CHAPTER 2
LITERATURE SURVEY

2.1 POWER OPTIMIZATIONS IN HLS

With the advent of portable devices the power dissipation of VLSI circuits is becoming a critical concern. Power variation also reduces the chip lifetime. Power offers a lot of scope for optimizations. Temperature also has a significant impact on power consumption, which is a critical feature in embedded systems. Making early decisions related to power will guide us to obtain optimized design.

It is shown that the Mean Time between Failure (MTBF) of a chip is multiplied by a factor of 10 for every 30 °C rise in the junction temperature. The increase of temperature decreases carrier mobility and thus switching speed of the transistors, which then increases the overall timing of the circuits. Also, temperature drastically increases the leakage power, which is becoming a major source of power consumption (Lim 2006).

If the power is estimated accurately during the design phase itself, then there will be no need for a costly redesign process to meet the power specifications. Power estimation refers to the average power dissipation of the circuit. Some of the techniques for power estimation as proposed in (Najm, 1994) are listed below:

Simulation - Using this method an average power supply waveform is obtained and it is more accurate. They are dependent on input signals and
hence strongly pattern dependent. Also, they are too slow to be used on large circuits.

**Delay models** - They are weakly pattern dependent. The user has to provide only some information on the behaviour at the circuit inputs. They are not accurate.

**Statistical techniques** - They are more accurate and the accuracy can be specified up-front.

**Probabilistic techniques** - They require specialised simulation models and are not as accurate but can be faster.

No approach is suggested best by the authors. Each approach has its own merits and demerits.

(Mike 1997) made a detailed study about the energy consumption due to software for a DSP processor. In DSP programs, an on-chip booth multiplier is a major source of energy consumption. Booth multiplier is an encoding scheme to reduce number of stages in multiplication. The target DSP processor used for their study is a Fujitsu 3.3 V, 0.5 m, 40MHz CMOS processor. The following techniques are proposed by the author.

1. **Memory bank assignment for low energy** - The target DSP processor contains two on-chip memory banks. If the operands required for an ALU operation are stored in the same bank, then two cycles are needed, one for each transfer. Storing the operands in separate banks reduces the number of cycles. Reducing the number of cycles saves energy.

2. **Instruction packing for low energy** - Two instructions could be packed into a single instruction for simultaneous execution. This
feature is called as instruction packing. However, their functionality remains the same as that of the sequence of two unpacked instructions. Packing the instructions leads to the reduction of the energy because the average current for the packed instructions is more than average current for the sequence of the two unpacked instructions.

3. Instruction scheduling for low power - The power cost of a program could be reduced by appropriate scheduling of instructions. List scheduling algorithm could be used in which a ready list of instructions whose operands are available is maintained. Then the instruction with lowest overhead cost is selected to be scheduled at the current cycle. The scheduled instruction is then deleted from the ready list. Incorporating this list scheduling approach will reduce power.

4. A novel cluster based register optimization technique called Power Islands Synthesis (PIS), is introduced in (Mansouri 2007). This technique reduces the number of registers and the power consumption of the registers. Each island is characterised as the cluster of logic whose power can be controlled independently and can be powered down when all the logic it contains is idling.

As a result of powering down an island the following advantages are gained.

1. Power consumption due to leakage in inactive components is eliminated, and

2. Spurious switching activity that results from the broadcast to idle components is silenced.
(Kim 2004) devised a technique that can be used effectively to the synthesis of arithmetic circuits aiming power minimization while satisfying the timing constraint of the circuit. The leakage power minimization problem could be solved in three phases.

1. Initial phase - Leakage power minimization.
2. Coarse-grained refinement phase - The leakage power obtained in the initial phase is traded with the circuit timing to meet the timing constraint. The trading process is iterative. During each iteration, the appropriate cell is selected and replaced.
3. Fine-grained refinement phase - Further refinement is done for the replaced cells in the coarse-grained phase under the condition that the critical timing of the circuit should not be satisfied.

Arrays in the behavioural specifications are usually mapped to off-chip memories during synthesis. Those arrays could be analysed for patterns and organized in the memory to minimize the transitions on the memory address buses. As a result, the system power could be reduced. This idea is proposed by (Dutt 1999).

Power consumption during memory accesses within a memory can be roughly classified into two categories, data-related and address-related. Data-related power is the component of memory power contributed by the process of transferring the data between the data bus and memory location where the data is stored, and the component dissipated in the refresh circuitry of the memory. Address-related power refers to the component contributed by the address buffers and decoders. Address related power minimization alone is focused by (Dutt 1999). The behavioural specification to be synthesized is taken as input. The output is the assignment of arrays given in the behavioural specification to physical address in memory.
The expression for accessing the memory element in the array is outputted as a function of its behavioural array index and the array dimensions that corresponds to the best memory mapping strategy (Single or Multiple memory mapping) for the array. The transfer count to and from the memories influences the power budget. By reorganizing the way in which the data are stored, by caching the most frequently accessed data localizing the data in the foreground memory (on-chip) would reduce the background memory (off-chip) accesses. Since accessing on-chip memories consumes no major I/O power and since these foreground memories will, in general, be much smaller than external memories, the total power consumption will drop significantly (Catthoor, 1994).

Traditionally, most automated power optimization tools have focussed at gate-level and physical-level optimizations. However, major power reductions are only possible by addressing power at the RTL and system levels (Anmol 2009), (Pedram 2005) devised a tutorial that reviews the number of RT-level design automation techniques that focus on low-power design. Digital circuits usually contain portions that are not performing useful computations at each clock cycle. Power reductions can then be achieved by shutting down the circuitry when it is idle. Some of the ideas presented in are listed below:

1. Precomputation logic - Some parts of the logic are duplicated, so that the values are precomputed one clock cycle earlier, before they are required. Those values could be used to reduce the total amount of switching in the circuit during the next clock cycle.
2. Clock gating - Gated clocks, selectively stop the clock and thus force the original circuit to make no transition. This is done whenever the computation that it to be carried out at the next clock cycle is redundant. This idea holds good in circuits where the number of clock cycles in which the design is idle in some wait states is usually large.
3. Computational kernels - A Computational kernel is a highly simplified logic block that imitates the steady state behaviour of the original specification. This block is smaller, faster and less power consuming than the circuit from which it is extracted, and can replace the original network for a large fraction of the operation time.

4. State machine decomposition - The State Transition Graph (STG) of a Finite State Machine (FSM) is decomposed into two STGs that together produce the equivalent behaviour as the original machine. Decomposition are done based on the procedure given in (Monteiro 1998). Power saving is achieved as a result, because, except for the transitions between the two sub-FSMs, only one of the sub FSMs needs to be clocked.

5. Guarded evaluation - Unlike precomputation and gated clocks, guarded evaluation does not require synthesis of any additional logic to implement the shutdown mechanism. Instead, the existing signals in the original circuit are used. The approach is based on placing some guard logic, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power-managed. When the block must execute some useful computation in a clock cycle, the enable signal makes the latches transparent. Otherwise, the latches retain their previous states, thus blocking any transition within the logic block.

The reviewed RT-level design automation techniques emphasize on the run time power management techniques and potential power gain will be obtained in the final circuit by adopting these techniques.

A compiler optimization technique called Dead Code Elimination checks whether the definitions are useful to the algorithm execution or the final output of the hardware. If not, it can be removed safely. This helps in decreasing
the energy consumption that will arise as a result of computing the definitions (Xiaoyong Tang 2003).

Binary networks are often target of global logic structure synthesis, serving as an intermediate description between input HDL and the final network in a specific target technology. They are technology-independent description of combinatorial circuits. Generally binary networks are represented using simple network structures like AND Inverter Graphs (AIG), NAND Graphs, OR-Inverter Graphs (OIG), AND-XOR-Inverter graphs. A novel efficient synthesis method for combinatorial logic circuits representation called NAND Inverter Graph (NIG) is proposed in (Edwards, 2007). Power efficiency of 33.76% and delay improvement of 18.57% could be achieved by optimal NIGs over minimal AIGs and OIGs.

Technology driven High-Level Synthesis (THLS) is a customized HLS tool proposed by Joseph et al. (2007) that makes right inference of the hardware during the early stages of the compilation and produces an unoptimized netlist in terms of silicon area, power and speed. It makes the present HLS knowledgeable of the target Field Programmable Gate Array. It makes right inference of hardware by attaching target specific attributes to the parse tree in it, which guides to generate optimized hardware in terms of speed, power and silicon area. It uses Technology Specific Library instead of Library of Parameterized Modules.

Power reduction in CMOS VLSI (Slawomir, 2003) could be achieved by,

1. Optimal scheduling
2. Proper choice of circuit parameters
3. Minimizing switched capacitance
4. Choosing proper technology.
The total power consumption of a CMOS circuit has 2 components. Dynamic power and static power. Dynamic power varies with operating frequency and the static power is independent of frequency. Capacitive current and the short circuit current are the sources of dynamic power consumption.

The off chip memory bandwidth affects the power consumption of the chip. Several loop transformations in the imperfectly nested loops provides an efficient approach to reduce the bottleneck (Cong 2012).

Application of the above techniques in the HLS design methodology would result in an optimized circuit design.

2.2 MEMORY OPTIMIZATIONS IN HLS

Memory offers a lot of scope for optimizations. There are several programming technologies available including the Static Random Access Memory (SRAM) that can be re-programmed any number of times. Drawback of SRAM is its volatility when the power is turned off. RAM with dual ports is also available. The read and write operations of memory uses one clock edge. With dual port RAM, both the ports can access any memory location at any time. To avoid conflicts in accessing the memory location, the user must observe certain restrictions. Memory offers a lot of scope for optimization.

Some arrays that are used inside the loops are too large to be kept in the on-chip memory (foreground memory), so they are kept in the off-chip memory (background memory). The access to the background memory is costly and must be reduced. The loops could be reorganized and the number of read/write accesses could be reduced. However, the intermediate data can be kept in foreground and they can be discarded as soon as they have been manipulated (Catthoor 1994). Consider the example given below:
For (i=1; i ≤ n; i++)
{
    b[i] = f(a[i]);
}
For (i=1; i ≤ n; i++)
{ c[i] = g(b[i]); }
{  
D = g(c[j],D);  
}  
for(i=1;i ≤n; i + +)  
{  
b[i] = f(a[i]);  
}  

**Optimized Code**

In the example given above from (Catthoor 1994), D is a scalar variable. Scalar variable does not require background memory. Assuming array c is no longer needed, the locations initially used by the c array can be used to store b array. This saves n memory locations.

Most frequently accessed data could be cached in the foreground memory, thereby reducing the background memory access. The total power consumption will drop significantly, as the power required to access the foreground memory is less than the power required to access the background memory.

Memory addressing is the primary source of overhead in the overall implementation cost (Miguel 1998). The memory address involves complex arithmetic calculations to access memory and they can be reduced by performing global scope algebraic transformations to exploit common sub expression, loop invariant code motion opportunities in the address expressions before the architecture mapping. This solution is proposed by Miranda et al.

Modern Dynamic Random Access Memories (DRAMs) commonly utilize the following six memory access modes (Nicolau 1997):

1. Read mode - single word read
2. Write mode - single word write
3. Read-Modify-Write mode (R-M-W) - single word update, involving read from an address, followed by write to the same address. This mode is faster than two separate read and write accesses.
4. Page mode read - successive reads to multiple words in the same page
5. Page mode write - successive writes to multiple words in the same page
6. Page mode read modify write - successive R-M-W updates to multiple words in the same page.

Present day HLS tools use the read mode and write mode alone, treating different memory accesses as independent multi cycle operations. This leads to the inefficient utilization of the memory features. In addition to read and write mode there is R-M-W modes. In this mode, there will be one row decode and one column decode stage and this mode is faster than two separate read and write accesses (Nicolau 1998).

The memory accesses can be reordered to exploit this access mode. For example, in the code a[i]=b[i]+a[i], there is read b[i]-read a[i]-write a[i] access which utilizes R-M-W mode. If the access pattern is, read a[i]-read b[i]-write a[i] then the accesses can be reordered to utilize the features of R-M-W mode (Nicolau 1997).

Another mode of operation is the page mode, which takes the spatial locality into account. Consecutive accesses to two different scalar variables can be implemented as a single page mode operation if both are located in the same memory page (Nicolau 1997).

Panda et al. proposes the following algorithm for incorporating memory optimizing transformations into the CDFG for scheduling.

1. Cluster the scalar variables into groups and assign memory addresses.
2. For each basic block [A basic block is sequences of statements which may be entered only at the beginning, and when entered are executed in sequence without halt or possibility of branch], reorder the basic block to exploit R-M-W and page mode.

3. For each conditional node, perform code hoisting transformation if applicable.

4. For each innermost loop, perform loop splitting transformation if applicable.

5. For each loop, if single page is accessed in one direction, perform loop restructuring for page mode else perform loop unrolling and restructuring.

To know further details about the transformations refer (Kamal Khouri 2005) and (Aho 1986).

Scalar variables can be grouped into clusters of size L (where L is the size of a cache line). They are clustered in such a manner so that access to one of them causes all the L variables to be fetched into the cache, thereby reducing number of cache misses (Nicolau 1997). The memory subsystem consists of instruction cache and data cache on-chip and a large memory off-chip.

(Chakrabarti 2001) devised a procedure in which

1. Reduction of power consumption due to memory traffic by memory optimizing loop transformations - 50-80% of the power cost is due to memory traffic caused by transfer between ASIC (Application Specific Integrated Circuit) and off chip memories. The following loop transformation ordering is proposed in (Chakrabarti 2001).
   a) Compiler optimization Techniques (Joseph 2007) (Aho 1986)
   b) Memory access transformations
c) Loop reordering  
d) Loop fusion  
e) Loop interchange enabled by loop skewing  
f) Loop fusion enabled by loop normalization and loop peeling  
g) Loop tiling enabled by loop skewing  
h) Loop unrolling  

2. Using memory exploration procedure to choose a cache configuration that satisfies area, number of cycles and energy consumption - The cache that plays the dominant role (cache miss rate) is identified and assigned larger size. Dominant cache is one with high miss rate.  

2.2.1 Impact of Memory Optimizations on Power, Energy and Performance  

Many modifications can be made to the HLS flow making many optimizations possible. A sequencer can be introduced in the HLS flow, which allows computation dominated applications with not fully predictable memory access sequences to be optimized by data flow optimizations. Memory access is unpredictable if the part of the data is unknown before the execution of the application. These accesses are called dynamic accesses (Bertrand Le Gal 2008).  

Distance of memory from computational units represents the effort needed to fetch or store some data from or to the memory. This effort can be expressed in terms of energy and steps could be devised to reduce the energy consumption (Benini 2003). Hierarchical memory organizations reduce energy by enabling access to smaller banks. By mapping the frequently accessed locations to low hierarchy levels, total energy could be reduced.
The idea of partitioned memories could be used to save energy. The address space is subdivided into many blocks. The blocks can be mapped to different physical memory banks. Energy for memory access is reduced when memory banks are small. Energy is dependent on two independent variables. They are,

1. Access cost
2. Access profile.

GAUT, a HLS tool works by the above idea. Dynamic address computation balancing algorithms applied to the previously annotated graph in order to move some dynamic address computations from the data path to the sequencer unit. This optimization attempts to relocate some address computation near the memory banks. This improves the area and power consumption of the post synthesis circuit. GAUT also performs memory aware synthesis.

In the memory aware flow, a Signal Flow Graph (SFG) is constructed from the behavioural descriptions. This SFG is parsed and a memory table is created. The memory table contain information about every data structure in the algorithm (like arrays) and its allocation in memory or register. In the standard flow, the processing unit is synthesized without any knowledge on the memory mapping. The memory architecture is designed afterward and many optimization opportunities are lost (Corre 2004).

GAUT implements the ageing vectors in the hardware by storing the new value at the address of the oldest one in the vector. This requires one write when compared to the other methods where, the new value is written at the same address in memory usually at the end of the vector which required a shift of every other values of the signal in the memory to free the place for the new
value. This requires \( n \) reads and \( n \) writes in the memory, where \( n \) is the size of the vector (Gwenole Corre 2004).

Memory related activities are one of the major sources of energy consumption in embedded systems. The operating modes used in memories are active, standby, nap, power down modes. Maximum utilization of the power modes is possible by solving the following 3 tasks simultaneously (Chun-Gi Lyuh 2004). They include 1) Variable assignment to memory banks 2) Scheduling memory access operations 3) Selecting operating modes of banks

Preeti Panda et al. gives the advantages of multiport memory allocation over single port allocation. An optimized multiport allocation results in better energy characteristics. Here, port is treated as an independent schedulable resource. The goal is to reduce the switching activity on the memory address bus maintaining the same schedule length.

1. Coloring Phase - Identification of accesses that are likely to cause only small number of address bit transitions when accessed consecutively eg \( a[i], a[i+1] \).....

2. At every cycle in the schedule, each memory accesses are assigned to ports whose previous access had the same color.

This results in an energy optimized schedule. If the schedule length could be modified, the ports to memory accesses are assigned first using color information to ensure that each port is assigned memory access of the same color as far as possible, resulting in spatial locality being preserved to the maximum extent. This results in more energy aggressive schedule (Nicolau 1997).

An efficient memory binding optimizes the performance. This binding could be obtained by the efficacy of the performance estimation procedure. This procedure is computationally infeasible. A high level
performance estimation technique is devised by Kamal et al. that estimate the performance of the scheduled behaviour without actually performing scheduling (Kamal Khouri 2005).

Mapping of values into memories must be determined so that whole area is minimized. Clustering techniques can be used as suggested by Rouzeyre et al. Each value is considered as one memory composed of one register. A proximity coefficient P is associated with each pair of memories. P can be used to measure the advantage of merging the memories into one. The pair emph (mi, mj) which exhibits the maximum P value is merged into a new memory mk. mk requires less addresses than the sum of those of mi and mj since the values can share the same addresses (Rouzeyre 1991).

With the advent of embedded DRAM technology, the on chip memory can satisfy the entire memory requirements of an application. To utilize the available memory intelligently, application specific memory bank customization is suggested by Panda (Panda 1999). In contrast to the responsibility of the designer to derive the most efficient and general mapping, the number of banks and the assignment of variables to banks for a specific application is considered and thereby increasing memory access performance. An example given in (Panda 1999) compares the effect of mapping arrays to single bank and multiple banks.

Example: for (i=0; i ≤1000; i + +) {a[i] = a[i] + b[i] * c[2i]}

1. **Choice 1:** Mapping arrays to single bank memory. This mapping is inefficient, since different pages corresponding to a[i], b[i] and c[2i] are accessed in every iteration overwriting page buffer every new access and destroying locality of reference across different loop iterations.
2. **Choice 2:** Mapping arrays to multibank memories

Arrays a, b and c are mapped into different banks. Since each banked memory has an independent page buffer, there is no interference among the pages accessed during a single iteration. Effective access rate is faster. The memory banks could be customized by varying the number of banks in the architecture and determining the best variable assignment and estimate the memory access performance for each configuration. Panda studies the customization and assignment of arrays alone. Other variables and data structures could be customized also.

Gord Harling answers many questions related to DRAM like, Why DRAM?, Why embedded DRAM?, Why compile DRAM? etc., The DRAM compiler must produce inputs in minutes with internal rules that optimizes the memory for the lowest power and smallest area (Harling 2001).

### 2.3 TEMPERATURE OPTIMIZATIONS IN HLS

Thermal effects are becoming an important factor in the design of the integrated circuits due to the adverse impact of temperature on performance. Making all the phases of the design flow aware of this phenomenon helps in reaching faster design. Temperature aware HLS is challenging because of the multi-objective nature of the problem, with several conflicting objectives like throughput rate, power, peak temperature, and chip area. The following describes the various Temperature optimizations that could be achieved by making the normal HLS flow aware of the Temperature available.

Among the stages of the HLS that includes scheduling, allocation and binding, scheduling indirectly controls the temperature of the functional units, binding determines the activity of the functional units thereby affecting their thermal profile. Both these stages are necessary for efficient thermal management (Yen 2002).
(Rajarshi 2006) presented an integrated approach to thermal management in architectural synthesis. The normal HLS flow is unaware of the target technology. The normal HLS flow is modified to become a thermal aware flow with the following steps:

1. Start with a resource constrained latency minimized schedule.
2. Perform low power binding - Low power binding assigns operations to functional units such that the total switching capacitance of the resources is minimized.
3. Create a thermal aware floorplan and perform a thermal simulation of the resulting design considering thermal properties, thermal interaction between different functional units into account. Repeat steps 1,2 and 3 until the temperature of the hottest resource is decreased.

(Rajarshi 2006) proposed techniques to incorporate temperature awareness into HLS. Temperature need to be taken into account at all levels of abstraction of the design process. The authors developed temperature aware resource allocation and binding algorithms to be deployed with HLS, to accomplish this goal. A temperature aware binding controls the maximum temperature reached in a design more effectively, thereby minimizing the occurrence of hotspots. Temperature has an additive nature i.e., temperature at a given point in time will depend on the entire history of activity in the past. If this is not taken into account, cumulative heat will cause hotspots although the average activity seems to be well bounded.

The temperature aware resource allocation has the following steps as proposed in (Rajarshi 2006):

1. Preprocessing of the scheduled Data Flow Graph (DFG) For each resource type a comparability graph of the operations that this
resource can execute is created. Comparability graph is a DFG. Comparability graph is a compatibility graph with transitive orientation. If operations \( u \) and \( v \) are compatible, there exists a directed edge from \( u \) and \( v \) if \( \text{start}(v) > \text{start}(u) \). Each edge of the comparability graph has a weight equal to the switching activity, if the two operations are bound to the same resource consecutively. The operations bound to a resource will be executed in the topological order dictated by the comparability graph.

2. Perform topological sort on the graph.

3. Perform Relaxation determining parent of each vertex. This indicates that if an operation were assigned to a particular resource, that parent would be the best candidate to have been assigned on the same resource prior to this operation.

4. After all the vertices have been relaxed, paths which represent permissible binding of the operations to a resource is left. The algorithm selects the longest path and assigns the operations to the same resource.

5. The vertices in the selected path are removed from the comparability graph and a new comparability is built with the remaining vertices and the above steps are repeated.

Peak local temperature influences the reliability, packaging costs, cooling costs, bulk and performance of IC. These considerations are important for the portable devices. Power and thermal variations can also lead to significant timing uncertainty, requiring more conservative timing margins, thereby reducing performance (Shang 2006). Voltage islands enable core level power optimization for System-on-chip designs by utilizing a unique supply voltage for each core (Jingcao Hu 2004).
The thermal optimizations that could be achieved by incorporating thermal awareness into the High level as well as Physical level flows as in (Shang 2006) is listed below:

1. Slack distribution: TAPHS redistributes slack among operations in order to support more energy-optimal assignment of functional units to voltage islands.
2. Voltage partitioning: TAPHS uses on-chip voltage islands to optimize IC thermal profiles and energy consumption.
3. Thermal aware floor planning: Use techniques like simulated annealing, greedy algorithms and hill climbing to improve the floor plan until an optimized one is obtained. Since, physical synthesis is not the interest, it is not elaborated further.

As there is technology scaling, designs with multimillion transistors is becoming common. This increase in the number of transistors with the limited silicon area causes a number of crucial design problems to be solved. One of them is power consumption that leads to the increase in the chip temperature (Lim 2006). Since different components of a chip can have different execution profiles, the temperature of the components of chip are not uniform. Hence the designer has to design the chip in such a way that hotspots do not appear.

The proposed iterative thermal aware flow minimizes the maximum switching activity of modules. It includes the steps,

1. Generate an initial binding solution by formulating a low power binding problem into a network flow problem. This initial solution attempts to minimize the quantity of the total switching activity.
2. For each operation bound to the module, attempt rebinding the operation by rerunning the network flow on an updated network.
3. Among the rebindings, the algorithm selects the one which results in the largest reduction in the switching activity and performs the rebinding.

4. The algorithm repeats steps 2 and 3 until there are no more reductions possible.

The Network flow method has got many advantages. The network flow method explores the search space globally and finds a solution close to the optimum. To deal with large designs, where the run time is quite long, the network is divided into a set of multiple segments by cutting the network (Network partition strategy).

(Junbo 2008) highlighted the importance of thermal area trade off. More number of resources can reduce the peak temperature, however large number of resources indices high leakage power and violates area constraint. Therefore, thermal area trade off is necessary. The method proposed identifies the resource number for proper optimal thermal area trade off and obtains rough synthesis results as the initial solution for incremental improvement in future. The thermal aware flow proposed in (Junbo 2008) consists of the following steps:

1. Resource Number Selection (RNS) - RNS aims to identify an array of resources numbers, using which all the resources almost have the same power density. This is done because the uniform power density distribution would achieve the lowest peak temperature. To evaluate the quality of resources number, a total flow from high level synthesis to thermal analysis has to process.

2. Heuristic Scheduling and Binding (HSB) - In order to effectively validate whether resources number obtained by RNS is proper, a thermal aware high level synthesis algorithm is needed. Moreover,
it generates the solution as a good starting point for future improvement.

The two heuristics considered are

1. Obtain resources with uniform power.
2. Obtain solution with minimal total power.

Low power may lead to low temperature. The better solution between 1 and 2 is selected.

Existing low power design techniques cannot adequately address thermal issues since their optimization objectives fail to capture the spatial nature of on-chip thermal gradients. (Vyas 2009) proposes an algorithm for thermally aware low power behavioural synthesis that concurrently minimizes the average power and peak chip temperature.

Temperature awareness is made as a part of low power behavioural synthesis. The following steps are proposed by the author,

1. The input DFG, is simulated with typical input traces to create input switching power tables for each resource type in the target data path circuit.
2. Task schedules, resource bindings, power models, an RTL design library, floorplanner, and a thermal model of the IC package are then used to evaluate the IC temperature profile, power, area, and performance of designs synthesized by the steps below.
3. To address the multi objective nature of the temperature aware low power synthesis problem, a two stage simulated annealing algorithm is used. The objectives optimized in the annealing stage are

**Stage 1:** High temperature annealing,
**Stage 2:** Low temperature annealing.

In the first stage of the annealing, a floorplan driven low power HLS is performed, given the resource constraints for the data path. The second stage is a low-temperature annealing stage, that takes the best low-power solution returned by the first stage, and uses a set of thermally aware simulated annealing algorithm moves to minimize peak module temperature and create an even thermal distribution across the chip.

The HDL compilers could be made thermal aware so that they reduce the peak temperature. Such an approach is proposed by (Benjamin 2007), where the Very Long Instruction Word (VLIW) compilers provide temperature control and reduction techniques which minimizes the peak temperature in the VLIW processor’s functional units.

1. **Technique 1:** The algorithm, Temperature aware instruction binding technique TempIB effectively binds the instructions executed in parallel to the coolest possible functional units for a given fixed schedule. It generates, for each instruction in a scheduled instruction word, a priority queue of the coolest functional units that can execute the instruction, and rebinds it to the coolest possible unit, considering the temperature as well as the power consumed by the instruction. This technique lowers the peak temperature of the initial design upto 13.82%. The algorithm has the following steps:
   
   a) For each instruction a priority queue is created, sorting the Functional Units (FU) on which the instruction can be executed from minimum to maximum temperatures as well as considering the power consumption of the instruction.
   
   b) In case there are no conflicts the instructions are bound to the FUs at the bottom of the queue. If there is a conflict, it is
resolved by looking at the next FU in the priority queue and the coolest unit of the second element of the queue is swapped by the conflicting FU in the queue.

c) Check for conflicts again until no more conflicts exist. This algorithm guarantees that each instruction is always bound to the coolest possible unit.

2. **Technique 2:** Temperature aware NOP instruction technique TempNOP. This technique is based on the assumption that in order to cool down a unit, the unit needs to rest in order to stop temperature build-up. A unit is rested by inserting NOPs appropriately.

The tasks could be assigned dynamically to FUs based on the temperature of the FU. Each Functional unit or units where a hotspot is most likely to occur would have a thermal sensor. The reading of these sensors could then be used to determine which units should be selected to execute the instruction and which units should be turned off to allow these to cool down. However, an excessive number of thermal sensors would affect the power consumption.

In the past, Dynamic Voltage and Frequency Scaling (DVFS) has been widely used for power and energy optimization in embedded system design. (Yongpan 2007) proposed a design time thermal optimization for embedded systems using DVPS. They examined the differences between the optimal energy solutions and optimal peak temperature solutions and they had proved that the optimizing the energy consumption can lead to unnecessarily high temperature. A Thermal Optimization (TO) problem is formulated whose optimization objective is to minimize the maximum temperature of all tasks considering the constraints like the execution time, task start times and operating voltages. An Energy Optimization (EO) problem is also formulated whose goal
is to minimize the total execution energy of all tasks by controlling processor voltages subject to the same constraints as the TO. Another optimization problem is Thermal Constrained Energy Optimization (TCEO) that permits the energy consumption to be minimized while guaranteeing that the threshold temperature is not violated.

Clock skew is a phenomenon in synchronous circuits in which the clock signal arrives at different components at different times. This can be caused by many different things, such as wire-interconnect length, temperature variations, variation in intermediate devices etc., (ChunChen 2008) made attempt to reduce skew variation considering time variant temperature variation. In a circuit, different routing paths with different temperature gradients will have different skews. Most of the existing synthesis techniques use roughly manhattan distance to simulate and evaluate the delay between each two nodes. Manhattan distance is not exact in some cases. Hence, thermal aware routing considering both distance and temperature correlation factor which is not only an accurate method for evaluating the delay between each node but also reduce the skews and skew variation significantly.

Icarus Verilog compiler is a hardware compiler that performs the HLS process. It consists of scanner (lexor.l), parser (parse.y), elaborator (elab_expr.cc), optimizer (cproc.cc), scheduling (synth.cc) and allocation (alloc.cc). This will produce a netlist as output. Netlist is a data structure that contains various components.

There are various technology specific attributes that could be added to the High Level Synthesis process during the parsing state itself. Some examples of the technology specific attributes includes,

1. ram_style attribute,
2. Mode block_power1
3. Mode block_power2
4. Safe temperature threshold
5. Final temperature

A two phase outline floorplanning framework is presented in (Behman 2017) to reduce the peak temperature. The dead space is distributed among the blocks of the floorplan and fixed outline floorplanning is performed to minimize the peak temperature.

Application of the above techniques in the HLS design methodology will result in an optimized circuit design.

This chapter presents the detailed survey for power, memory and temperature optimization techniques in the High-Level Synthesis presented in the research literature. This survey helps us to obtain the correlation between the power and temperature, memory and temperature and power and memory. Both power and memory must be exploited to obtain temperature minimization of the IC.