Deep Blue

This paper describes the Deep Blue’s (henceforth referred at DB) system design and elaborates on the specific techniques and factors at play that contributed to DB beating reigning world champion Garry Kasparov in 1997.

Previously, DB I (a single chip system) had lost to Kasparov. DB II was designed keeping the DB I failures and inadequacies in mind. The chess chip was upgraded with a brand-new evaluation heuristic. The number of chips on board were doubled and SP computer system was used to keep up with the extra processing requirements for deeper searches. Additionally, a bunch of SW tools were designed for better debug-ability.

DB is a at-scale, massively parallel system designed with over 500 processors for chess game tree searches. The SP processor had a Master-Slave design, with the master searching at TOP level and providing the leaf nodes to workers for further evaluation. The chess chips would search last few levels of the tree.

The chess chip comprises of: the move generator, the evaluation function, and
the search control -

- move generator – for searches to be effective and efficient, the best moves need to be amongst the first few in the order, move generator chip helps with that
- evaluation function – fast & slow evaluations helped with the better evaluations design than DB I. Piece placement value was an important part of fast evaluation. Slow evaluation scans for concepts like square control, pawn structure etc.
- search control – was designed to implement null-window alpha-beta search
- an external FPGA was sometimes used for fast transposition table lookup

Search comprised of two parts, the Software search and hardware search. Software search was based on several principles, some of them were extending forced pair of moves, delayed extensions etc. A key feature was the dual credit system – this was to separate the accumulated credits for player and opponent. IT is a zero-sum game for credit, if one player encashes, the other player has to give up the same amount of credit.

To further improve the behavior, large number of credit generation mechanisms were introduced - Singular, binary, trinary, Absolute singular threat, mate threat, Influence and Domain dependent. The credit assigned for various conditions is depth dependent, with positions near the root of the tree generally receiving more credit than positions far from the root.

The hardware search is fast but shallow. Some of the parameters used were - Depth of search, Depth of offset searches, Endgame rules assertions off or on, Number of “mating” checks allowed for each side in the quiescence search etc.

The parallel search algorithm was a complex one to design. Control distribution and process hierarchy would be required for better search line management, there were instances defined that called for parallelism and in cases where parallelism was not used. Post processing for the parallel track included synchronizing the results. There were three major issues with the Parallel search that need to be addressed: Load balancing, Master overload and Sharing between nodes. The performance varied based between deep forcing sequences to quieter positions. It was not easy to measure as the search results would be non-deterministic.

Evaluation function features ranged from very simple to extremely complex features. Dynamic feature values were used to scale up at run time based on value and type of pieces. Additionally, the evaluation generator makes abstractions, dictating relationships between groups of related feature values rather than setting them independently. Some automation of evaluation function analysis was conducted as well.

From a past learning perspective - *Opening book, extended book and end game databases* were created to study the thousands of moves.

Conclusion –
Given the large space and time complexity of a game like chess, having a trivial system to handle the goodness of the board and picking the right search tracks was going to be inadequate. Massively parallel computing, Large searching ability, different search strategies for different part of the game and a complex evaluation function were important components for success. Additionally, having a rich reference database for end games helped immensely.