ESE 566

Design using Programmable Mixed-Signal Systems-on-Chip

Sharwari Ramesh Joshi (112950662)

Rahul Reddy Kondeti (112961400)

Vignesh Priyadarshan Nagarajan (112749011)

ESE 566: Hardware/Software Co-Design of Embedded Systems Application : Simulated Annealing

Problem Statement :

The goal of the project is to implement the partitioning algorithm using the scripting method that helps in reducing the execution period or improving the speed of the overall program. The main motivation of this project is to identify the components of the software part automatically using the scripting method programatically which includes the computations that is time consuming and to replace them with the customised hardware. The hardware components that is being built using the circuit reduces the time and the memory required to run the program. The process graph is identified and the hardware & software components are constructed using the random replacement, the so called simulated annealing. The processes are considered as nodes and the software & hardware connections are minimised which ultimately reduces the overall time taken for the processes.

Simulated Annealing :

The Simulated annealing is a method in which the optimisation is done stochastically based on the analogy between the annealing process of the solid matter and the optimisation task. Through this method the global optimisation solution is achieved and and the computational time is reduced significantly. The temperature is peaked initially and is slowly reduced in the lated stages in this simulated annealing process. In the extreme high temperatures the atoms/ molecules take a rapid movement making the atoms to take the different directions. In this annealing process the movement is made to slow down by lowering the energy levels of the atoms. The total energy of the matter is minimised and the placement of the individual atoms are fixed. This process can be simulated and applied to solve optimisation problems, especially tasks with very large number of variables.The clock cycles are very less compared to the other hardware circuits in the PSoC blocks. The simulated annealing algorithms finds the near best solution to partition the hardware and software blocks.

Application Description :

Chroma-key is a technique that a foreground object extracted from a foreground frame will be combined with a background frame to create a new composite frame for a special effect. The foreground frame has two parts consisting of foreground and background objects, in which the background object is a solid colour, usually green or blue. Chroma-key effect is often used in films, television programs, and in the weather forecasting, especially. For example in the weather reporter generally stands in front of a green or blue screen. Meanwhile, the green or blue background colour will be replaced with a weather map when the weather report is telecasting.

Code Organisation of chroma key:

The chroma key algorithm majorly functions using the depth of the image, assume that the background image in the first section is filled with the green colour, so it not that difficult to isolate the human out of the green background. The pixels that correspond to the green colour of the 2D matrix will be of same numerics, the average of the pixel values are taken to isolate the differed numerics where the red, blue and the green are taken into consideration. Initially, the input in the form of an image is fed to the program which converts to the integer array of the 3D matrix (red, blue and the green colour). After the average is being computed, the depth of the image is calculated and it is taken into the separate matrix/array. In order to find the threshold for an image there are two methods being proposed one defines in setting up of the manual threshold for the image and the other iterates through the image to find the optimal threshold.

Partitioning :

For the automation to be done for the Hardware & Software partitioning the cost function and the optimisation algorithm together plays a great factor in determining the efficient partitioning. Each computational part is treated as the different process, in which some of the sub processes are taken as the hardware and the rest of the sub processes are taken as the software. On taking a closer look, the sender and the receiver data of the processes are main factor in determining the delay in the communication between the nodes. So the partition has to be very efficient such the critical paths should be maximised so to minimise the interconnects, greater the value

of the critical path (K_i^P) lesser the communication between the software and the hardware, which ultimately leads us to the improved performance in the circuit. The cost and the performance comes hand in hand and the tradeoff occurs. For the better performance, the cost function seems to be high. Similarly if the cost needs to the low, the performance of the system will be impacted and expected to be low as well.

The process involves:

- Randomly altering the states.
- Using the objective function assessing the energy of the newly generated state.
- Deciding on whether to reject the new temperature or accept the new solution by the comparison of the energy of the previous states.
- Iterate and repeat the process until the converging answer is met.

For a move to be accepted, one of the two requirements must be met,

- The move gets impacted and leads to the reduced state energy
- The state energy is increased by the move but within the limits of the temperature (threshold level). As the algorithm progresses in the temperature is significantly reduced. By this way, we can avoid getting locked by the local minimum initially with the process but to start getting the possible solution by the end of the process.

Pseudocode / Algorithm Description :

- Start with random temperature initially and let's say *Temp* and the constant *alpha*.
- Generate and target for the random solution.
- Start scoring the solution with the hyper parameters (“neighbouring parameters”), it is termed as the *new_score*
- Comparison should be done for the new and the old score :
    - If *new_score* > *old_score*: switch to the other neighbouring solution
    - If *new_score* < *old_score* : possible movement to the neighbouring solution

- When the temperature gets decreased : *Temp = Temp * alpha (constant)*
- Iterate and repeat the process until the stopping criteria is met :
    - *Temp < minimumTemperature*
    - *numberOfIterations > maximumIterations*
    - *totalRunningTime > maximumRunningTime*
- Return the hyper parameters of the best cost solution along with the score.

The switching from old solution to the new solution is temperature dependent and it is very probabilistic. To be very clear, the comparison between the solutions is achieved by the computational capabilities of the the acceptance probability

a = exp((new_score - old_score)/Temp)

The "a" value is made to compare with the randomly generated results that ranges between [0, 1].

- If "a" > randomly generated number :
    - Move to the hyper parameters of the neighbouring solution

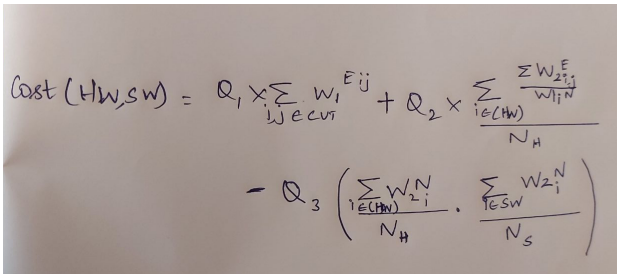
This literally means that if the "Temp" is large, almost all the new preferred solutions regardless of the total score. As *Temp* decreases, the likelihood of moving to hyper parameters resulting in a poor solution decreases

Data Structures :

Data structure	Purpose
Structure (Struct) :	• The structure is used here to define the node. The node contains several information like • NodeEdges • SoftwareNodes • HardwareNodes • NodeId
Vector (Integer list) :	Each lines of the nodes are taken in the Vector. The vector for the current node and the new node are compared and updated if the stopping criteria for the nodes are met.

* Data structures are majorly used in the cost function

The nodes are iterated and the cost function is computed. The computations are done using the



The frequent data transfers between the software and the hardware nodes are not advisable. Therefore in order to minimise the data transfer the cut between the software and the hardware should be efficient. Parallelism between the interconnects will also help us to achieve the time efficient communication between the HW & SW. Greater the critical path more parallelism is possible. Then the HW and SW are interchanged and the same process is repeated.

Implementation process of Simulated Annealing :

(i) The initial temperature is taken as a random values, the best case and the worst case values are determined by iterating the current node values and these values are being updated for each cost function call. The constant is set for the temperature calculation.

(ii) The current nodes are computed for each cost function and it is taken as the worst case.

(iii) The worst case cost function values are calculated for the each of the software nodes and hardware nodes which is in the form of the vector defining the form of a list.

(iv) The iteration is performed till the last process (total length of the software nodes) the worst case and best case are printed.

(v) The stopping criteria is estimated (initially taken as random value), this time the iteration is performed till the length of the temperatures data length (T_L). For each iteration the random number is generated using the rand() function.

(vi) The new cost is computed using the Cost Function for each iteration through which the change in the cost function is noted.

(vii) The computed new cost is compared with the best cost and the best cost values are updated on each loop and the cost difference is determined.

(viii) The cost difference is verified as positive and negative and on each positive values of cost difference, the hardware nodes and the software nodes are updated.

Functions and Functionalities :

(i) readInput() - This function is used to read the lines of the cost function file that is used to differentiate the software and the hardware nodes

(ii) printNodes() - This function is used to display the software nodes and the hardware nodes before the partitioning

(iii) estimateCostFunction() - This helps in finding the cost function which later is useful for the optimisation algorithm (i.e.) Simulated Annealing

(iv) simulatedAnnealing() - This function helps in optimising the partitioning process through which the best and the worst costs are determined

Implementation issues :

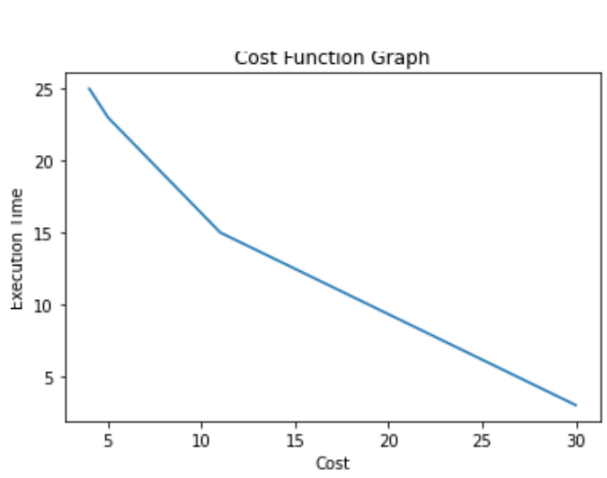
Owing to the difficulty in processing with the input file, the test file (source.txt) is auto-generated with some readable values for which the cost function and the optimisation algorithm is being performed. This leads us to face the challenge for determining the best and the worst cost for the software and the hardware partitioning. The critical path is roughly estimated and the cut is calculated accordingly. Having said this, the cost function of the project2 is compared with the automated scripting partitioning. The manual optimisation will be helpful for the smaller computation whereas for most of the realtime scenarios the embedded application uses the automated process like the simulated annealing or the Tabu search algorithm.

Results and Output:

The cost function using the automated optimisation through the scripting is compared with the manual computation cost function in which the results are similar to each other. The local minima in the cost function graph is identified virtually and is marked as the significant spot on the results. The gradient descendent through the plot gives us the idea about which part of the curve seems to the best cost with the help of the local minima.

Cost Function graph :

Manual optimisation (Project-2) ***Automated/Scripted optimisation***



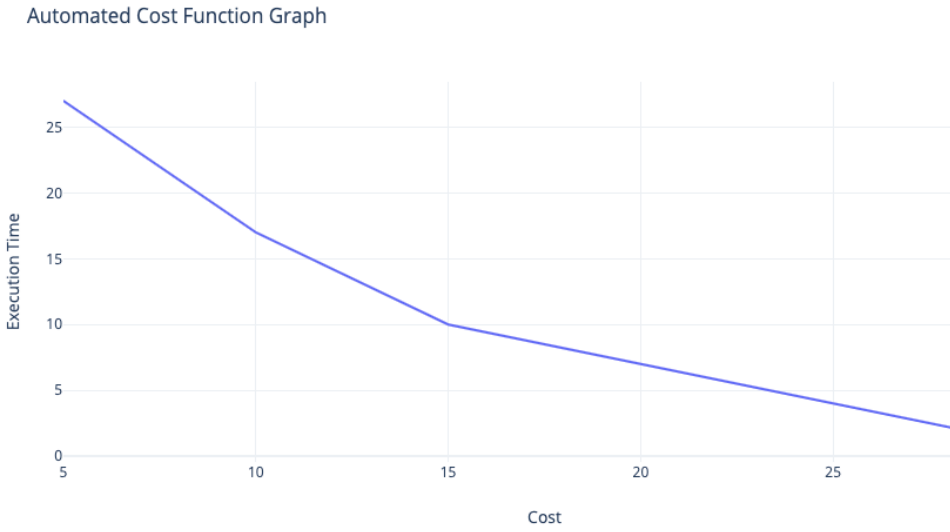


Image 1 : Cost function graph obtained using the manually optimised - Project 2 result.

Image 2 : This is cost function graph obtained using the scripting technique. For the plotting purpose matPlotLib were used in the python file. Rest of the processes deals with the CPP and the Perl scripting language.

Comparison : When comparing the current cost function variation/plot with that of the manually optimised cost function graph, the result seems very much similar and the local minimum can be seen > 25. The gradient descendent seems to be slightly varied at the point 10 to 15 but the results achieved slightly correlate with each other

Conclusion :

Though the cost function graph could be achieved using the automation by scripting, owing to the difficulty with the input file that should be fed to the perl script, the simulated annealing algorithm didn't turn out as expected. I claim here that the simulated annealing algorithm implemented will work fine if the input to the script is properly given, the software and the hardware nodes that has been separated using the algorithm is outputted through print statement which the working is verified. The nodes that's been cut tells us that the critical path is lesser where the parallelism were achieved.

Learning outcome :

The manual optimisation seems effective for the smaller circuits but for the larger implementation and for the real world scenario this simulated annealing algorithm for partitioning will gives much insight to achieve the partitioning very effectively. Manual computations for the real world applications is time consuming that it may result with some errors as well. This project taught us how the automated mechanism helps in dealing with the real world application much faster and efficiently.

References :

(i) https://ieeexplore.ieee.org/document/1346809

(ii) https://www.researchgate.net/publication/4099873_VLSI_circuit_partition_using_simulated_annealing_algorithm

(iii) http://www.wseas.us/e-library/conferences/2010/Bucharest/CI/CI-10.pdf

(iv) https://www.springer.com/gp/book/9780898382815