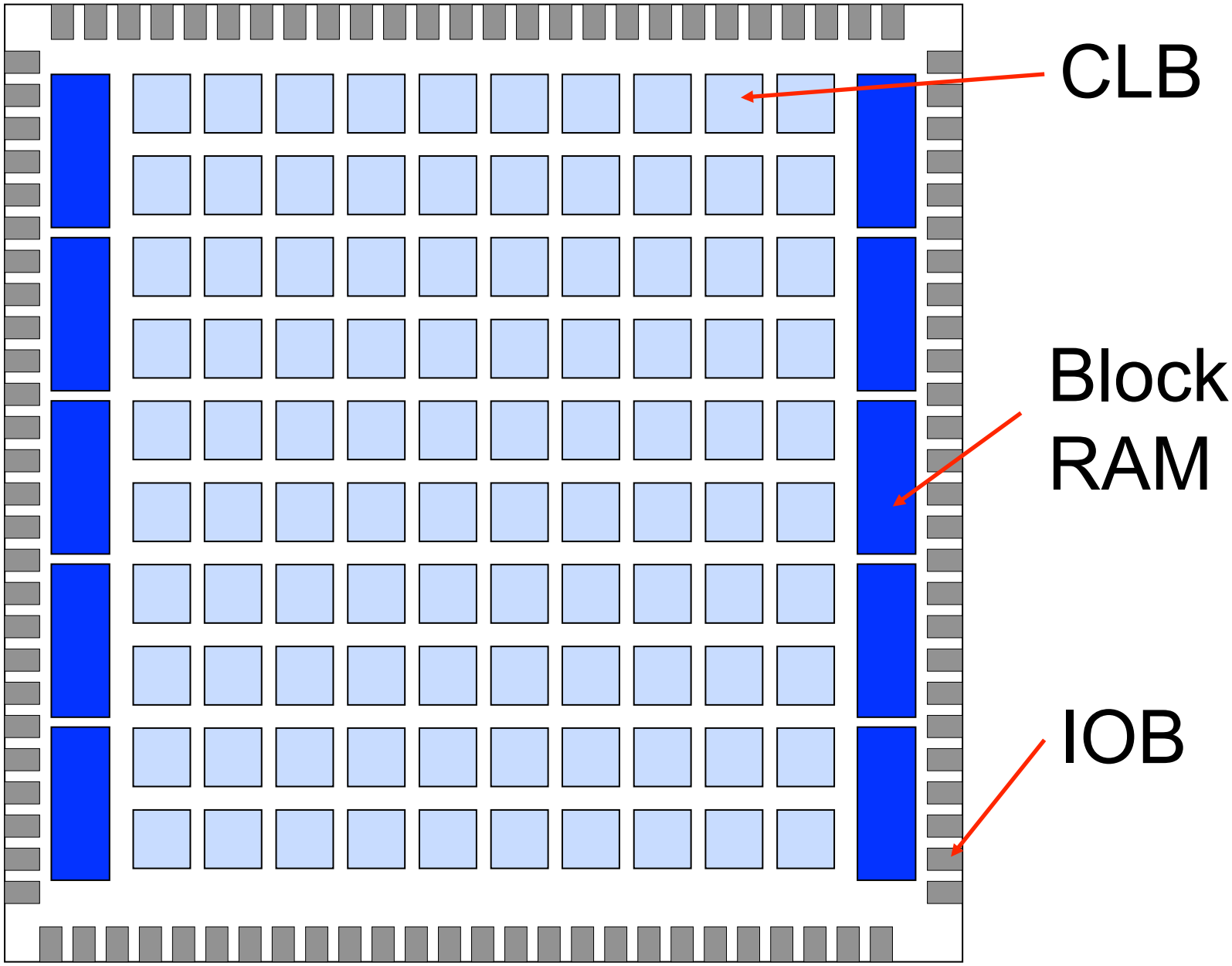
1. Das Hinzufügen von mehr Schachwissen erfordert zusätzlichen Platz, kostet aber nahezu keine zusätzliche Berechnungszeit

2. FPGA Code kann ge-debugged und schnell wie in Software geändert werden, ohne langwierige ASIC Entwicklungszeiten

3. feingranulare Parallelität kann genutzt werden -> genMove, doMove, eval, undoMove in 9 Taktzyklen, bei einer clock-rate von 50MHz

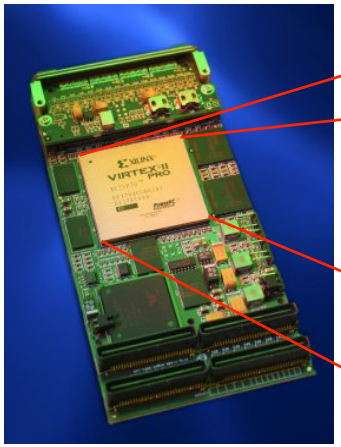


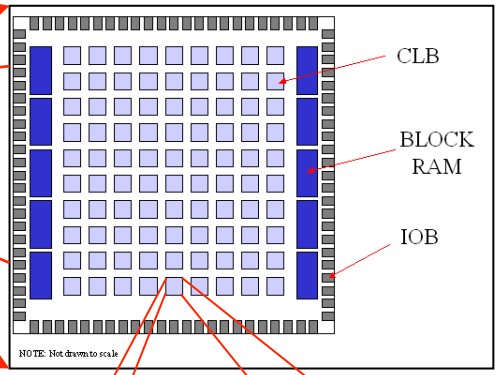
CLB: Configurable Logic Block LUT: Look-Up-Table

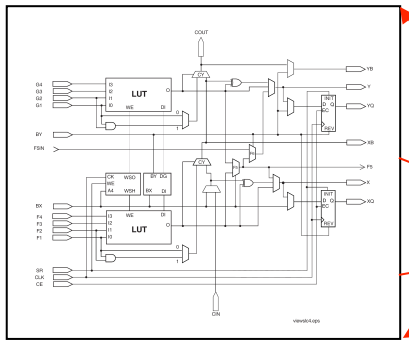

CLB
Block
RAM
IOB

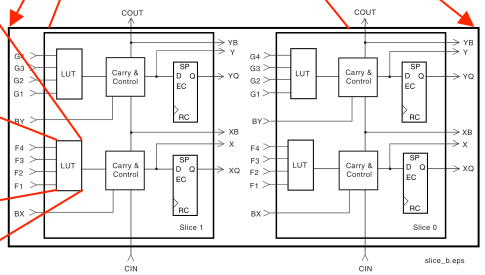

XILINX
VIRTEX-II
PRO

HPRCC
High Performance Reconfigurable Cluster-Computing









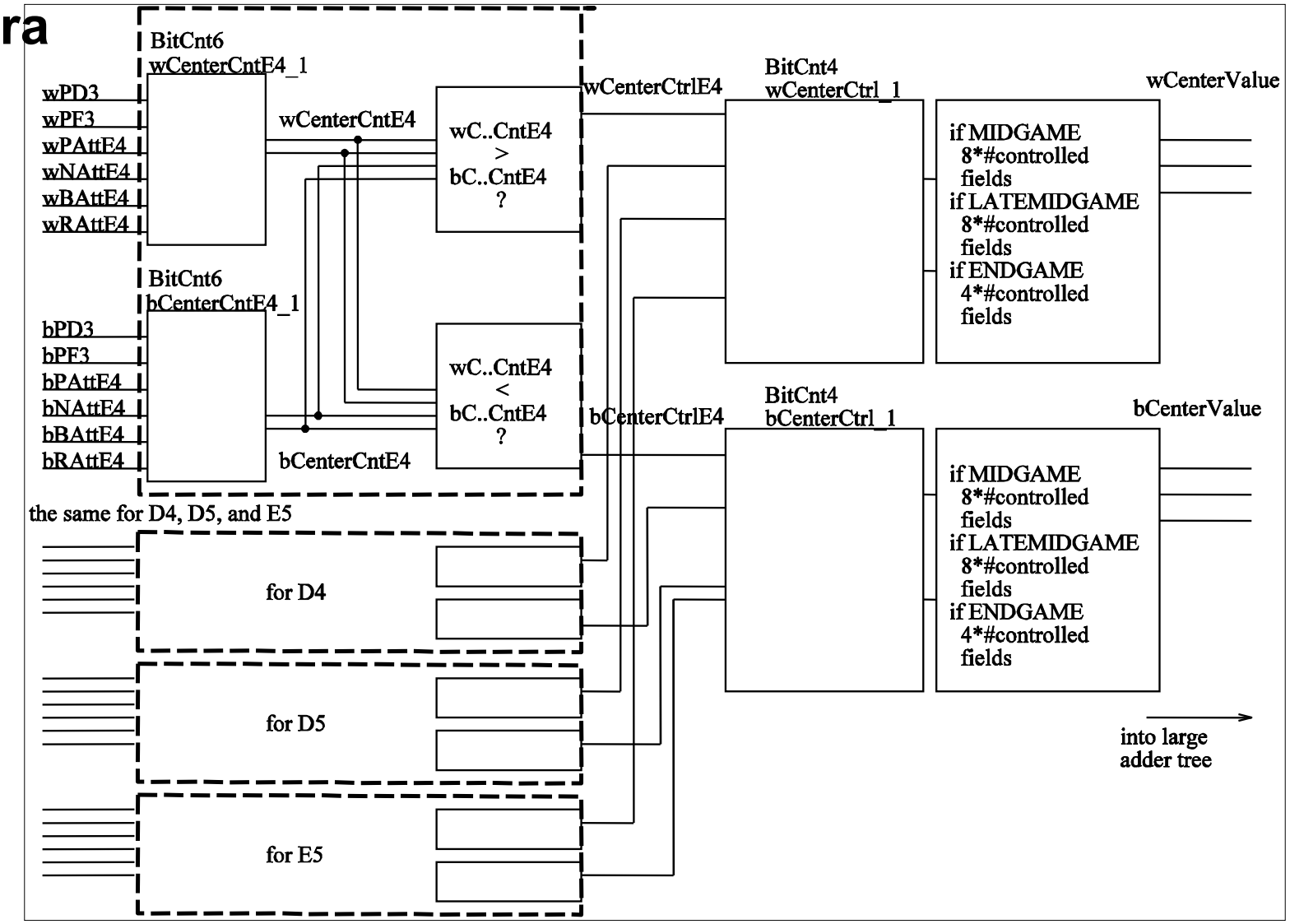
Detailed View of Virtex Slice

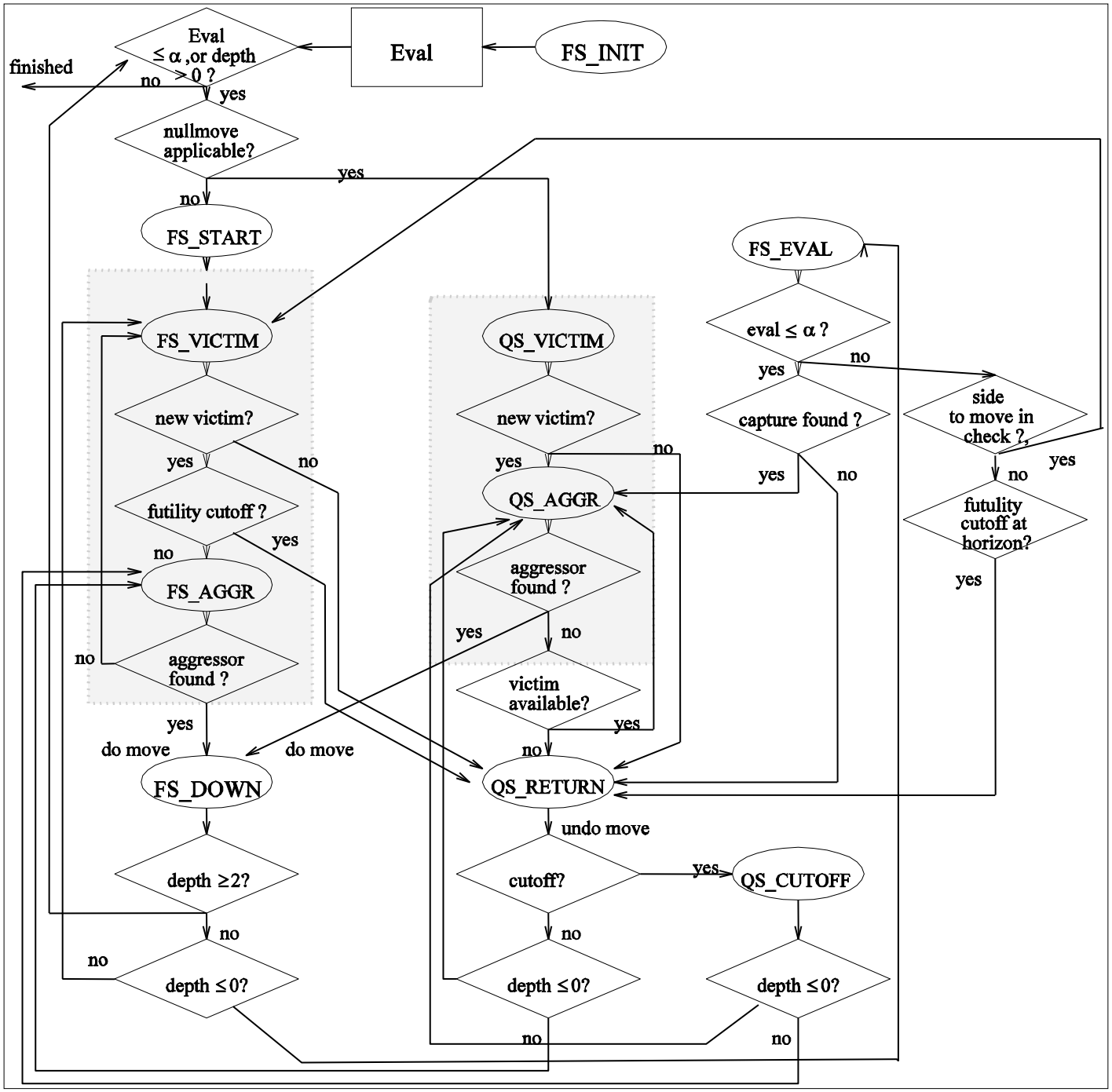
2 Slices Virtex CLB

FPGA, Verilog

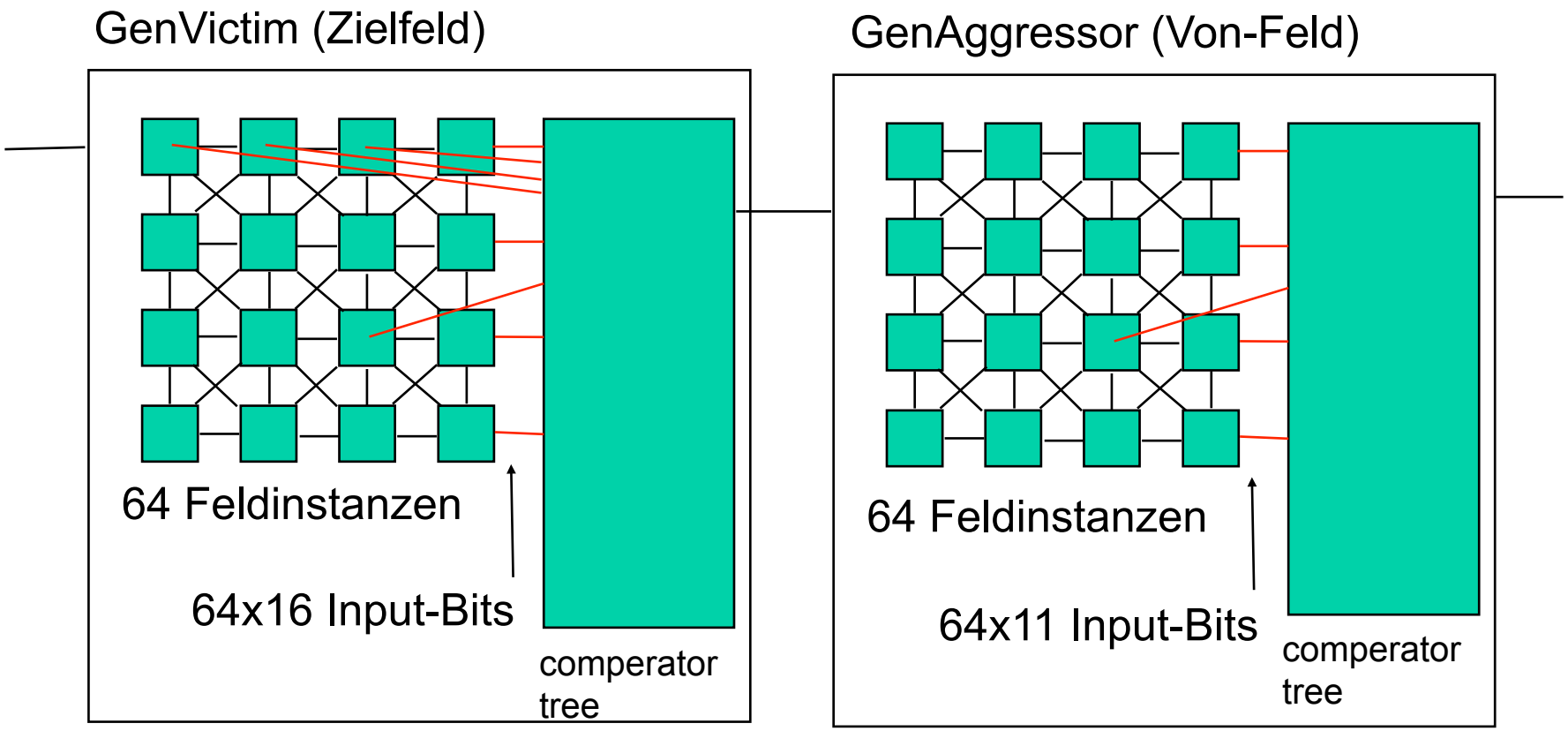
```
module Search();
  ControlLogic;
          // Enables/disables the modules and converts data to the correct format.
          // E.g. the module Board does not know whether DoMove or UndoMove is
          // done. The input is always piece-from, piece-to. The
          // ControlLogic converts the move
          // information in the correct format for the Board module.
  AlphaBetaFSM;   // Alphabeta search with Finite State Machine.
          // The FSM sets a few signals like "DoMove", "UndoMove",
          // "Next-Victim". The next state is determined from the
          // output of the modules and the current state. The signals are used by the
          // ControlLogic to control the operation of the modules.
          // The FSM has 54-States, but some states are simply waitstates. It changes no
          // signals. Only a transition to the next
          // state can be done.
  CastleStatus();      // Determine castle status                                  0.5 cycles
  GenVictim();         // Generates next ToSquare.                                 2.0 cycles
  GenAggressor();    // Generates next FromSquare for a given ToSquare.  2.0 cycles
  TestCheck();        // Tests if the king is in check.                           2.0 cycles
  Board();             // Board respresenation and board update                    1.0 cycles
  SearchStack();     // "Local variables" for the search control.                0.5 cycles.
  Evaluate();          // Evaluation.                                              3.0 cycles.
endmodule

module Evaluate();
  HandleSpecialCases; // E.g: Is there insufficient mate material?
  WeightedSum;          // Different weights for different game phases.
  GamePhase();          // Determines the game phase.
  PieceValues();        // Sums up the material value.
  FirstOrderValues();   // piece square tables
  PawnStructure();      // pawn structure and passed pawns
  KingCover();          // pawn-cover around the king
  Dynamic();            // Main part. This evaluation bases on attack and defense
                        // relations. E.g. attack on opponents king, mobility
endmodule
```

Hydra


BitCnt6
wCenterCntE4_1
wPD3
wPF3
wPAttE4
wNAttE4
wBAttE4
wRAttE4
wCenterCntE4
wC..CntE4
>
bC..CntE4
?
wCenterCtrlE4
BitCnt4
wCenterCtrl_1
wCenterValue
if MIDGAME
8*#controlled
fields
if LATEMIDGAME
8*#controlled
fields
if ENDGAME
4*#controlled
fields
BitCnt6
bCenterCntE4_1
bPD3
bPF3
bPAttE4
bNAttE4
bBAttE4
bRAttE4
wC..CntE4
<
bC..CntE4
?
bCenterCntE4
bCenterCtrlE4
BitCnt4
bCenterCtrl_1
bCenterValue
if MIDGAME
8*#controlled
fields
if LATEMIDGAME
8*#controlled
fields
if ENDGAME
4*#controlled
fields
the same for D4, D5, and E5
for D4
for D5
for E5
into large
adder tree


Eval
FS_INIT
Eval ≤ α ,or depth > 0 ?
finished
no
yes
nullmove applicable?
yes
no
FS_START
FS_VICTIM
new victim?
yes
no
futility cutoff ?
yes
no
FS_AGGR
aggressor found ?
no
yes
do move
FS_DOWN
depth ≥2?
no
no
depth ≤0?
QS_VICTIM
new victim?
yes
no
QS_AGGR
aggressor found ?
yes
no
victim available?
yes
no
QS_RETURN
undo move
cutoff?
yes
QS_CUTOFF
no
depth ≤0?
no
depth ≤0?
no
FS_EVAL
eval ≤ α ?
yes
no
capture found ?
yes
no
side to move in check ?
yes
no
futulity cutoff at horizon?
yes

Hydra



1. Besetzte Felder senden ein Signal in GenVictim.
2. Freie Felder leiten Signale weiter
3. Alle Felder, die ein Signal bekommen, sind potentielle Zielfelder.
4. Der Comperator-tree wählt attraktivstes Zielfeld (z.B. Schlagzug).

1. Gewinnerfeld generiert Signal von 'super piece'.
2. Freie Felder leiten das Signal weiter.
3. Von eigenen Figuren besetzte Felder sind potentielle Von-Felder.
4. Der Comperator-tree wählt das attraktivste Von-Feld

Das Entwicklerteam hinter Hydra

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr. Chrilly Donninger

GM Christopher Lutz

Dr. Ulf Lorenz

Universität Paderborn

Prof. Dr. B. Monien

Paderborn Center for Parallel Computing

Nasir Ali

PAL Computer Systems

PAL Gruppe Abu Dhabi

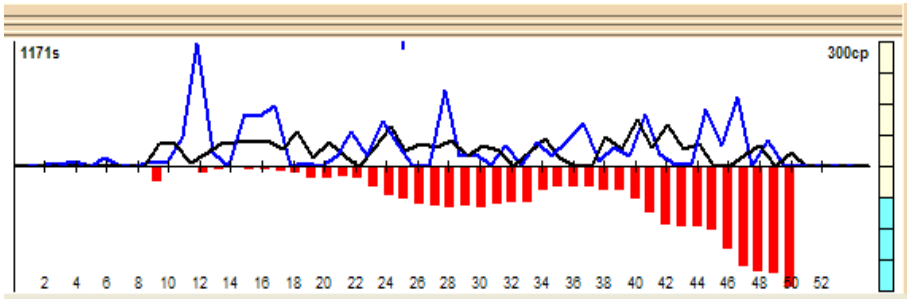
Scheich Tahnoon

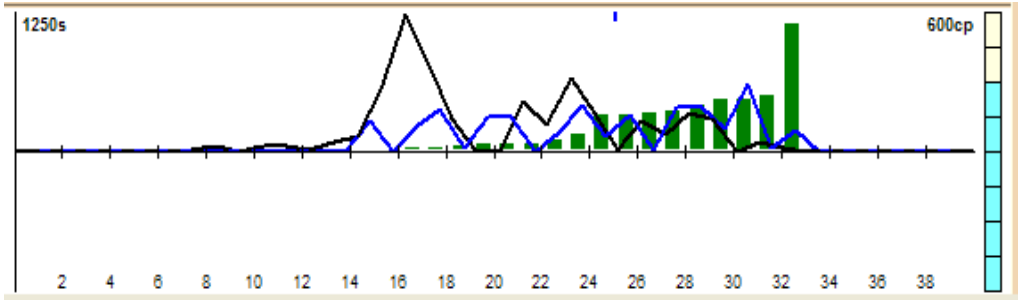
London: Hydra gegen Adams 5,5 : 0,5

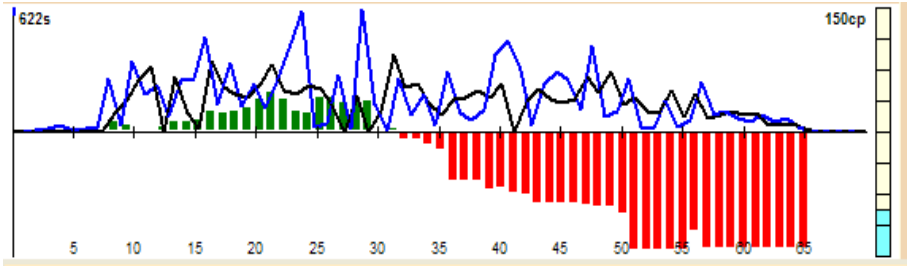
TECHNISCHE UNIVERSITÄT DARMSTADT

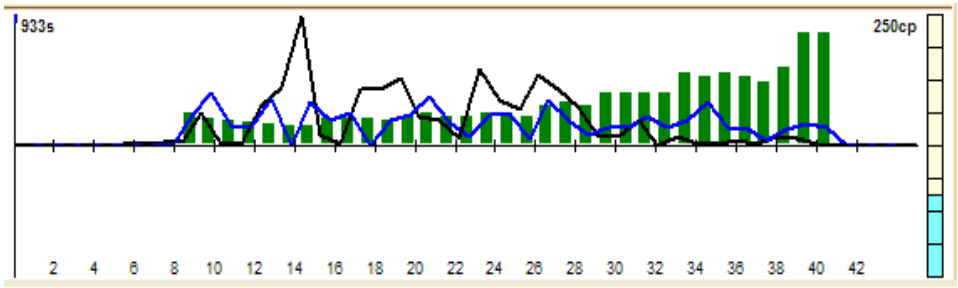
London 2005

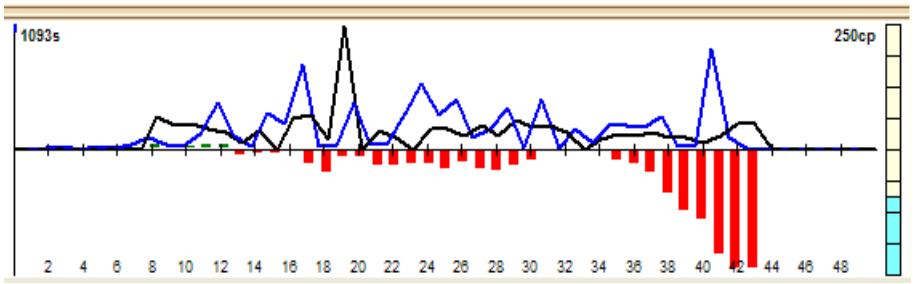
TECHNISCHE UNIVERSITÄT DARMSTADT

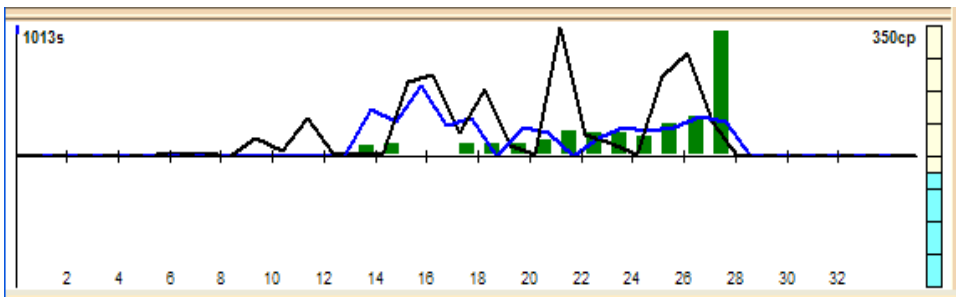












Typische Thesen aus den 80er Jahren (*) :

- Top-Großmeister spielen fast perfekt Schach
- Ein Top-Großmeister kann mit weiß immer Remis halten, wenn er nur kein Risiko eingeht und von vornherein auf Remis spielt.
- 3000 Elo ist eine natürliche Perfektionsgrenze
- Turmendspiele mit einem Wenigerbauern sind einfach Remis zu halten.

Alles Falsch!

Katastrophale Selbsteinschätzung.

(*) Diese Folie präsentiert auch aus Sicht Ihres Dozenten keine wissenschaftliche Erkenntnis. Sie dient allein der Anregung zu Diskussion!

Ein zarter Hauch von Intelligenz (*)

TECHNISCHE UNIVERSITÄT DARMSTADT

1. **Die Menschlichen Meister beanspruchen, daß sie, obwohl sie gegen ein Programm verlieren, das Spiel besser verstehen. Sie sprechen von einem 'tieferen Verständnis'.**

Frage: Tun sie das? Was könnte das sein: 'tiefer verstehen'? Gibt es überhaupt etwas zu verstehen, außer 'Jede Stellung ist entweder gewonnen, verloren oder remis'?

Antwort: Ja, es gibt da was ...

(*) Diese Folie präsentiert auch aus Sicht Ihres Dozenten keine wissenschaftliche Erkenntnis. Sie dient allein der Anregung zu Diskussion!