A Particle Swarm Optimization Approach for Synthesizing Application-specific Hybrid Photonic Networks-on-Chip

Shirish Bahirat and Sudeep Pasricha
ECE Department, Colorado State University, Fort Collins, CO
{shirish.bahirat, sudeep}@colostate.edu

Abstract- Hybrid nanophotonic-electric networks-on-chip (NoC) have been recently proposed to overcome the challenges of significant power dissipation and high data transfer latencies in traditional electrical NoCs. However, with increasing embedded application complexity, hardware dependencies, and performance variability, optimizing hybrid nanophotonic-electric NoCs requires traversing a massive design space. No prior work has addressed this problem to the best of our knowledge. We present the first effort in this direction. In this paper, we propose an evolutionary algorithm framework to synthesize hybrid photonic NoCs by utilizing Particle Swarm Optimization (PSO) and Simulated Annealing (SA). Our synthesis results indicate that the PSO-based framework generates more power efficient NoC fabrics compared to SA. The PSO synthesis approach optimizes the hybrid NoC fabric, achieving up to 18× power reduction compared to the traditional electrical NoC demonstrating significant promise towards generating low power communication fabrics that meet performance objectives.

1. Introduction

According to the ITRS roadmap [1], reducing energy consumption has become an increasingly important goal for semiconductor electronics. Future chip multiprocessors (CMPs) will need to support this vision with innovations to meet aggressive targets for energy efficiency and performance. In order to satisfy power and performance challenges, these future CMPs will inevitably integrate multiple cores (as many as 256 within the next five years based on projections) onto a single die to optimize processing power-per-watt. With the advent of highly parallel CMPs, the network-on-chip (NoC) interconnection fabric [2] will be the limiting factor for achieving operational goals comprised of stringent bandwidth and latency constraints. NoCs will play a crucial role to ensure reliable and scalable connectivity between processing cores, memory modules, cache banks, and other I/O devices. Unfortunately, traditional electrical-based NoCs are severely constrained due to their long multi-hop latencies and high power consumption, making it practically impossible to stay within the on-chip power budget. CMOS compatible on-chip photonic interconnects with silicon-on-insulator waveguides provide a potential substitute for electrical interconnects, particularly for global communication, allowing data to be transferred across a chip with much faster light signals. Based on recent technology advancements, critical length at which photonic interconnects are advantageous over electrical interconnects has fallen well below expected chip die size dimensions [3].

To maximize power savings, recent research [4]-[10] has been focused on novel hybrid photonic NoC architectures that optimize local and global communication distribution between electrical and photonic communication channels. However, automated optimization of hybrid photonic NoCs for application specific performance is an NP-hard problem that has not been addressed to date. A photonic interconnect with $n=256$ waveguides and $m=256$ wavelengths will require $n+(n)^2+(n)^3+\ldots+(n)^m$ configurations if performance with every combination is to be analyzed. This is most certainly practically exorbitant. Moreover, analyzing these two parameters is just a small part of the much larger set of parameters that must be explored during optimization. Finding an optimum solution for such a combinatorial optimization problem could take years if we have to search through the entire solution space.

The best way to solve this problem is by developing heuristics that permit us to identify and search through a relevant portion of the solution space in a tractable amount of time to find a near optimal solution. Greedy heuristics are unlikely to find good quality solutions due to their inclination for getting stuck in local optima instead of global optima. In contrast, evolutionary algorithms such as simulated annealing (SA) [11] operate through repeated transformations and have the hill-climbing ability to escape local optima by allowing acceptance of worse solutions within the evaluation process. A population of solutions being simultaneously manipulated is one of the major differences between the SA and traditional greedy search algorithms. Evolutionary algorithms have proven highly effective in recent years for several hard problems in the realm of VLSI physical design, such as partitioning, placement, and routing [16].

In this paper, for the first time, we address the problem of synthesizing application-specific hybrid photonic NoCs by using a class of evolutionary algorithm called Particle Swarm Optimization (PSO) [17]. To compare the performance of our near optimal synthesized solution, we also developed a novel formulation of the problem using a SA algorithm. By addressing the hybrid photonic NoC synthesis problem using evolutionary algorithms, we are able to effectively explore interesting trade-offs in the design space within a reasonable amount of time to generate a power optimized solution while meeting application latency requirements. Our experimental studies demonstrate that the PSO-based approach can be a natural fit for hybrid photonic NoC synthesis problems, outperforming the SA-based approach.

2. Related Work

The on-chip photonic interconnection concept was initially introduced by Goodman et al. [18]. In recent years, several works [19]-[21] have presented a comparison of physical and circuit-level properties of on-chip electrical and photonic interconnects based on advances in the fabrication and integration of photonic elements on CMOS chips. A few recent works have presented hybrid photonic NoC architectures. Pan et al. [4] proposed a hierarchical photonic crossbar coupled with a concentrated mesh electrical network. Shacham et al. [5] described a circuit-switched on-chip photonic torus network with reconfigurable broadband optical switches coupled with a topologically identical torus electrical network. Bahirat et al. [6] presented concentric photonic rings coupled with a reconfigurable electrical mesh. Li et al. [7] described a photonic broadcast bus with an electrical packet switched network. Vantrease et al. [8] proposed a fully photonic crossbar with electrical buffering in the electrical realm. Joshi et al. [9] presented a similar fully photonic topology but using a Clos network and electrical buffering. Morris et al. [10] created a hybrid of an all-photonic crossbar

and a photonic fat-tree. We choose the hybrid photonic ring/electrical mesh topology [6] as the baseline fabric for our synthesis effort due to its favorable low power, latency characteristics, and configurability.

Efforts on regular NoC topology synthesis have primarily focused on mapping uniform sized cores and their communication flows on regular mesh topologies [21]-[25] to optimize energy and performance. For example, Ascia et al. [23] utilized a genetic algorithm approach to map cores to minimize communication power within a mesh NoC. Kapadia et al. [25] proposed a heuristic approach to synthesize a voltage island-aware low-power mapping of cores and communication routes on a regular mesh NoC. Other efforts have focused on application-specific synthesis of irregular NoCs [26]-[29]. For instance, Murali et al. [26] presented a synthesis technique with min-cut partitioning to allocate switches to groups of custom cores and minimize NoC power consumption. Srinivasan et al [27] presented a genetic algorithm based approach to synthesize a low power custom NoC topology. None of these synthesis approaches focus on hybrid photonic NoCs, as we do.

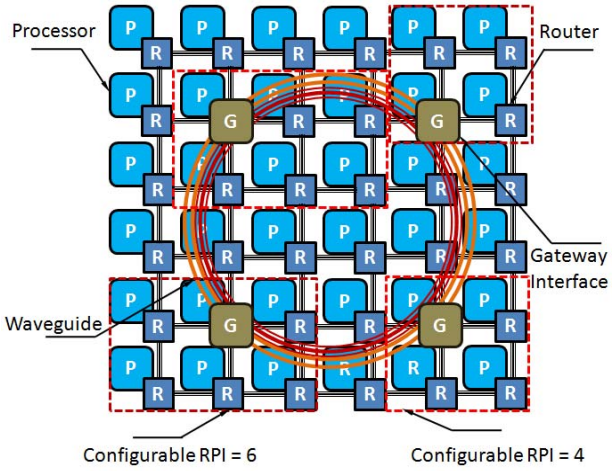


Figure 1: Hybrid ring-mesh photonic architecture [6]

3. Hybrid Photonic NoC Architecture Overview

We consider the ring-mesh hybrid photonic topology presented in [6] as our baseline architecture. We summarize some of its key features in this section. Figure 1 presents an architectural overview which consists of concentric photonic ring waveguides on a dedicated photonic layer, interfacing a 2D electrical mesh NoC. The electrical mesh is comprised of two types of routers: *(i)* conventional four stage pipelined electrical mesh routers that have 5 I/O ports (N, S, E, W, local core) with the exception of the boundary routers that have fewer I/O ports, and *(ii)* gateway interface routers that are also four-stage pipelined but have six I/O ports (N, S, E, W, local core, photonic link interface) and are responsible for sending/receiving flits to/from photonic interconnects in the photonic layer. Both types of routers have an input and output queued crossbar with a 4-flit buffer on each input/output port, with the exception of the photonic ports in gateway interface routers that use double buffering to more effectively cope with the higher photonic path throughput.

A unique feature of this hybrid photonic architecture is the reconfigurable traffic partitioning between the electrical and photonic links. To minimize implementation cost, the number of gateway interfaces are kept lower. However, limiting gateway interfaces reduce photonic path utilization as CMP size increases. A programmable photonic region of influence (PRI) is used to ensure appropriate scaling and utilization, which refers to the number of cores around the gateway interface that can utilize the photonic path for communication. A modified PRI-aware XY routing scheme routes packets in this architecture. Thus for larger CMPs, having a larger PRI size can ensure appropriate photonic path utilization. Figure 1 shows a 6×6 CMP with varying PRI sizes at four gateway interfaces. Communicating cores lying within the same PRI communicate using the electrical NoC (intra-PRI transfers). Cores that need to communicate and reside in different PRIs communicate using the photonic paths (inter-PRI transfers), provided they satisfy two criteria: *(i)* the size of data to be transferred is above a user-defined size threshold M_{th}, and *(ii)* the number of hops from the source core to its local PRI gateway interface is less than the number of hops to its destination core.

The concentric ring photonic waveguides are logically partitioned into four channels: *reservation*, *reservation acknowledge*, *data*, and *data acknowledge*. A fully photonic path setup and acknowledgement mechanism is used within the reservation and acknowledge channels, utilizing a Single Writer Multiple Reader (SWMR) configuration. The data channel utilizes a low cost Multiple Writer Multiple Reader (MWMR) configuration. High throughput is achieved by using dense wavelength division multiplexing (DWDM), with multiple wavelengths per waveguide available to transfer multiple streams of concurrent data. To reduce the number of photonic components (waveguides, buffers, transmitters/receivers), and consequently reduce area and power dissipation in the photonic layer, the concept of serialization is also utilized at the gateway interfaces.

TABLE I SYNTHESIS PARAMETERS

Synthesis Parameters	*Range low*	*Range high*
Uplinks	4	32
PRI	1	(num_cores)/4
WDM	32	256
Serialization Degree	1	32
Clock Frequency (GHz)	1	5
PRI data size threshold (Mth)	4	1024
Flit Width (bytes)	4	8
Waveguides	32	256

4. Synthesis Problem Formulation

Our synthesis problem includes the following inputs: *(i)* A core graph $G(V, E)$ representing the cores on which tasks have already been mapped, and the set of M edges $\{e_1, e_2, e_3, \ldots, e_M\}$ and N vertices $\{V_1, V_2, V_3, \ldots, V_N\}$ with weights that represent latency constraints between communicating cores. *(ii)* A regular mesh-based CMP with T tiles such that $T = (d^2)$, where d is the dimension of the mesh, and each tile consists of a compute core and a NoC router, *(iii)* The upper and lower bounds for a set of parameters relevant to hybrid photonic NoCs (Table I).

Our goal is to synthesize a hybrid photonic-ring/electrical-mesh NoC architecture for a specific application that will determine *(i)* number and location of photonic uplinks (i.e., gateway interfaces), *(ii)* PRI sizes, *(iii)* density of wavelength division multiplexing, *(iv)* vertical serialization degree, *(v)* link clock frequency, *(vi)* data threshold size, *(vii)* flit widths, and *(viii)* number of photonic waveguides; while satisfying communication latency constraints and minimizing communication power.

5. Synthesis Framework Overview

In this section, we present our novel Particle Swarm Optimization (PSO) framework for synthesizing hybrid photonic NoCs. Subsequently, we also describe a novel Simulated Annealing (SA) formulation for the same problem.

5.1 Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) metaheuristic was initially proposed by R. Eberhart and J. Kenned [17] in 1995. The fundamental idea behind PSO is inspired by the coordinated and collective social behavior of species like a flock of birds, fish, termites, or even humans. In nature, each individual bird, bee, or fish shares some information with its neighbors and by utilizing shared information collectively, they strive to organize efforts such as collect food or develop flying patterns to minimize aerodynamic drag, etc. Although by itself, a single entity such as a bird or a bee is a simple and unsophisticated creature, collectively as part of a swarm they can perform useful tasks such as building nests, and foraging (searching for food). Within PSO, an individual entity is called a particle and it shares information with other entities, either in the form of direct or indirect communication to coordinate their problem-solving activities. In recent years, the PSO algorithm has been applied to many combinatorial optimization problems such as optimal placement of wavelength converters in WDM networks [30] and dynamic reconfiguration of field-programmable analog circuits [31].

To implement the PSO algorithm, particles are placed in the search space of some problem, and each particle evaluates the objective function at its current location to determine its movement by combining the best (best-fitness) locations in the vicinity. The next iteration takes place after all particles are relocated to the new position. This process is repeated for all particles and eventually for the swarm as a whole; similar to the flock of birds collectively foraging for food. A particle on its own does not have power to solve the problem; evolution occurs only when the particles interact and work together, utilizing a social network consists of bidirectional communication. The movement of each particle is affected by its inertia or own weight and directional velocity towards local and global best solutions. As the algorithm iterates, particles move towards local as well as global solution optima forming a swarm pattern. For example, when one particle or entity finds a good solution such as a food source, other particles are more likely to be attracted by following a positional path. This social interaction feedback eventually causes all particles to move towards a globally optimal solution path.

In summary, the idea of the PSO is to mimic the social collective behavior found in nature and utilizing it to solve complex problems. As the flits are routed from the source to the destination in a NoC, the PSO process can select among various values for the parameters in Table I, such as number of WDM channels, or PRI size. As more core communications are considered, gradually one dominant solution emerges. This best configuration satisfies application-specific latency constraints and has the lowest power paths that all particles follow with a specific PRI size, WDM, link clock frequency etc. The particles search or move in the solution space by gravitating towards optimality based on the neighborhood and global particle fitness. This transversal phenomenon is similar to the social interactions where a leader or leaders emerge from the swarm and followers attempt to follow them. A group of random particles or random solutions are initialized during the initial phase of PSO and then the optimal or near optimal solution is constructed using an iterative process. Within an iteration, a particle tracks the personal best solution (p_k), which is the best solution found by the particle k, and the global best solution (g_k), which is the best solution that was found by the entire population. Every particle moves towards the better solutions with some velocity and position. The computation step includes some amount of randomness instead of following an exact profile. This randomness can produce an even better solution which may result in attracting other particle towards it. Each particle updates its velocity and position based on the following set of equations:

$$v_{k+1} = wv_k + c_1 r_1 (p_k - x_k) + c_2 r_2 (g_k - x_k) \quad (1)$$

$$x_{k+1} = x_k + v_{k+1} \quad (2)$$

where, the current velocity and position for each particle is defined by v_k and x_k respectively, p_k designates current best solution based on particle k's history and g_k defines current best solution based on the entire population or swarm. Positive inertial weight w is assigned to control how fast or slow each particle can move based on its own weight or inertia, c_1 and c_2 are constant numbers denoted as learning parameters that control the learning rate of global vs. local optima, i.e., higher the weights, the faster the particles gravitate towards the current best solution. Instead of just following the current best solution in a linear path, r_1 and r_2 are random numbers from 0 to 1 that change every iteration, adding randomness to the path thus, finding newer better solutions on the way. The stability of the PSO algorithm is one of the key concerns where position and velocity can diverge instead of achieving convergence. To ensure convergence we tune the learning and inertial weights carefully and also implement a velocity limit parameter V_{max}, where if the updated velocity exceeds the velocity limit, we saturated the velocity to the max limit.

```
done = 1
while(done)
    for (i=0; i++; i< N_ITER)
      InitializeParticles();    // generate m particles
      UpdateParticleSystem();  // update particle system for local
                                  and global best solution, move
                                  the particles to new position
      UpdatePositionMatrix();// position update for each particle
    end for // end of the loop
ValidateResults();//cycle-accurate simulations for validation
    if Results Validated then
        done = 0
    else
        done = 1 //Continue with next PSO iteration
    end if
end while
```

Figure 2: Particle swarm optimization formulation

Figure 2 shows the pseudo-code for our PSO formulation. The algorithm starts by initializing each particle when it calls *InitializeParticles()*. The function initializes inertia and learning weights for each parameter such as WDM. For the PRI, it also generates source and destination locations of the communication neighborhoods. The function *UpdateParticleSystem()* iterates and updates velocity and positions of the individual particles, using relations (1) and (2). At the end of the evaluation loop, particle positions are updated and moved to new positions by calling *UpdatePositionMatrix()*.This process continues for the entire application, until a dominating solution emerges. We ultimately call *ValidateResults()* to validate the best solution by using cycle-accurate SystemC[36] level simulation. This is done to ensure that latency constraints are satisfied in the presence of communication congestion and computation delays (hard to ascertain with just an analytical approach). If the solution fails simulation-based validation we repeat the PSO algorithm with different trail parameter and selection probability values.

5.2 Simulated Annealing (SA)

Simulated Annealing (SA) algorithms [12]-[14] are evolutionary algorithms that generate solutions to optimization problems using techniques inspired by annealing in solids. SA algorithms simulate the cooling of a metal in the heat bath known as annealing where the structural properties depend on

the cooling rate. When a metal is hot and in liquid state, if cooled in a controlled fashion, large and consistent grains can be formed. On the other hand, grains can contain imperfections if the liquid is cooled rapidly or quenched. By slowly lowering the temperature, globally optimal solutions are approached asymptotically by allowing hill climbing or worse moves (inferior quality) to be taken within the initial part of the iteration process. Based on the law of thermodynamics, at temperature t the probability of an increase in energy of magnitude δE is given by:

$$P(\delta E) = e^{(-\frac{\delta E}{kt})} \quad (3)$$

where k is the Boltzmann's constant. This equation is directly applied to SA by dropping the Boltzmann constant which was only introduced into the equation to cope with different materials. The probability of accepting a state in SA is:

$$P = e^{(-\frac{c}{t})} < r \quad (4)$$

where c defines the change in evaluation function output, t defies current temperature which is decremented at every iteration by some regression algorithm such as a linear method $t_{(k+1)}=\alpha.t_{(k)}$, with $\alpha < 1$, and r is a random number between 0 and 1.

```
done = 1
while(done)
    for (i=0; i++; i< N_ITER)
      GenerateInitialSolution()   // Initial solution
      ScheduleCoolingRate()   //Evaluate solution at cooling rate
      ComputeFitnessValue()  // Update fitness value
    end for
  ValidateResults()//cycle-accurate simulations for validation
    if Results Validated then
        done = 0
    else
        done = 1 //Continue next N_ITER generations
    end if
end while
```

Figure 3: Simulated annealing algorithm formulation

An SA algorithm involves the evolution of an individual solution over a number of iterations, with a fitness value used for evaluating solution quality whose determination is problem dependent. At each iteration, individual parameters are selected randomly and the probability of accepting a solution is determined by equation *(4)*. A high enough starting temperature is selected to allow movement through the entire search space. As the algorithm progresses, the temperature is cooled down to confine solutions, allowing better solutions to be accepted until the final temperature is reached. As SAs are heuristics, the solution found is not always guaranteed to be the optimal solution. However in practice, SA has been used successfully to generate fairly high quality solutions in several problem domains.

Figure 3 shows the pseudo-code for our SA formulation of the synthesis problem. Our SA implementation begins with the calling of *GenerateInitialSolution()* to generate an initial solution. We utilize a genetic algorithm (GA) as an initial heuristic to ensure high quality for the SA seed. Each GA chromosome consists of constituent parameters as defined in Table 1. At the end of 200 generations we use the best solution with the highest fitness value as a seed for SA. Subsequently, four key parameters for annealing are initialized by calling *ScheduleCoolingRate()*: *(i)* Starting temperature, *(ii)* Temperature decrement, *(iii)* Final temperature, and *(iv)* Iterations at each temperature. We tuned the starting temperature hot enough to allow our hybrid NoC parameters to traverse farther along in the solution space. Without this consideration the final solution would be very close to the starting SA solution. Based on the number of iterations for which the algorithm will be running, the temperature needs to be decremented such that it will eventually arrive at the stopping criterion. We also need to allow enough iterations at each temperature such that the system stabilizes at that temperature. We evaluated another method first suggested in [15] that proposes implementing one iteration at each temperature by decreasing the temperature very slowly. The formula we used was $t_{(k+1)}= t_{(k)}/(1+\beta t_{(k+1)})$ where β is a suitably small value as defined in [15]. However the approach did not yield any benefits in terms of improvement in results. As SA is a stochastic search algorithm, it is difficult to formally specify convergence criteria based on optimality. The results are expected to get better with every step, however sometimes the fitness of a solution, calculated by calling *ComputeFitnessValue(),* may remain unchanged for a number of cooling steps before any superior solution can be created. The best solution is ultimately validated with an in-house SystemC-based[36] cycle-accurate hybrid nanophotonic-electric NoC simulation by calling the function *ValidateResults()* to ensure that latency constraints are satisfied under realistic traffic conditions. If the solution fails validation we repeat the SA algorithm with a different initial seed.

TABLE II Delay and energy consumption for photonic elements (32nm)
DDE = Data traffic dependent energy, SE = Static energy (clock, leakage), TTE = Thermal tuning energy (20K temperature range)

Component	Delay	DDE	SE	TTE
Modulator driver	9.5 ps	20 fJ/bit	5 fJ/bit	16 fJ/bit/heater
Modulator	3.1 ps			
Waveguide	15.4 ps/mm	-	-	-
Photo Detector	0.22 ps	20 fJ/bit	5 fJ/bit	16 fJ/bit/heater
Receiver	4.0 ps			

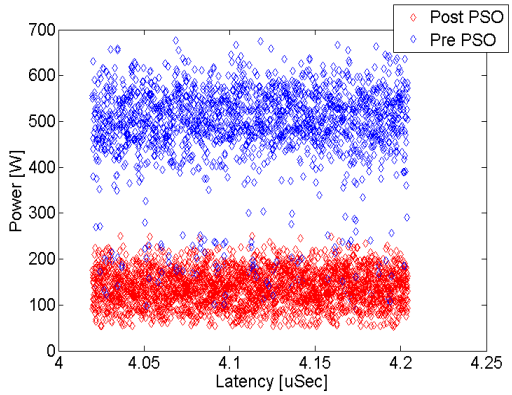


Figure 4: Pre and post PSO power consumption and latency for solution space of *lu* benchmark

6. Experiments

6.1 Experimental Setup

We conducted an experimental analysis to compare the performance of our PSO and SA based synthesis frameworks for mid size 6×6 (36-core) and large size 10×10 (100-core) CMPs with a 2D mesh hybrid photonic ring/mesh NoC fabric. The parallel implementation of seven SPLASH-2 benchmarks [32] (*barnes, lu, cholesky, fft, fmm, radiosity, radix*) were utilized to guide the application-specific synthesis in each of the two frameworks. We targeted a 32nm process technology with the assumption of a 400 mm^2 active die area budget. Table II shows delays for the 32 nm node that we assumed, obtained from [3] and from device fabrication results [33]. The delay of an optimally repeated and sized copper wire at 32nm was

assumed to be 42ps/mm [34]. The power dissipated in the hybrid photonic NoC can be categorized into *(i)* electrical network power and *(ii)* photonic ring network power. The static and dynamic power dissipation of electrical routers and links was derived from Orion 2.0 [35]. For the energy dissipation of the modulator driver and TIA power we used ITRS device projections [1] and standard circuit procedures. Energy dissipation values for the modulators and receivers are summarized in Table II [9]. In addition, an off-chip electrical laser power of 3.3W (with 30% laser efficiency) is also considered in our energy calculations. The laser power value accounts for per component optical losses for non-linearity (1dB at 30mW), coupler/splitter (1.2dB), waveguide (3dB/cm), waveguide crossings (0.05dB), ring modulator (1dB), receiver filter (1.5dB) and photodetector (0.1 dB).

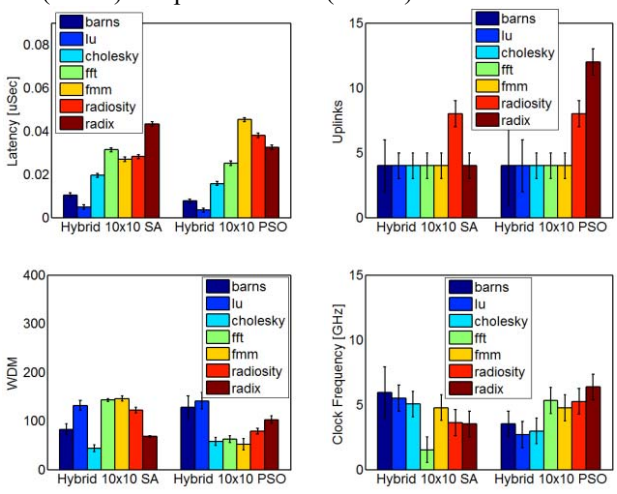


Figure 5: SA and PSO performance comparison for parameter selection

6.2 Results

Our first experiment provides insights into the workings of the PSO algorithm. Figure 4 shows the solution space pre- and post-PSO, and compares the power and average packet latency of the solutions, for the *lu* benchmark from the SPLASH-2 suite. The solution space is relatively randomly distributed in 2D with higher power consumption before the PSO algorithm begins execution. However, the solutions swarm towards the lower end of the 2D space by following velocity profiles relative to local and global optima's after the PSO algorithm has finished. This indicates an improvement in power while maintaining average packet latency characteristics, and shows that the PSO approach leads to desirable quality solutions.

To gain further insight regarding the quality of the generated results, we evaluated various parameters of the best solutions as shown in Figure 5 generated by PSO and SA for the seven SPLASH-2 benchmarks, and for the 10×10 CMP implementation. For benchmarks that require more frequent local and global communication, PSO utilizes a higher degree of photonic path communications while enabling higher clock frequency for the electrical path and achieving an elegant trade off. Also to enable higher photonic path communication, the PSO algorithm generates solutions with greater number of uplinks. The latency and WDM degree results indicate that PSO and SA each have a unique set of benchmarks for which the synthesized architecture provides lower average packet latency and lower WDM than the solution generated by the other approach. As far as the number of uplinks are concerned, with the exception of *radix* and *radiosity*, PSO and SA both select the same number of links for their best solutions.

TABLE III SA SYNTHESIS RESULTS

Synthesis Parameters	*barns*	*Lu*	*cholesky*	*fft*	*fmm*	*radiosity*	*Radix*
WDM	102	79	58	128	93	141	63
Uplinks	4	4	4	4	4	4	4
PRI	4	4	4	4	4	4	4
PRI Data Threshold	176	16	60	96	280	459	168
Clock Frequency	4	2	4	5	6	5	4
Source PRI Uplink	4	4	4	4	5	5	4
Dest PRI Uplinks	5	5	4	4	5	5	4
Flit Width	23	64	85	43	26	15	37
Serialization	11	4	3	6	10	17	7
Waveguides	48	2	25	200	105	128	90

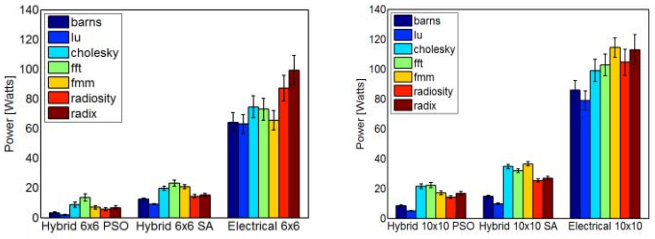
TABLE IV PSO SYNTHESIS RESULTS

Synthesis Parameters	*barns*	*Lu*	*cholesky*	*fft*	*fmm*	*radiosity*	*radix*
WDM	68	122	44	83	135	132	143
Uplinks	4	4	4	4	4	8	12
PRI	15	12	10	10	12	12	12
PRI Data Threshold	96	4	120	48	7	54	96
Clock Frequency	5	4	5	4	4	2	2
Source PRI Uplink	9	7	8	9	8	9	9
Dest PRI Uplinks	9	9	9	9	9	9	9
Flit Width	43	256	28	85	256	128	128
Serialization	6	1	9	3	1	2	2
Waveguides	18	12	135	136	112	20	18

Tables III and IV summarize the hybrid photonic NoC solutions generated by SA and PSO algorithms respectively for the SPLASH-2 benchmark applications. The communication traffic pattern for each application is different, so these results provide insights into the inner workings of each synthesis approach. As the communication traffic goes up, both algorithms tend to adapt differently towards solution configurations. Runtime configuration can enable custom solutions that balance various trade-offs, for example PSO adapts more efficiently to higher PRI size for *radix* and *radiosity* than SA. Both algorithms successfully increase WDM degrees as core to core communication increases. PSO provides higher number of uplinks compared to SA as the amount of global communication increases. The PRI data threshold M_{th}, diverts communication through photonic channels if data length exceeds beyond this limit. The PSO algorithm optimizes the M_{th} limit to a lower number than SA thus increasing the volume of data traversing the photonic communication path. We also monitored number of average hops from the source and destination cores to the uplinks within a PRI region to better understand how far data packets needed to travel to reach an uplink. The number of PSO generated hops was higher than SA confirming the consistency with higher PRI size achieved by PSO, particularly for the *radiosity* and *radix* benchmarks.

Figure 6 shows the most power efficient solutions (that satisfy latency constraints) generated by PSO and SA for the 36 core and 100 core configurations. The figure also includes power dissipation of the baseline electrical 2D mesh NoC as a reference. Despite the overhead of a separate photonic layer, it can be seen that using hybrid photonic NoCs can lead to significant orders of magnitude savings in power dissipation. Among the synthesized approaches, it can be seen that the PSO generates solutions that are more power efficient than those generated by the SA. The synthesized solutions with PSO have as much as 1.2× lower power dissipation than average solutions

generated by SA with up to 2.2× for the best case. Figure 7 shows the energy-delay product improvements for PSO and SA over the baseline electrical 2D mesh NoC for the 36 and 100 core CMPs. Again, significant (up to 18×) improvements can be seen with PSO generated hybrid photonic NoC solutions compared to the baseline 2D electrical mesh NoC architecture, as well as the SA generated solutions.



(a) (b)

Figure 6: Power consumption comparison for PSO and SA synthesized solutions with electrical NoC (a) 6x6 NoC (b) 10x10 NoC

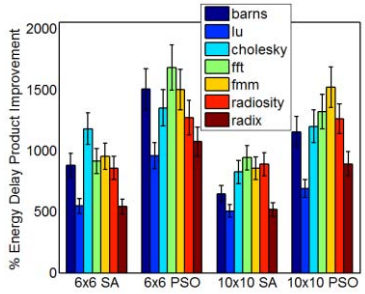


Figure 7: Energy delay product improvements for solutions generated by SA and PSO over electrical NoC

7. Conclusion

In this paper, we proposed an approach for synthesizing hybrid photonic NoC architectures for emerging CMPs. We formulate the synthesis problem as a Particle Swarm Optimization (PSO) problem, and contrast it with another novel formulation using Simulated Annealing (SA). Our results and experimental data demonstrate significant promise for the PSO-based synthesis framework, to determine application-specific architectural parameters that minimize power dissipation while satisfying application latency constraints. Our future work will further explore novel synthesis techniques for other hybrid photonic NoCs.

References

[1] ITRS Technology Working Groups, http://public.itrs.net. International Technology Roadmap for Semiconductors (ITRS) 2007.

[2] S. Pasricha, and N. Dutt. “On-Chip Communication Architectures”, Morgan Kauffman, ISBN 978-0-12-373892-9, Apr 2008

[3] G. Chen, et al., “Predictions of CMOS Compatible On-Chip Optical Interconnect,” Proc. of SLIP, pp. 13-20, 2005.

[4] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A.Choudhary, “Firefly: Illuminating future network-on-chip with nanophotonics,” Proc. ISCA, pp. 429–440, 2009.

[5] A. Shacham, K. Bergman, and L.P. Carloni, “The Case for Low-Power Photonic Networks on Chip,” Proc. DAC, pp. 132-135, 2007.

[6] S. Bahirat, S. Pasricha, "Exploring Hybrid Photonic Networks-on-Chip for Emerging Chip Multiprocessors", Proc. CODES+ISSS, 2009.

[7] Z. Li, et al., “Spectrum: a hybrid nanophotonic-electric on-chip network,” Proc. DAC 2009: 575-580

[8] D. Vantrease et al., “Corona: System Implications of Emerging Nanophotonic Technology,” Proc. ISCA, pp. 153-164, 2008.

[9] A. Joshi et al., “Silicon-Photonic Clos Networks for Global On-Chip Communication”, Proc. NOCS 2009.

[10] R.W. Morris, A.K. Kodi, “Power-Efficient and High-Performance Multi-level Hybrid Nanophotonic Interconnect for Multicores," Proc. NOCS, pp.207-214, 2010

[11] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, “Optimization by Simulated Annealing”, Science 220, 1983, pp. 671-680.

[12] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. “Equations of State Calculations by Fast Computing Machines”. Journal of Chemical Physics, 21(6):1087-1092, 1953.

[13] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by Simulated Annealing, Science”, 220(4598): 671-680, 1983.

[14] V. Černý, “A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm.” Journal of Optimization Theory and Applications, 45:41-51, 1985.

[15] M. Lundy, A. Mees, “Convergence of an Annealing Algorithm”. Math. Prog., 34, 111-124, 1986.

[16] P. Mazumder, Genetic algorithms for VLSI Design, Layout & Test Automation. Prentice-Hall, 1999.

[17] R. Eberhart, J. Kenned, “A new optimizer using particle swarm theory”, Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.

[18] J. W. Goodman et al., “Optical Interconnects for VLSI Systems," Proceedings of the IEEE, Vol. 72, No. 7, July 1984.

[19] J.H. Collet, F. Caignet, F. Sellaye, and D. Litaize, “Performance Constraints for Onchip Optical Interconnects,” IEEE J. Selected Topics in Quantum Electronics, vol. 9, no. 2, pp. 425-432, Mar./Apr. 2003.

[20] G. Tosik, et al., “Power dissipation in optical and metallic clock distribution networks in new VLSI technologies”, IEE Electronics Letters, vol. 40, no. 3, pp. 198-200, Feb. 2004.

[21] M.J. Kobrinsky et al., “On-Chip Optical Interconnects,” Intel Technology J., vol. 8, no. 2, pp. 129-142, May 2004.

[22] S. Murali, G. D. Micheli, “Bandwidth-Constrained Mapping of Cores onto NoC Architectures,” Proc. DATE, 2004.

[23] G. Ascia, V. Catania, and M. Palesi. “Multi-objective Mapping for Mesh-based NoC Architectures”. Proc. CODES+ISSS 2004.

[24] J. Hu, R. Marculescu, “Energy-Aware Mapping for Tile-based NoC Architectures under Performance Constraints,” Proc. ASPDAC 2003.

[25] N. Kapadia, S. Pasricha, “VISION: a framework for voltage island aware synthesis of interconnection networks-on-chip,” Proc. Great Lakes Symposium on VLSI, 2011, pp. 31-36.

[26] S. Murali., et al., “Designing application-specific networks on chips with floorplan information”, Proc. ICCAD 2006, pp. 355-362.

[27] K. Srinivasan, et al., “A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures,” Proc. DATE 2006

[28] J.Chan, et al., “NoCOUT: NoC Topology Generation with Mixed Packet-switched and Point-to-Point Networks,” Proc. ASPDAC, 2008.

[29] S. Kwon et al., "POSEIDON: A framework for application-specific network-on-chip synthesis for heterogeneous chip multiprocessors", Proc. IEEE ISQED, 2011

[30] C. Teo, Y. Foo, S. Chien, A Low, B. Venkatesh, A. You. “Optimal placement of wavelength converters in WDM networks using particle swarm optimizer.” IEEE International Conference on Communications, pages 1669 – 1673, 2004.

[31] P. Tawdross, A. Konig, “Investigation of particle swarm optimization for dynamic reconfiguration of field-programmable analog circuits”. In Hybrid Intelligent Systems, Fifth International Conference on, page 6 pp., 2005.

[32] S.C. Woo et al.“The SPLASH-2 programs: Characterization and methodological considerations”, ISCAS, 1995.

[33] I.-W. Hsieh, et al., “Ultrafast-Pulse Self-Phase Modulation and Third-Order Dispersion in Si Photonic Wire-Waveguides,” Opt. Exp., 2006.

[34] M. Haurylau, et al., “On-chip Optical Interconnect Roadmap: Challenges and Critical Directions,” IEEE JQE 12(6), Nov 2006.

[35] A. Kahng, et al., “ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration” Proc. DATE, 2009.

[36] SystemC initiative. www.systemc.org.