Published in IET Circuits, Devices & Systems
Received on 24th February 2009
Revised on 27th October 2009
doi: 10.1049/iet-cds.2009.0049

ISSN 1751-858X

On-chip thermal optimisation by whitespace reallocation using a constrained particle-swarm optimisation algorithm

D. Chatterjee[1] *T.W. Manikas*[2]

[1]*Department of Electrical Engineering, Stanford University, 161 Packard Building, 350 Serra Mall, Stanford, CA-94305-9505, USA*
[2]*Department of Computer Science and Engineering, Southern Methodist University, POB 750122, Dallas, TX-75275-0122, USA*
E-mail: manikas@lyle.smu.edu

Abstract: The density of chip power dissipation has been increasing steadily over the past several years. High operating temperatures and the existence of hotspots are degrading chip performance and undermining chip reliability. Reducing maximum on-chip temperatures is becoming increasingly important as technology scales below 65 nm. Existing thermal floorplanner compact blocks at the lowest leftmost position allowed by the floor plan encoding. Such compaction minimises chip area but is sub-optimal for wire length and thermal objectives. It is possible to move the blocks in the whitespace (unoccupied chip area) to minimise maximum on-chip temperature without affecting the overall chip area and with a minimal wire length increment of ~2–3%. However, reallocation of whitespace for thermal optimisation has not been addressed by researchers to date. Here, the development of a constrained particle swarm optimisation algorithm to find an optimal solution to the problem has been described. Simulation results on MCNC benchmark circuits indicate that this method can reduce the maximum on-chip temperature of thermal-aware floor plans by 0.58–7.10°C.

1 Introduction

Floorplanning is an important stage of the very large-scale integration circuit design cycle. Given a set of rectangular blocks, floorplanning determines a non-overlapping placement of the blocks to minimise chip area and wire length. In order to facilitate hierarchical design, the classical rectangle packing problem has been replaced by a fixed-outline soft block packing problem [1]. Minimising the Floorplan whitespace, which is the chip area unoccupied by blocks, is therefore no longer an issue. In fact, a minimum amount of whitespace is essential for gate sizing, buffer insertion, congestion improvement and cross-talk noise reduction. As technology scales below 65 nm, it is likely that the minimum requirement for whitespace will be dictated by thermal objectives. However, there is no existing method for reallocation of whitespace for thermal optimisation. In this paper, we demonstrate that it is possible to lower the operating temperature by redistributing whitespace.

The floorplanning problem has been studied in depth over the past two decades. Introduction of placement constraints, net-length constraints and thermal objectives have increased the complexity of floor plan optimisation. Simulated annealing (SA) [1], genetic algorithms (GA) [2] and multiobjective evolutionary algorithms [3] are popular in determining the optimal floor plan. These methods model the block placements using structures such as the sequence pair (SP) [4], bounded slicing grid [5], O-tree [6] and B* tree [7] representations.

In recent times, the microprocessor clock frequency has doubled every three years, and supply voltage scaling has been unable to stem the rise in power dissipation [8]. To keep designs within power budget, multi-core processors are being launched and convex-optimisation based processor speed control techniques are being developed to keep the hotspot temperature in control [9]. However, the benefits of whitespace reallocation still

remain to be exploited in both single-core as well as multi-core processors.

Reducing the peak temperature is important because high temperature has severe detrimental effects on the performance of the chip. First, an increase in chip temperature deteriorates the chip's reliability due to electro-migration. Second, due to the temperature dependence of carrier mobility, the current driving capability of transistors decreases with increase in temperature. Also, high temperature increases interconnect resistivity. Low current driving capability of transistors coupled with high interconnect resistivity may increase interconnect delay to such an extent that timing constraints might be violated. Furthermore, high temperature increases the risk of a thermal runaway caused by its exponential dependence on leakage power. In order to avoid failures, thermal packages are designed to withstand peak power dissipation. As the peak temperature increases, so does the cost of cooling. The estimated increase in the overall cost of chip due to every watt of power dissipated above 35–40 W is $1/W [10]. The motivation behind whitespace reallocation is to improve performance and reliability and reduce packaging cost by reducing the maximum on-chip temperature.

2 Previous research

A wealth of literature exists on thermal optimisation of 2D integrated circuits (ICs) that aid in reducing the packaging cost. Such cooling solutions are provided either at the either package-level, chip-level or architecture-level. At the package level, improved cooling systems (heat sink, air circulation) can significantly cool the chip. A popular chip-level technique is dynamic thermal management (DTM) [11], which is a major design feature of the Intel Pentium 4 microprocessor [12]. The key observation behind DTM is that typical applications running on a processor would very rarely cause the processor temperature to ramp up beyond a certain threshold value. In DTM-enabled processors, the thermal package is targeted for the threshold temperature which is much less than the maximum attainable temperature. In the rare event when the on-chip sensors detect that the maximum temperature is approaching the threshold value, a controller dynamically throttles the frequency of the chip to reduce dynamic power dissipation and runs the processor at a lower frequency until the thermal virus section of the application is executed. Thus, DTM reduces packaging cost, albeit by sacrificing performance. Architecture level techniques [13–15] have the advantage of reducing temperature without slowing down the processor. The importance of micro-architecture in reducing the hotspot temperature has been underscored in [13]. Skadron et al. have developed a compact-substrate thermal model to accurately simulate the temperature on ICs. The model is available in the form of a tool named HotSpot. Thermal aware floorplanning typically involves the use of HotSpot within a GA floorplanner [14] or an SA floorplanner- Hotfloorplan [15]. Fast thermal floorplanning techniques use thermal metrics computed from heat diffusion [16] or power-density gradients [3]. To date, all research on thermal floorplanning has focused on determining the relative positioning of the blocks that minimises the peak temperature. Thermal-aware floor planners usually compact blocks towards the bottom-left corner to minimise chip area. Such compaction is suboptimal for minimising thermal objectives. We propose a method by which the peak on-chip temperature can be minimised by determining the optimal positioning of the blocks (or processors) in the whitespace.

Whitespace reallocation is not uncommon in physical design. Whitespace planning is utilised for improving stability of placement algorithms used in physical synthesis [17]. Post-floorplanning whitespace reallocation has been used for wire length minimisation [18]. Using the semi-perimeter of the bounding box as an approximation for the net-length, it is possible to formulate the post-floorplanning wire length minimisation (by whitespace reallocation) problem using linear programming and solve it with the help of a min-cost flow-based algorithm [18]. Other whitespace reallocation methods for optimising various non-thermal objectives are provided in [19, 20]. However, reallocation of whitespace to optimise maximum on-chip temperature has not been reported so far. In this paper, we present a heuristic based on Swarm Intelligence to locate an optimal thermal placement. We select the SP floor plan representation because of its ability to represent non-slicing floor plans. However, our method can also be easily extended to other floor plan representations.

3 SP representation

A floor plan with N blocks can be represented by a pair of sequences ($\mathbf{S_1}$, $\mathbf{S_2}$) each having N integer elements. This pair imposes certain constraints on the relative positioning of the blocks on chip. For every element i in the sequence, we can find sets **B(i)** and **L(i)** denoting blocks that are before (left of) i and lower than i in the floor plan, respectively, where

$$\mathbf{B(i)} = \{j | j \text{ is before } i \text{ in both } \mathbf{S_1} \text{ and } \mathbf{S_2}\} \quad (1)$$

$$\mathbf{L(i)} = \{j | j \text{ is after } i \text{ in } \mathbf{S_1} \text{ and before } i \text{ in } \mathbf{S_2}\} \quad (2)$$

Based on the constraints imposed by the set **B(i)**, a horizontal constraint graph **GH(V, E)** can be constructed as follows: The vertex set **V** consists of a source s, sink t and N vertices labelled with module names. The edge set **E** consists of directed edges. Edges are drawn from source to each of the vertices. Each vertex in turn is connected to the sink t. A directed edge exists from vertex j to i in **GH(V, E)** iff $j \in$ **B(i)**. Vertex weights are zero for source and sink and equal to the module width for the rest of the vertices.

IET Circuits Devices Syst., 2010, Vol. 4, Iss. 3, pp. 251–260
doi: 10.1049/iet-cds.2009.0049

The second part of the SP is represented by a vertical constraint graph **GV(V, E)**, which can be constructed in a manner similar to the horizontal constraint graph. For this graph, a directed edge exists from vertex j to i in **GV(V, E)** iff $j \in$ **L(i)**.

The x-coordinate of the lower left corner of a particular block in the floor plan represented by ($\mathbf{S_1}$, $\mathbf{S_2}$) is the length of the longest path from source to the vertex representing the block. The longest path from source to sink in **GH(V, E)** represents the minimum width of the packing imposed by the SP ($\mathbf{S_1}$, $\mathbf{S_2}$). The height of the packing can be calculated in a similar manner for **GV(V, E)**. Fig. 1 shows a floor plan along with its corresponding horizontal and vertical constraint graphs.

4 Problem formulation

Mathematically, the problem of thermal optimisation by whitespace reallocation can be stated as follows: Given a set of N hard modules (macro cells or processors) $\mathbf{\Pi} = \{\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_N}\}$, a specific dimension set $\mathbf{D} = \{(w_i, h_i) \mid w_i$ is the width and h_i is the height of module $i\}$, a set of power values $\mathbf{\Phi} = \{\varphi_i \mid \varphi_i$ is the average power dissipated by block $i\}$ and an SP ($\mathbf{S_1}$, $\mathbf{S_2}$) imposing certain topological constraint on the blocks, find a non-overlapping placement of $\mathbf{\Pi}$ given by $\mathbf{P} = \{(x_i, y_i) \mid x_i$ and y_i are the coordinates of the lower left corner of module $i\}$ that minimises the maximum on-chip temperature T

$$\min \; T{:}P \rightarrow \Re \tag{3}$$

subject to specific constraints on the width and height of the chip. The exact nature of these constraints depends on the mode of floorplanning and is explained later in this section.

On-chip temperature can be calculated analytically from the average power dissipated in the modules. The generalised equation for a transient temperature analysis in a 3D substrate is given by [21]

$$\nabla.(k(r, T)\nabla T(r, t)) + Q(r, t) = C_p \rho \frac{\partial T(r, t)}{\partial t} \tag{4}$$

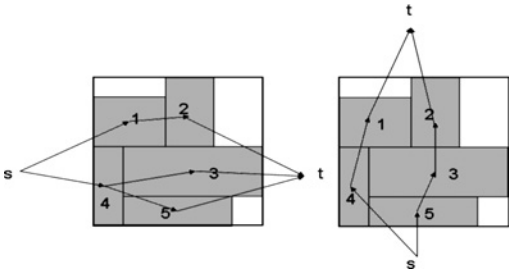

Figure 1 *Floorplan with five modules and corresponding horizontal and vertical constraint graphs*

subject to the boundary condition

$$k(r, T)\frac{\partial T(r, t)}{\partial n_i} + h_i T(r, t) = f_i(r) \tag{5}$$

where T is the temperature as a function of the position vector $\boldsymbol{r}$ and time t; k, ρ and C_p are the thermal conductivity (W/m°C), density (Kg/m^3) and specific heat (J/kg°C) of the material respectively, Q is the heat generation rate (W/m^2), $\partial/\partial n$ represents the differentiation along the outward normal drawn at the boundary surface and f_i is any arbitrary function. At steady state

$$\frac{\partial T(r, t)}{\partial t} = 0 \tag{6}$$

Substituting (6) in (4) and neglecting the temperature dependence of thermal conductivity, the equation for computing the steady-state temperature becomes

$$k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) + Q(x, y, z, t) = 0 \tag{7}$$

In view of the complex boundary condition involved, (4) is solved most commonly using numerical techniques such as finite difference method or finite-element method. An analytical solution of (7) can be obtained by simplifying the boundary conditions. The temperature profile on a planar substrate with insulated bottom face containing a single power-dissipating module is obtained by solving (7) and is given by [22]

$$\theta(a, r) = \begin{cases} c_1 I_0(mr^+) + \dfrac{t}{2ab_i}, & 0 \le r^+ < 1 \\ c_4 K_0(mr^+), & r^+ > 1 \end{cases} \tag{8}$$

$$\theta(a, r) = \frac{T - T_{amb}}{P_m{}^* a/k}, \quad r^+ = \frac{r}{a}, \quad b_i = \frac{bt}{k}$$

$$m = a\sqrt{\frac{2h}{kt}}, \quad c_1 = \frac{-t/2ab_i}{I_0(m) + K_0(m)I_1(m)/K_1(m)}$$

$$c_4 = -c_1\frac{I_1(m)}{K_1(m)}, \quad a = \sqrt{\frac{WH}{\pi}}$$

where r is the radial distance from the centre of the module; W, H, P_m, k and t are the width, height, power consumption, thermal conductivity and thickness of the module respectively; I_0 and I_1 are the zeroth and the first order modified Bessel function of the first kind; K_0 and K_1 are the zeroth and the first order modified Bessel function of the second kind. Equation (8) suggests that the temperature falls off like a Gaussian surface with the radial distance. In the presence of more than one power dissipating module, the overall temperature at any point is obtained by superposition of the temperature values produced by each individual power-dissipating module. The equation for temperature T_i at the centre of module

i is given by

$$T_i = \sum_{i=1}^{N} T_{ij} \tag{9}$$

where T_{ij} is the contribution of temperature at the centre of module i because of power dissipation in module j, as calculated using (8). Evidently, each term T_i being the sum of Gaussian-like function is itself a non-linear function of the block location. Thus, the objective function (T) of the optimisation problem (3) is non-linear.

In general, there are three constraints that have to be satisfied to generate a valid placement: First, modules should not overlap each other. Second, the modules must be packed within a rectangle of specified chip dimensions (in fixed-outline floor planning) or the boundaries imposed by compaction of modules towards the lower left corner (in conventional floor planning). Third, the relative order of the blocks as obtained from floor planning must be preserved. Mathematically, the constraints may be stated as

$$x^j + w^j \le x^i \quad \forall j \in B(i), \quad \forall i \in \Pi \tag{10}$$

$$y^j + h^j \le y^i \quad \forall j \in L(i), \quad \forall i \in \Pi \tag{11}$$

$$0 \le x^i \le W_{\text{chip}} - w^i \quad \forall i \in \Pi \tag{12}$$

$$0 \le y^i \le H_{\text{chip}} - h^i \quad \forall i \in \Pi \tag{13}$$

where W_{chip} and H_{chip} are the width and height of the chip, respectively. The non-overlapping constraints are imposed by (10) and (11); chip-boundary constraints are imposed by (12) and (13). The relative ordering constraint is imposed by the SP ($\mathbf{S_1}$, $\mathbf{S_2}$) of the floor plan. The above conditions make the solution space highly constrained.

The non-linear nature of the thermal objective and the various constraints involved render reallocation of whitespace for thermal optimisation to be a constrained non-linear optimisation problem (CNOP) [23]. Owing to the complexity and unpredictability of CNOP, there is no deterministic solution to this problem. Because of the non-discrete nature of the solution space, optimisation techniques like GA and SA cannot be applied to solve this problem. We, therefore, develop a constrained particle swarm optimisation (CPSO) framework to solve this problem.

5 Particle swarm optimisation

Particle swarm optimisation (PSO) introduced by Eberhart and Kennedy [24] is a swarm intelligence technique inspired by the social behaviour of bird flocking. A population of collaborating agents (particles) flies in a multidimensional search space. These agents have a common fitness function that they want to minimise by locally searching the landscape and globally coordinating with each other. Each particle in the swarm retains the memory of the best position (locally lowest fitness) it has encountered in the past and is aware of the global best position (globally lowest fitness) of the swarm. Based on this information and the inertia of its motion, particles keep updating their velocity in search of global minima. The procedure continues for several iterations and the global best position is selected as the final solution. Our CPSO flow is shown in Fig. 2. In this problem, each particle in the swarm moves in a constrained $2N+2$ dimensional space, where N denotes the number of blocks in the floor plan. The position P^r and velocity V^r of each particle r in the swarm are represented by vectors as follows

$$P^r = \langle x^0, y^0, x^1, y^1, x^2, y^2, \ldots, x^N, y^N \rangle \tag{14}$$

$$V^r = \langle v_x^0, v_y^0, v_x^1, v_y^1, v_x^2, v_y^2, \ldots, v_x^N, v_y^N \rangle \tag{15}$$

Each particle actually represents a floor plan and therefore (x^i, y^i) denote the coordinates of the lower left corner of the block i in the floor plan and (x^0, y^0) are the source coordinates. (v_x^i, v_y^i) are the velocity-components of block i in particle r along their respective dimensions.

5.1 Legal random initial placement

In order to start the CPSO procedure, the position of each particle in the swarm must be initialised. Positions obtained by random initialisation are mostly inconsistent with the constraints mentioned earlier. Thus, random initialisation is excessively time-consuming. We present an algorithm for generating feasible initial solutions. A key concept that we use here is that of a block slack [1]. The slack of a block in

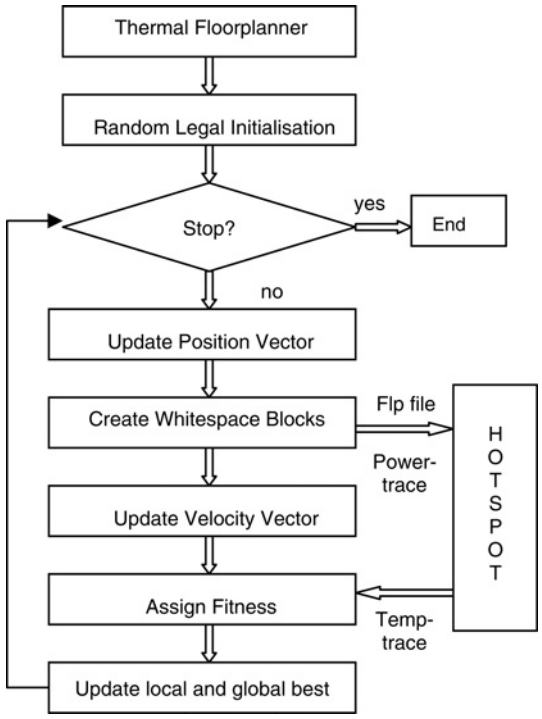

Figure 2 *CPSO flow*

IET Circuits Devices Syst., 2010, Vol. 4, Iss. 3, pp. 251–260
doi: 10.1049/iet-cds.2009.0049

the x-direction may be defined as the distance through which the block is translated horizontally if the blocks are compacted at the upper right corner instead of the bottom left corner. The lower bound $x^i_{\min}$ and $y^i_{\min}$ on the position of a module imposed by (10)–(13) when all blocks are compacted towards the bottom-left corner is obtained by applying the longest path algorithm on the constraint graphs. The upper bound $x^i_{\max}$ and $y^i_{\max}$ imposed by (10)–(13) can be computed using the pseudo-code SP_EVAL_REV presented in [1]. The slacks of the blocks can then be computed using

$$X\text{Slack}^i = x^i_{\max} - x^i_{\min} \tag{16}$$

$$Y\text{Slack}^i = y^i_{\max} - y^i_{\min} \tag{17}$$

The method for legal random initialisation of position and velocity is presented in the pseudo-code LEGAL_RAND_INIT (Fig. 3). The algorithm starts off by initialising the coordinates of the source vertex to zero. The variable R^i and T^i represents the minimum value of the x-coordinate and y-coordinate of block i computed after placing all blocks previous to it in the constraint graphs. Thereafter, the effective slack is computed using the following equations

$$X\text{Slack}^i_{eff} = x^i_{\max} - R^i \tag{18}$$

$$Y\text{Slack}^i_{eff} = y^i_{\max} - T^i \tag{19}$$

Clearly, any value of x_i in the range $[R_i, R_i + X\text{Slack}^i_{\text{eff}}]$ will satisfy all the constraints. All components of the velocity vector are initialised to zero, and the local best position of each particle is initialised by the current position vector of the particle. The global best position is initialised by the position vector of the particle having the smallest fitness value. The procedure for fitness assignment will be discussed later.

5.2 Update position and velocity vector

The general rules for updating the position and velocity vector in a PSO problem are straightforward and are given by the following equations

$$P^i \leftarrow P^i + V^i \tag{20}$$

$$V^i \leftarrow wV^i + c_1R_1(P^i_{\text{lbest}} - P^i) + c_2R_2(P_{\text{gbest}} - P^i) \tag{21}$$

where w is the inertial constant, c_1 and c_2 are constants that dictate the proportion of cognitive and social components in the velocity vector. At any instant of time, the velocity vector can be decomposed into three components. The first component is along the direction of velocity vector in the previous iteration and is known as the inertial component. The second component is along the direction of the local best position w.r.t current position, termed as the cognitive component. The remaining component is along the direction of the best known global position w.r.t the current position and is coined as the social component. The relative proportion of w, c_1 and c_2 affect particle behaviour by determining its tendency to trust a particular direction in comparison with the others. For example, a high c_1 to c_2 ratio indicates a less social particle because it trusts its own knowledge over collective knowledge and has greater tendency to search around its own local minima. The mechanism of position and velocity updates is explained in Fig. 4. $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are random vectors, with each component being a uniform random number between 0 and 1. Updating the position and velocity vector according to (20) and (21) does not necessarily restrict the particle within the feasible region. It has to be ensured that particle position is updated only if the newly generated position vector lies in the feasible region. Because of extreme limitations in degrees of freedom of each particle in this problem, the particles generally freeze once they violate the legality criteria. Two steps are adopted to circumvent this problem. First, the particle velocity vector is set to zero whenever the particle hits the feasible

begin
input: (S_1, S_2) /* Sequence Pairs*/
output: $P^r, V^r, P^r_{lbest}, P_{gbest}$
1: $B(i) = \{j \mid j \textit{ is before } i \textit{ in } S_1 \textit{ and } S_2\}$
2: $L(i) = \{j \mid j$ is after i in S_1 and before i in $S_2\}$
3: $x^0 \leftarrow 0,\ y^0 \leftarrow 0$/* Initialise source coordinates*/
4: $SP\_EVAL\_REV(S_1, S_2)$/* Finds $x^i_{\max}, y^i_{\max} \forall i \in \Pi$ */
5: **do**
6: **if** $x^j \neq \phi\ \ \forall j \in B(i)$
7: $R^i \leftarrow \max\{x^j + w^j \mid j \in B(i)\}$
8: $r \leftarrow random\ in\ [0,1]$
9: $XSlack^i_{eff} \leftarrow (x^i_{\max} - R^i)$
10: $x^i \leftarrow R^i + XSlack^i_{eff} * r$
11: $v^i_x \leftarrow 0$
12:**while** $\exists i \in \Pi \mid x^i = \phi$
13: **do**
14: **if** $y^j \neq \phi\ \ \forall j \in L(i)$
15: $T^i \leftarrow \max\{y^j + h^j \mid j \in L(i)\}$
16: $r \leftarrow random\ in\ [0,1]$
17: $YSlack^i_{eff} \leftarrow (y^i_{\max} - T^i)$
18: $y^i \leftarrow T^i + YSlack^i_{eff} * r$
19: $v^i_y \leftarrow 0$
20:**while** $\exists i \in \Pi \mid y^i = \phi$
21: $P^r_{lbest} \leftarrow P^r$
22: $P_{gbest} \leftarrow \min_{P^r} f(P^r)$
end

Figure 3 *LEGAL_RAND_INIT*

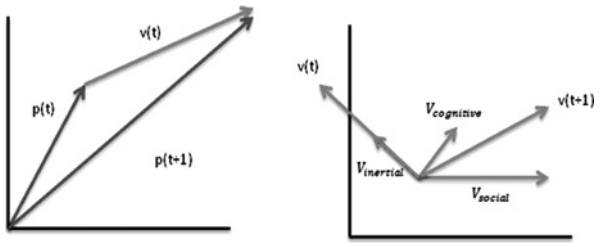

Figure 4 *Position and velocity updates in PSO*

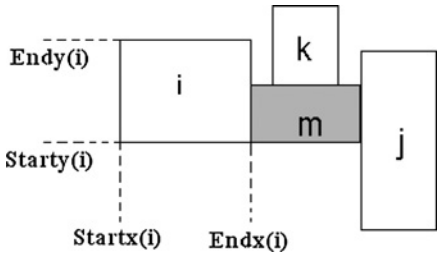

Figure 6 *Whitespace block at LR corner of block i*

boundary. This causes the particle to loose inertia, thereby preventing it from stagnating in subsequent iterations. Second, only those components of velocity vector are updated which have considerable slack associated with them. See UPDATE_POS_AND_VEL (Fig. 5) below.

5.3 Create whitespace blocks

Simulating temperature at block level granularity requires splitting up the whitespace into rectangular blocks. Whitespace blocks are treated as ordinary blocks with zero power dissipation. The method for splitting whitespace into rectangles is a two-step process: (i) Find whether whitespace exists at the lower-right (LR) and upper-left (UL) corner of each block and accordingly modify the appropriate flags associated with the block, (ii) depending on the status of the flags, create whitespace block m (as shown in Fig. 6) at the LR and/or UL corners until all whitespace has been enclosed by rectangles. CREATE_WHITESPACE_LR (Fig. 7) provides the pseudo-code for creating a whitespace rectangle at the LR corner of a block.

5.4 Parameter selection

In the PSO algorithm, there are several parameters that need to be tuned. The population size is usually between 10 and 30. In this case, a population size of 10 gives a sufficiently

begin
1: **for** $r \in Swarm$
2: $P^r_{old} \leftarrow P^r$ /* Store Position Vector*/
3: $P^r \leftarrow P^r + V^r$ /*Update Position Vector*/
4: **if** $(CHECKLEGALITY(P^r))$
5: $P^r \leftarrow P^r_{old}$ /*Restore Position Vector*/
6: $V^r \leftarrow 0$
7: **for** $j \in \Pi$
8: **if** $(XSlack^j > \varepsilon)$ /* ε is a small number> 0*/
9: $v^j_x \leftarrow wv^j_x + c_1 r_1 (x^j_{lbest} - x^j) + c_2 r_2 (x_{gbest} - x^j)$
10: **if** $(YSlack^j > \varepsilon)$ /* ε is a small number > 0*/
11: $v^j_y \leftarrow wv^j_y + c_1 r_1 (y^j_{lbest} - y^j) + c_2 r_2 (y_{gbest} - y^j)$
end

Figure 5 *UPDATE_POS_AND_VEL*

begin
1: **do**
2: *For all block i* /*including whitespace blocks*/
3: $If\,(Flag\_LR(i) \neq 1)$
4: *Create m* /*m is a whitespace block*/
5: $Startx(m) = Endx(i)$
6: $Starty(m) = Starty(i)$
7: *Find j:* $Startx(j)$ *is smallest value* $> Endx(i)$ *and* $Starty(j) \leq Starty(i) < Endy(j)$
8: $Endx(m) = Startx(j)$
9: $Flag\_LR(m) = 1$
10: *Find* $k : Starty(k)$ *is smallest value* $> Starty(i)$ *and* $Endx(k) > Endx(i)$ and $Startx(k) < Startx(j)$
11: $Endy(m) < Starty(k)$
12: $Pushback(m)$
13: $CHECK\_FLAG\_UL(m)$
14: **while**$(\exists i \mid Flag\_LR(i) \neq 1)$
end

Figure 7 *CREATE_WHITESPACE_LR*

good result in a small runtime. The inertial constant w, cognition learning rate c_1 and social learning rate c_2 are set to 0.98, 0.99 and 0.97, respectively. The termination of the CPSO procedure can be defined by the user. We set the maximum number of iteration to 400 and decide to terminate the process if fitness has not improved in the last 50 generations.

5.5 Assign fitness

To assign fitness values, we integrate HotSpot [13] with our CPSO. HotSpot uses the well known duality between electric and thermal quantities to generate an equivalent RC model of the chip. For the steady-state temperature analysis, the

doi: 10.1049/iet-cds.2009.0049

equivalent circuit consists of a resistor circuit as shown in Fig. 8. Power dissipating blocks are modelled as constant current sources and the lateral heat diffusion is modelled by resistors. Given the location of each module, whitespace blocks and their power consumption, HotSpot calculates the temperature at each node (centre of the block) by solving nodal equations for the equivalent circuit. After temperature evaluation, particle r is assigned a fitness value f_r given by

$$f^r = \max_{j\{T_j\}} \tag{22}$$

6 Simulation results

At first, we used Hotfloorplan [15] to generate initial floor plans for MCNC benchmarks circuits. Block power densities were randomly assigned using a uniform distribution with mean μ. Present generation high performance ICs have μ in the range of 1–2 W/mm^2, and μ is expected to reach 5 W/mm^2 for technologies below 50 nm [8]. We present our results for various values of μ from 0.5 to 4 W/mm^2. Table 1 shows the reduction in maximum on-chip temperature (ΔT) due to whitespace reallocation when $\mu = 1$ W/mm^2. For large benchmarks like hp, ami33 and ami49, ΔT was found to be 3.46, 5.87 and 7.10°C, respectively. On the other hand, apte and xerox had minimal room for thermal optimisation because of very small slacks associated with the blocks after floor planning.

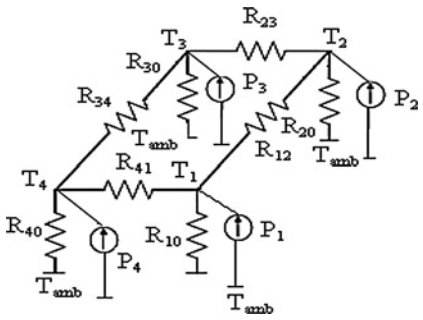

Figure 8 *Resistor network for thermal simulation*

In the next set of experiments, we analysed in detail the effect of mean power-density and mode of floor planning on ΔT and also the wire length trade off involved. Initial floor plans generated by Hotfloorplan [15] were fed to the PSO for the whitespace reallocation. In Mode-I (fixed-outline floor planning), initial floor plans were made to satisfy a fixed-outline constraint and were additionally optimised for wire length (W) and maximum on-chip temperature (T). The fixed outline constraint implicitly imposes area and aspect ratio constraint on the initial floor plan. Table 2 shows the results of applying CPSO on an initial floor plan of ami33 benchmark circuit. The initial floor plan had 20% whitespace and unit aspect ratio. ΔT was 0.85°C for $\mu = 0.5$ W/mm^2. As μ was increased, the benefits of whitespace reallocation became more pronounced. For $\mu = 4$ W/mm^2, ΔT was 6.65°C. This indicates a 7.8× increment in ΔT for an 8× increment in μ, an almost linear increase. In Mode-II, the whitespace reallocation of the same initial floor plan was done, but this time taking the boundaries of the left-compacted floor plan as the chip boundary. For $\mu = 4$ W/mm^2, ΔT was 3.5°C which is much less than the corresponding ΔT value in Mode I. Observe that for $\mu = 4$ W/mm^2, the internal whitespace (IW) of 16.21% contributes towards 3.5°C reduction in T and external whitespace (EW) of (20–16.21)% = 3.79% contributes towards (6.65–3.5)°C = 3.15°C reduction in T. Clearly, EW is more effective in reducing T. The intuitive reason behind this is that the availability of IW does not necessarily guarantee the availability of a block slack. On the contrary, EW increases available slack for all modules thereby giving the CPSO more degrees of freedom for temperature optimisation. The average increase in HPWL due to post floor planning thermal optimisation was 3.03% for fixed-outline floor planning and 2.67% for conventional floor planning of ami33.

Table 3 shows the results on ami49 benchmark circuit. The initial floor plan had 20% whitespace and unit aspect

Table 1 Benchmark characteristics, IW after floorplanning and maximum on-chip temperature before and after applying CPSO

MCNC benchmarks	Blocks	Nets	Area (mm^2)	IW (%)	Maximum temperature (°C)			
					Initial	Final	ΔT_{max}	Time, s
apte	9	97	46.56	3.8	81.99	81.41	0.58	20
xerox	10	203	19.35	7.12	65.92	65.42	0.5	36
hp	11	83	8.83	14.83	82.56	79.01	3.46	59
ami33	33	123	1.16	16.14	93.25	87.38	5.87	998
ami49	49	408	35.45	16.53	123.18	116.08	7.1	652

Mean power density $\mu = 1$ W/mm^2

Table 2 Comparison of HPWL and T_{max} for left compacted $W + T$ optimised fixed-outline floorplans (percentage whitespace $\omega = 20$ and a $\rho = 1$) of ami33 produced by HotFloorplan with the HPWL and T_{max} of the same floorplan after the whitespace reallocation

Mean power density (W/mm^2)	HotFloorplan outputs	CPSO										
		Mode-I (available %Whitespace = 20)						Mode-II (available %Whitespace = 16.21)				
	HPWL (mm)	T_{max} (°C)	HPWL (mm)	%diff in HPWL	T_{max} (°C)	ΔT_{max}	τ	HPWL (mm)	%diff HPWL	T_{max} (°C)	ΔT_{max}	τ
0.5	45.22	66.22	46.45	2.72	65.37	0.85	0.3	46.08	1.9	65.76	0.46	0.2
1	45.22	92.29	46.96	3.85	90.64	1.65	0.4	46.69	3.25	91.4	0.89	0.3
2	45.22	144.5	46.4	2.61	141.2	3.28	1.3	46.4	2.61	142.7	1.78	0.7
4	45.22	248.7	46.55	2.94	242.1	6.65	2.3	46.55	2.94	245.2	3.5	1.2

Table 3 Comparison of HPWL and T_{max} for left compacted $W + T$ optimised fixed-outline floorplans ($\omega = 20$ and $\rho = 1$) of ami49 produced by HotFloorplan with the HPWL and T_{max} of the same floorplan after the whitespace reallocation

Mean power density (W/MM^2)	HotFloorplan outputs	CPSO										
		Mode-I (available %Whitespace = 20)							Mode-II (available %Whitespace = 15.37)			
	HPWL (mm)	T_{max} (°C)	HPWL (mm)	%diff in HPWL	T_{max} (°C)	ΔT_{max}	τ	HPWL (mm)	%diff HPWL	T_{max} (°C)	ΔT_{max}	τ
0.5	1114	87.35	1139.81	2.32	83.11	4.24	1.83	1138.67	2.21	83.8	3.55	1.61
1	1114	134.4	1140.71	2.4	126.3	8.03	3.35	1140.77	2.4	127.8	6.51	2.71
2	1114	228.5	1137.19	2.08	212.8	15.8	7.58	1142.15	2.53	214.9	13.6	5.38
4	1114	416.9	1142.24	2.54	385.4	31.6	12.4	1136.49	2.02	391	26	12.9

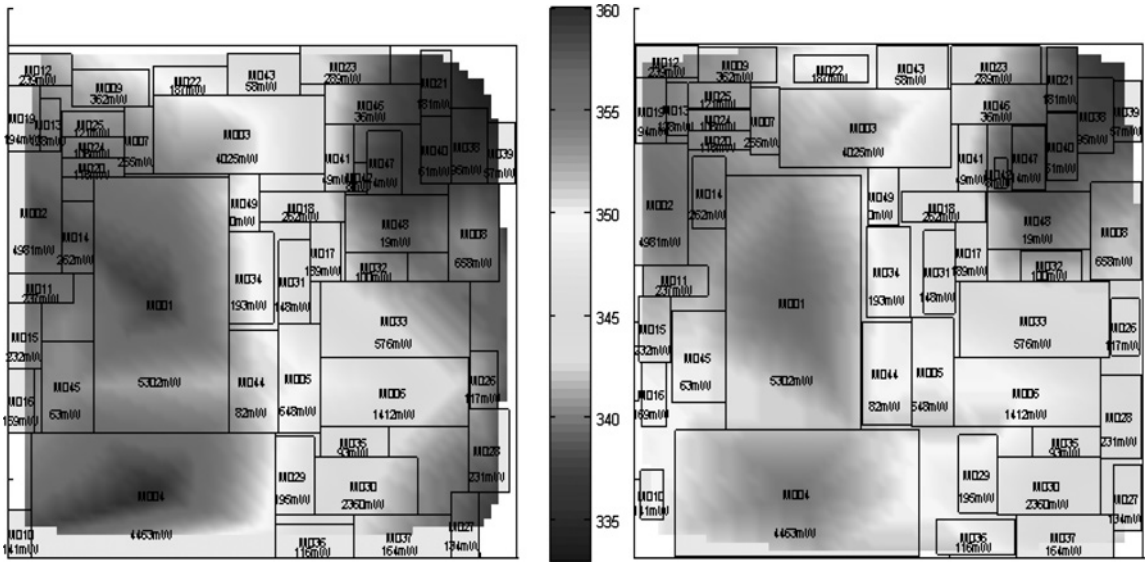

Figure 9 *Temperature profile of left compacted floorplan of ami49 (on the left) and the temperature profile of the same floorplan after whitespace reallocation using CPSO (on the right)*

IET Circuits Devices Syst., 2010, Vol. 4, Iss. 3, pp. 251–260

doi: 10.1049/iet-cds.2009.0049

ratio. In this case, $\Delta T = 4.24^{\circ}C$ for $\mu = 0.5\ W/mm^2$ and $\Delta T = 31.55^{\circ}C$ for $\mu = 4\ W/mm^2$, a 7.44× increment in ΔT for an 8× increment in μ. A similar linear trend is observed as before, indicating that reduction in T by this method becomes more effective in future technologies with higher power densities. In Mode-II, $\Delta T = 3.55^{\circ}C$ for $\mu = 0.5\ W/mm^2$ and $\Delta T = 25.99^{\circ}C$ for $\mu = 4\ W/mm^2$ which is much less than the corresponding ΔT value in Mode I, although the difference in whitespace availability is only 4.63%. We conclude that a small percentage of EW is capable of considerable reduction in T. Thus the trend towards hierarchical design using fixed outline floor planning can actually benefit from this method. The average increase in HPWL was 2.34% for fixed-outline floor planning and 2.29% for conventional floor planning of ami49. τ indicates the ratio of ΔT and ΔHPWL due to the whitespace reallocation. In all cases, increase in HWPL was ~2–3%, which is negligible unless it affects the critical nets. If the initial floor plan has critical nets that cannot afford minor net-length increment, then the CPSO procedure can also be carried out. In that case, all blocks connected to the critical net are pinned during particle position and velocity update.

The runtimes on a 1.8 GHz P4 with machine with1 GB RAM is reported in Table 1. Smaller runtime for ami49 as compared with ami33 is because the CPSO procedure was saturated in a lesser number of iteration and was terminated early. Fig. 9 shows thermal plots of ami49 before and after the whitespace reallocation. The hottest spot in the left-compacted floor plan (87.35°C) was at the centre of module M004. After the whitespace reallocation, the hottest spot shifted to the centre of the module M001 and the new hotspot temperature was reduced to 83.11°C.

7 Conclusion

In this paper, we present a robust method for thermal optimisation at a post floor planning stage. The method is capable of reducing the maximum on-chip temperature of floor plans that have already been optimised for temperature during floor planning. The method could also be applied for optimal placement of processors in a multi-core chip. The whitespace reallocation can reduce the chip temperature by few degrees to several tens of degrees depending on the availability of module slack, EW, power-density distribution and mode of floor planning. The efficacy of whitespace reallocation is expected to become more pronounced as the device size shrink, power density escalates and design complexity pushes floor planning to follow a hierarchical fixed-outline methodology.

8 Acknowledgment

The authors thank Dr Igor Markov, University of Michigan and Dr Subhashish Mitra, Stanford University for their helpful comments and suggestions.

9 References

[1] ADYA S.N., MARKOV I.L.: 'Fixed-outline floorplanning: enabling hierarchical design', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2003, **11**, (6), pp. 1120–1135

[2] VALENZUELA C.L., WANG P.Y.: 'VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions', *IEEE Trans. Evol. Comput.*, 2002, **6**, (4), pp. 390–401

[3] CHATTERJEE D., MANIKAS T.W.: 'Power-density aware floorplanning for reducing maximum on-chip temperature'. 18th IASTED Int. Conf. on Modelling and Simulation (ICMS), 2007, pp. 319–324

[4] MURATA H., FUJIYOSHI K., NAKATAKE S., KAJITANI Y.: 'VLSI module placement based on rectangle-packing by the sequence-pair', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1996, **15**, (12), pp. 1518–1524

[5] NAKATAKE S., FUJIYOSHI K., MURATA H., KAJITANI Y.: 'Module placement on BSG-structure and IC layout applications'. IEEE/ACM Int. Conf. on Computer-Aided Design, 1996, pp. 484–491

[6] GUO P.-N., CHENG C.-K., YOSHIMURA T.: 'An O-tree representation of non-slicing floorplan and its applications'. Proc. 36th ACM/IEEE Conf. on Design Automation Conf., 1999, pp. 268–273

[7] CHANG Y.-C., CHANG Y.-W., WU G.-M., WU S.-W.: 'B*-Trees: a new representation for non-slicing floorplans'. Proc. 37th Conf. on Design Automation, 2000, pp. 458–463

[8] 2007 International Technology Roadmap for Semiconductors: Process Integration, Devices, and Structures, http://www.itrs.net/

[9] MURALI S., MUTAPCIC A., ATIENZA D., GUPTA R., BOYD S., DE MICHELI G.: 'Temperature-aware processor frequency assignment for MPSoCs using convex optimization'. Int. Conf. on Hardware/Software Codesign and System Synthesis, 2007, pp. 111–116

[10] BORKAR S.: 'Design challenges of technology scaling', *IEEE Micro*, 1999, **19**, (4), pp. 23–29

[11] BROOKS D., MARTONOSI M.: 'Dynamic thermal management for high-performance microprocessors'. Proc. HPCA Seventh Int. Symp. on High-Performance Computer Architecture, 2001, pp. 171–182

[12] GUNTHER S.H., BINNS F., CARMEAN D.M., HALL J.C.: 'Managing the impact of increasing microprocessor power consumption', *Intel Technol. J.*, 2001, **5**, (1), pp. 1–9

[13] SKADRON K., STAN M.R., HUANG W., SIVAKUMAR V., KARTHIK S., TARJAN D.: 'Temperature-aware microarchitecture'. Proc. 30th Annual Int. Symp. on Computer Architecture, 2003, pp. 2–13

[14] HUNG W.L., XIE Y., VIJAYKRISHNAN N., ADDO-QUAYE C., THEOCHARIDES T., IRWIN M.J.: 'Thermal-aware floorplanning using genetic algorithms'. Proc. 6th Int. Symp. on Quality Electronic Design, 2005, pp. 634–639

[15] SANKARANARAYANAN K., VELUSAMY S., STAN M.R., SKADRON K.: 'A case for thermal-aware floorplanning at the microarchitectural level', *J. Instr.-Level Parallelism*, 2005, **8**, pp. 1–16

[16] YONGKUI H., ISRAEL K.: 'Simulated annealing based temperature aware floorplanning', *J. Low Power Electron.*, 2007, **3**, (2), pp. 141–155

[17] ADYA S.N., MARKOV I.L., VILLARRUBIA P.G.: 'On whitespace and stability in mixed-size placement and physical synthesis'. ICCAD-2003. Int. Conf. on Computer Aided Design, 2003, pp. 311–318

[18] TANG X., TIAN R., WONG M.D.F.: 'Minimizing wire length in floorplanning', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2006, **25**, (9), pp. 1744–1753

[19] ALPERT C.J., NAM G.-J., VILLARRUBIA P.G.: 'Free space management for cut-based placement'. IEEE/ACM Int. Conf. on Computer-Aided Design, 2002, pp. 746–751

[20] KAHNG A.B., TUCKER P., ZELIKOVSKY A.: 'Optimization of linear placements for wirelength minimization with free sites'. Proc. ASP-DAC '99 Asia and South Pacific Design Automation Conf., 1999, pp. 241–244

[21] OZISIK M.N.: 'Boundary value problems of heat condition' (Dover, New York, 1968)

[22] PRASITJUTRAKUL S., KUBITZ W.J.: 'Path-delay constrained floorplanning: a mathematical programming approach for initial placement'. Proc. 26th ACM/IEEE Design Automation Conf., 1989, pp. 364–369

[23] LALL B.S., ORTEGA A., KABIR H.: 'Thermal design rules for electronic components on conducting boards in passively cooled enclosures'. Proc. Intersociety Conf. on Thermal Phenomena in Electronic Systems, 1994, pp. 50–61

[24] EBERHART R., KENNEDY J.: 'A new optimizer using particle swarm theory'. MHS'95, Proc. Sixth Int. Symp. on Micro Machine and Human Science, 1995, pp. 39–43

doi: 10.1049/iet-cds.2009.0049