The Potential of On-Chip Multiprocessing for QCD Machines*

Gianfranco Bilardi[1], Andrea Pietracaprina[1], Geppino Pucci[1], Fabio Schifano[2], and Raffaele Tripiccione[2]

[1] Dipartimento di Ingegneria dell'Informazione, Università di Padova, Via Gradenigo 6/B, 35131, Padova, Italy. E-mails: {bilardi,capri,geppo}@dei.unipd.it
[2] Dipartimento di Fisica, Università di Ferrara, and INFN, Via del Paradiso 12, 44100 Ferrara, Italy. E-mails: {schifano,lele}@fe.infn.it

Abstract. We explore the opportunities offered by current and forthcoming VLSI technologies to on-chip multiprocessing for Quantum Chromo Dynamics (QCD), a computational grand challenge for which over half a dozen specialized machines have been developed over the last two decades. Based on a careful study of the information exchange requirements of QCD both across the network and within the memory system, we derive the optimal partition of die area between storage and functional units. We show that a scalable chip organization holds the promise to deliver from hundreds to thousands flop per cycle as VLSI feature size scales down from 90 nm to 20 nm, over the next dozen years.

1 Introduction

The high-end supercomputers of the near future will consist of many thousands of processing chips, each featuring billions of transistors [32]. Yet, in today's general purpose microprocessors, only a small percentage of the available transistors goes into the functional units that actually process data, while the vast majority goes into memory and into the circuitry that orchestrates the execution of instructions. The main reason can be traced to a major bottleneck in computing systems stemming from the limited bandwidth available across the chip boundary. Thus, although a chip could host hundreds of functional units, it would generally be difficult to feed them with data so to attain a significant fraction of peak performance.

The impact of the chip I/O bottleneck varies with the application, critically depending upon its computation/communication ratio. For specialized domains, chip organizations with many functional units are receiving growing attention. Announced in 2005 are products such as the IBM/Sony/Toshiba Cell capable of 64 flop/cycle at about 4 GHz [24] and the ClearSpeed CSX600, capable of 192 flop/cycle, at 250MHz [16]. A number of research projects have also been active for a few years, such as BlueGene/Cyclops (64 flop/cycle at 500 MHz) [5] and TRIPS (targeting 5 Tflops in a 35 nm VLSI implementation) [36].

In this paper, we explore whether aggressive on-chip multiprocessing is a viable avenue for supercomputing in the domain of Quantum Chromo Dynamics (QCD). QCD

* This research was supported in part by MIUR of Italy under project "ALGO-NEXT: ALGOrithms for the NEXT generation Internet and the Web", and by the University of Padova under Grant CPDA033838.

is the quantum field theory of the strong interaction explaining the structure of the hadrons, a family of particles including the proton and the neutron, as composites of "elementary" entities known as quarks, which interact by exchanging "colored" gluons. While widely believed to be the correct theory of the strong interaction, QCD is almost intractable computationally and it is regarded as a computational grand challenge [18]. Several specialized QCD computers have been designed, built, and successfully operated over the last two decades (see [28] for an extensive coverage). A sample of such machines includes the sequel of APE computers, developed in Europe [4, 8, 33–35], the CP-PACS computer, developed in Japan mainly for QCD simulations [23], and several similar efforts in the United States starting by the pioneering GF11 supercomputer [9] and continuing with the QCDSP [27] and QCDOC [15] projects.

Asymptotically, QCD computations are constrained by chip I/O bandwidth. In fact, the four dimensional nature of QCD lattices leads to communication requirements that scale with the (3/4)-th power of the computation, whereas in a (planar) chip I/O bandwidth only scales with the (1/2)-th power of the area. Suitable adaptations of arguments developed in [29, 19] show that, if chip area were to grow arbitrarily large, only a vanishing fraction of it would be occupied by effectively utilized functional units[3]. In this paper, we investigate whether and when, with the evolving VLSI technology, increasing chip size (in square feature sizes) will yield diminishing returns[4]. We reach the following encouraging conclusions regarding QCD machines:

- Current (90 nm) VLSI technology is far from the asymptotic chip I/O bottleneck, which will not be severe even for a feature size of 20 nm, to become feasible around 2017 according to the International Technology Roadmap for Semiconductors (ITRS, 2004 update).
- It is currently feasible to realize QCD machines with a few thousands nodes, each containing a processing chip with 8-16 MByte of on-chip (embedded DRAM) memory and achieving 100-150 flop/cycle.
- A uniformly scalable machine organization can harness the technological potential becoming available as feature size shrinks down to 20 nm, presumably over the next 12 years. At 20 nm, a few Kflop/cycle will be achievable on one chip.

Even at modest frequencies, say 20% of state of art, the above figures appear attractive[5].

The reminder of this paper is organized as follows. In Section 2, we introduce the notion of information exchange of a computation, modeled by a function $I(n, m)$. This is the number of bits that a subsystem equipped with m bits of memory exchanges with the rest of the system when handling a subcomputaton of size n. We also consider the number $w(n)$ of operations for the same subcomputation. In terms of these quantities, we derive the optimal split of the available area between memory and functional units. In Section 3, we develop a careful study of $I(n, m)$ for the computation of the Dirac operator, which takes nearly 90% of running time for a typical QCD simulation. The main

[3] Three dimensional integration would postpone, but not escape the same conclusion.

[4] Beyond this point, further increases of computational power may still be achievable, by increasing clock frequency and by assembling systems with larger numbers of chips.

[5] ITRS on-chip local clock rates are 5.2 GHz for 2005, 39.7 GHz for 2016, and 53.2 GHz for 2018.

contribution of this section is the identification of schedules that reduce $I(n, m)$. The analysis precisely determines constant factors to enable finite-horizon considerations, in addition to asymptotic ones. In Section 4, we evaluate the potential of QCD machines realizable in current and future technologies, reaching the conclusions outlined above.

2 Methodology for Memory-Communication-Processing Tradeoffs

We focus on the space of multiprocessor machines realized as networks of nodes, interconnected according to a suitable topology. Each node consists of a processing chip, equipped with some on-chip memory and directly connected both to a local off-chip memory and to the processing chips of the neighbouring nodes, in the given topology.

The key result of this section is a characterization of the optimal partition of the node area between memory and functional units, in terms of the computational requirements of the target application, the number of nodes of the multiprocessor, and the area of the processing chip at a node. We begin by introducing a number of quantities related to machine, to technology, and target computation.

Machine Parameters. We characterize the machine through the following quantities: the number P of processing nodes; the number F of operations per cycle that can be executed at a node; the number m of on-chip memory bits; and the bandwidth b, in bits per cycle, between the processing chip and the rest of the system (both incoming and outgoing). This bandwidth is partitioned as $b = b_{lc} + b_{nb}$ where b_{lc} is the bandwidth with the local off-chip memory and b_{nb} is the aggregate bandwidth between the chip and its neighbors.

Technology Parameters. We consider the following parameters of the underlying VLSI technology (whose values and their scaling with VLSI feature size will be discussed in Section 4): the area A of the processing chip; the area A_{F} of one functional unit, pipelinable at one operation per cycle (this parameter clearly depends also on the adopted logic design); and the area A_{m} required for the storage of one memory bit on the processing chip.

Computation Requirements. The target computation is characterized by the following quantities, some of which assume a specific mapping onto the machine under consideration (an accurate analysis of these quantities for QCD computations will be developed in Section 3): the input size N (in QCD, the number of lattice points); the input size per node $n = N/P$; the word length L_w, in bits; the total work or number of operations $W(N)$; the work per node $w(n) = nW(N)/N$ (assuming, as reasonable, even distribution of work among the processing nodes); and the *information exchange* in bits, $I(n, m)$, between the processing chip and the rest of the system. The latter quantity can be decomposed as $I(n, m) = I_{lc}(n, m) + I_{nb}(n, m)$, where the two terms account for exchanges with the off-chip memory and near neighbors, respectively.

Under ideal conditions (i.e., all resources fully used at all times) *execution time* is

$$T^{id}(n, m) = \max\{w(n)/F, I(n, m)/b\} \ . \tag{1}$$

In order to minimize T^{id} we would like to increase both F and m, the latter because more storage on the processing chip will reduce the exchange with local off-chip

memory, hence $I(n, m)$. Given the total area budget A, the question then is how to best partition it between memory and functional units. In summary, we have the following optimization problem:

$$\min \max\{w(n)/F, I(n, m)/b\} \tag{2}$$

$$s.t. \quad A_{\mathrm{F}}F + A_{\mathrm{m}}m \leq \eta A \ , \tag{3}$$

where η is the fraction of the chip area actually used for functional units and memory (as opposed to control logic, instruction caches, intra-chip communication, etc.) Estimates of η require further assumptions about chip organization; in Section 4, we provide such estimates for QCD. It is straightforward to argue that, at the optimal point for (2, 3), the two terms in the objective function (2) must be equal, and constraint (3) must be satisfied with equality, so that:

$$w(n)/F = I(n, m)/b \ , \tag{4}$$

$$A_{\mathrm{F}}F + A_{\mathrm{m}}m = \eta A \ . \tag{5}$$

Combining the two above equations yields

$$A_{\mathrm{F}}w(n)b/I(n, m) + A_{\mathrm{m}}m = \eta A \ . \tag{6}$$

If the functions $w(n)$ and $I(n, m)$ are known for the computation, given n and A, this equation determines a value $m^*(n, A)$ for m. Eq. (4) then provides the number of functional units as $F^*(n, A) = (\eta A - A_{\mathrm{m}}m^*(n, A))/A_{\mathrm{F}}$. Computation time becomes $T^{id*}(n, A) = w(n)/F^*(n, A)$. The values of n and A, which are arbitrary in the preceding analysis, could be chosen to optimize a suitable cost-performance function.

Remarks on Information Exchange. Information exchange plays a pivotal role in the methodology outlined above. The analysis of communication has often been developed by quantifying the information exchanged across a suitable partition of the system into two complementary subsystems, such as two subsets of nodes in a network (see, *e.g.*, [38, 31, 11]). Communication also arises within hierarchical memory systems; in this context, of particular interest is information exchanged across levels of the hierarchy, as reflected *e.g.*, in the notions of I/O complexity [21] and access complexity [10].

Our information exchange $I(n, m)$ measures simultaneously effects due to distribution of data across different nodes and across different memory levels, since the chip boundary acts as a separator for both the network and the memory system. Correspondingly, the techniques for minimizing the information exchange combine those traditionally used when optimizing the mapping of a given computation onto a fixed-topology network [25] with those used when optimizing the performance on a memory hierarchy [2, 7, 3]. Interactions between network and memory-hierarchy communication arise naturally in machines where speed of light is an active constraint [12]. A close relationship between network proximity and temporal locality has been exposed in [20].

3 Memory-Communication-Processing Tradeoffs for LQCD

In this section, we apply the methodology developed in Section 2 to lattice QCD (LQCD), a discretized version of QCD, currently the most amenable for non-perturbative com-

putations. After a short introduction of the needed concepts and notations, we analyze the information-exchange of the Dirac operator, the core module within LQCD.

3.1 The LQCD Computation

The Lattice In LQCD, space-time is discretized and represented as a a four-dimensional toroidal graph, whose arcs are the domain of the gauge field and whose vertices are the domain of the particle field. More formally, for $X \geq 0$, let $Z(X)$ denote the cyclic group $\{0, 1, \ldots, X-1\}$ w.r.t. addition modulo X. For $\mathbf{N} = (N_1, N_2, N_3, N_4)$, let $Z(\mathbf{N}) = Z(N_1) \otimes Z(N_2) \otimes Z(N_3) \otimes Z(N_4)$ where $\otimes$ denotes the direct product operation between groups. Also, let $\mathcal{V}_d = \{\hat{\mu}_1, \hat{\mu}_2, \hat{\mu}_3, \hat{\mu}_4\}$, where $\hat{\mu}_i$ denotes the 4-tuple with the i-th component equal to 1 and the other components equal to 0. We are interested in modeling a four-dimensional toroidal space-time lattice defined as the directed graph $L(\mathbf{N}) = (Z(\mathbf{N}), E(\mathbf{N}))$, where $E(\mathbf{N}) = \{(x, x \pm \hat{\mu}) : x \in Z(\mathbf{N}),\ \hat{\mu} \in \mathcal{V}_4\}$. For $N = N_1 N_2 N_3 N_4$, it is $|Z(\mathbf{N})| = N$ and $|E(\mathbf{N})| = 8N$.

The Gauge Field Recall that SU_3 denotes the *special unitary* (Lie) group of the 3×3 unitary matrices with determinant equal to one. (A matrix U is *unitary* when $U^\dagger = U^{-1}$, i.e., when its transpose conjugate is also its inverse.) A *gauge field* is a map $U : E(\mathbf{N}) \rightarrow SU_3$ which associates an SU_3 matrix with each arc of the lattice, with the property that $U(x, x+\hat{\mu}) = (U(x+\hat{\mu}, x))^{-1} = (U(x+\hat{\mu}, x))^\dagger$, for any $x \in Z(\mathbf{N})$ and $\pm\hat{\mu} \in \mathcal{V}_4$.

The Fermion Field A *(pseudo) fermion field* is a map $\Psi : Z(\mathbf{N}) \rightarrow C^{3\times 4}$ which associates a 3×4 complex matrix with each vertex of the toroidal lattice. In LQCD, the row index ranges over color eigenstates, while the column index ranges over spin eigenstates. It is useful to regard each 3×4 complex matrix as a 12-component complex vector.

The Dirac Operator The Dirac operator provides the dynamical description of a given quark and depends both upon the gauge field and on a parameter K related to the mass of the quark being modeled by the operator. Mathematically, the Dirac operator is a (sparse) linear function which maps a fermion field into another fermion field.

Definition 1. *Given a real scalar quantity K and a gauge field configuration U, the* Dirac operator $D_{K,U} : C^{12N} \rightarrow C^{12N}$ *is the linear operator on the space of fermion fields defined by the relation*

$$\Phi(x) = [D_{K,U}\Psi](x) = K\Psi(x) + \sum_{\pm\hat{\mu}\in\mathcal{V}_4} U(x, x+\hat{\mu})\Psi(x+\hat{\mu})\Gamma_{\hat{\mu}} \ , \qquad (7)$$

for each $x \in Z(\mathbf{N})$, where the $\Gamma_{\hat{\mu}}$'s are the 4×4 Dirac matrices, a key feature of which is that in each row/column all entries are 0, except for one which belongs to $\{1, i, -1, -i\}$.

An LQCD computation is based on a Montecarlo-Metropolis approach that generates a random walk in the space of gauge field configurations. The key kernel of the computation is the inversion of the Dirac operator which is typically obtained through

its iterated application according to Eq. (7) starting from an initial fermion field. In current simulations, this operation accounts for nearly 90% of the overall simulation time.

3.2 Computation Requirements of the Dirac Operator

In this subsection, we determine the requirements of the computation of the Dirac operator $\Phi = D_{K,U}\Psi$, defined by Eq. (7). We assume that a complex number is represented by two words, hence each 3×4 matrix $\Phi(x)$ or $\Psi(x)$ occupies 24 words, while each 3×3 matrix $U(x, x+\hat{\mu})$ occupies 18 words. To evaluate the total work $W(N)$, measured in flop (floating point operations), required by the operator, we consider the computation of $\Phi(x)$ for a lattice point x, taking into account that the multiplications by matrices $\Gamma_{\hat{\mu}}$ do not contribute any flop, since they essentially amount to permutations and sign changes. Each of the 8 $U(x, x+\hat{\mu})\Psi(x+\hat{\mu})$ products accounts for $3 \times 3 \times 4 = 36$ complex multiplications (cmul) and 24 complex additions (cadd). Further 8×12 cadd are required by the summation, which yields a total equivalent to $8 \times 36 = 288$ complex multiply and add (cmadd). Since each cmadd requires 8 flop and 24 more flop are needed to compute $K\Psi$, we conclude that the computation of $\Phi(x)$ requires $\chi = 2328$ flop, hence

$$W(N) = \chi N = 2328N \ . \tag{8}$$

For concreteness, we assume that the Dirac operator is implemented on a P-node 3D-torus, which is a typical topology for today's supercomputers (a similar analysis could be carried out for other topologies). We partition the lattice evenly among P processing nodes so that each node is in charge of a sublattice of size $n = k \times k \times k \times N_4$, with $k = (N/(N_4P))^{1/3}$. For convenience, we order dimensions so that $N_4 = \min\{N_1, N_2, N_3, N_4\}$. We assume that at the beginning of the computation a node stores the Ψ and U fields restricted to lattice points and incident arcs of its assigned sublattice. Since the matrices $U(x, y)$ and $U(y, x)$ associated with the two arcs between points x and y are one the transpose conjugate of the other, we can store only one instance if both x and y are assigned to the node. At the end of the computation the node will also store the Φ fields restricted to its sublattice points. It is easy to see that all of the data residing at the node add up to $\sigma k^3 N_4 L_w$ bits, with $\sigma = 120$.

The following technical result, whose proof, omitted here for brevity, will be provided in the full version of this extended abstract, establishes the existence of efficient schedules tailored to various values of the on-chip memory size m:

Theorem 1. *With the above notation, there is a schedule for the computation of the Dirac operator which executes in time* $T(n, m) = \max\{w(n)/F, (I_{lc}(n, m) + I_{nb}(n, m))/b\}$ *where* $I_{nb}(n, m) = (\iota n/k)L_w$, *and* $I_{lc}(n, m)$ *exhibits the following dependence upon* m*:*

1. *for* $m = \sigma n L_w = 120 n L_w$ (large memory), $I_{lc}(n, m) = 0$*;*
2. *for* $m = (96k^3 + 432k^2 + o(k^2))L_w$ (medium memory), $I_{lc}(n, m) = \sigma n L_w$*;*
3. *for* $m = (96k^2 s + 288ks + O(1))L_w$ *with* $1 \le s \le k$ (small memory), $I_{lc}(n, m) = ((\sigma + 66/s)n + 132k^2)L_w$*;*

Clearly, $I_{lc}(n, m)$, hence its contribution to running time, increases as m decreases. In particular, once m goes substantially below the threshold of $\sigma n L_w$, one can see that, if the machine is balanced in the sense that the two terms in the expression for $T(n, m)$ are equal, then less than $w(n)L_w/I_{lc}(n, m) = \chi/\sigma < 20$ flop per word exchanged with the local off-chip memory can be sustained. Considerably larger values are instead achievable when $m \geq \sigma n L_w$, since in this case there is no need to transfer the fields back and forth between the processing chip and the off-chip memory, at each iterative application of the Dirac operator.

4 Performance Potential of Future QCD Machines

Based on the resource tradeoff equations derived in the previous sections, we now evaluate the performance potential of future QCD machines. We introduce a number of assumptions on technology parameters, formulated in terms of VLSI feature size λ, to allow for scaling considerations.

Technology Assumptions. We let $A = (\ell\lambda)^2$ denote the die area, assuming, for simplicity, a square of sidelength ℓ, in units of λ. Referring to an area range of 81-324 mm^2, we see that $\ell \in [1 \div 2] \cdot 10^5$ is representative of today's 90 nm scenario, while $\ell \in [4.5 \div 9] \cdot 10^5$ would represent the 20 nm scenario forecast for 2017. We express the area taken by one bit of storage as $A_{\rm m} = \alpha_{\rm m}\lambda^2$, estimating $\alpha_{\rm m} = 50$, for embedded DRAM. We express the area of a floating-point functional unit as $A_{\rm F} = \alpha_{\rm F}\lambda^2$. Clearly, $\alpha_{\rm F} = \alpha_{\rm F}(L_w)$, that is, the area does depend upon the wordlength of the operands. For the case study below, where $L_w = 64$, we generously estimate $\alpha_{\rm F}(64) = 10^8$, so that a number of registers and some auxiliary logic is also accounted for (see, *e.g.*, [17, 26]). Asymptotically, $\alpha_{\rm F}(L_w) = \theta(L_w^2)$, due to the $A = \theta(L_w^2/T^2)$ complexity of a VLSI multiplier (see [1, 14] for lower bounds and [13] for upper bounds). However, since in our context $\alpha_{\rm F}(L_w)$ is actually an average area between adder, multiplier, and register file, and since relatively small values of L_w are under consideration, we can expect the actual behavior being between linear and quadratic. Finally, we assume a chip bandwidth proportional to the perimeter, *i.e.*, $b = \beta 4\ell$. We estimate $\beta = 1/3000$, which is conservative at $\lambda = 90$ nm (for $\ell = 2 \cdot 10^5$, $b = 266$, compared to the ITRS figure of 1800 I/O signals per chip) and somewhat conservative at $\lambda = 20$ nm (for $\ell = 9 \cdot 10^5$, $b = 1200$, compared to the ITRS figure of 3000).

Input Assumptions. Consider a lattice of size $N = kP_1 \times kP_2 \times kP_3 \times N_4$ to be mapped onto a $P_1 \times P_2 \times P_3$ three-dimensional torus of $P = P_1P_2P_3$ nodes, each processing a $k \times k \times k \times N_4$ sublattice of $n = k^3N_4$ points. Since (a) $N = 64^4$ is a rather large size processed on today's teraflop machines, (b) the overall computation requirements of a QCD simulation (including $O(N^{3/4})$ Dirac computations) grow approximately as $O(N^{7/4})$, and (c) a thousandfold improvement can be expected during the horizon we are investigating, it is reasonable to assume for lattice size a range $64^4 - 128 \cdot 256^3$, throughout which one can always choose to completely map within a node an entire dimension of size (approximately) $N_4 = 128$.

Machines in the Large Memory Regimen. In this regimen, where all fields are stored

on chip, we have: $n = N_4 k^3$, $w(n) = \chi n$ with $\chi = 2328$ (from Eq. (8)), $m = L_w \sigma n$, with $\sigma = 120$ and $L_w = 64$, and $I(n,m) = L_w \iota n/k$, with $\iota = 288$, from Theorem 1. Then, Eq. (4) gives the number of functional units as $F = w(n)b/I(n,m) = \chi n 4\beta\ell/(L_w \iota(n/k)) = (97/576000)\ell k$. Assuming for now $\eta = 1$, Eq. (5) can be rewritten in the λ-invariant form $\alpha_{\rm F} F + \alpha_{\rm m} m = \ell^2$, whence, after plugging in the technology parameters and the above relation for F, we have:

$$\alpha_{\rm F} F + \alpha_{\rm m} m = \gamma \ell k + \delta N_4 k^3 = \ell^2 \ , \tag{9}$$

where $\gamma = 97 \cdot 10^5/576$ and $\delta = 384 \cdot 10^3$. Assuming $N_4 = 128$, Table 1 shows the numeric solutions of the above equations for a sample of values of chip sidelength ℓ. We can make a few observations:

- Current technology ($\ell = 10^5, 2 \cdot 10^5$) would already enable hundreds of flop/cycle.
- Projected 2017 technology ($\ell = 4 \cdot 10^5, 8 \cdot 10^5$) holds the promise of thousands of flop/cycle.
- The fraction $\alpha_{\rm F} F/\ell^2$ of the die area utilized for functional units is substantial in the range being considered for ℓ, although it does decrease with ℓ. Indeed, one could derive from Eq. (9) that this fraction vanishes asymptotically as $(\gamma/(\delta N_4)^{2/3})\ell^{-1/3}$.

Machines in the Medium Memory Regimen. Consider now the case of medium memory. From Theorem 1 we have $I(n,m) = (\iota(n/k) + \sigma n)L_w$ and $m \simeq (96k^3 + 432k^2)L_w$. Hence, $F = (97/(576 + 240k))10^{-3}\ell k$ and $\ell^2 = (97/(576 + 240k))10^5 \ell k + 32 \cdot 10^2 (96k^3 + 432k^2)$.

As we can see from Table 1, for $\ell \leq 25000$, the medium-memory regimen achieves a better floating point performance than the large-memory regimen. The reason is that, when m is below a certain threshold, the node sublattice approaches a 1-dimensional array of N_4 points, with an unfavourable computation/communication ratio. As m and n increase with ℓ, this situation is quickly reversed, since $\alpha_{\rm F} F/\ell^2$ vanishes as $(97 \cdot 10^4)/(24\ell)$. We also observe from the table that the growth rate of n as a function of ℓ is much smaller for large memory than for medium memory, so the latter regimen affords implementations with smaller numbers of nodes $P = N/n$.

Table 1. QCD chip parameters ($L_w = 64$ bits): sidelength ℓ (units of λ); b: I/O bandwidth (bits/cycle); $n = k^3 N_4$: number of sublattice points processed ($N_4 = 128$); m: on-chip memory (bits); F: flop/cycle; $\alpha_{\rm F} F/\ell^2$: fraction of area devoted to FP units.

		Large Memory Regimen				**Medium Memory Regimen**			
$\ell/10^5$	b	n	$m/10^6$	F	$\alpha_{\rm F} F/\ell^2$	n	$m/10^6$	F	$\alpha_{\rm F} F/\ell^2$
0.25	33	$2.57 \cdot 10^2$	1.9	5	0.84	$5.26 \cdot 10^3$	0.5	6	0.95
0.50	67	$1.54 \cdot 10^3$	11	19	0.76	$2.47 \cdot 10^5$	15	17	0.68
1.00	133	$8.48 \cdot 10^3$	62	67	0.67	$2.24 \cdot 10^6$	120	37	0.37
2.00	267	$4.36 \cdot 10^4$	319	233	0.58	$1.23 \cdot 10^7$	616	77	0.19
4.00	533	$2.12 \cdot 10^5$	1549	788	0.49	$5.68 \cdot 10^7$	2752	157	0.10
8.00	1067	$9.82 \cdot 10^5$	7194	2630	0.41	$2.45 \cdot 10^8$	11601	317	0.05
16.0	2133	$4.41 \cdot 10^6$	32298	8677	0.34	$1.02 \cdot 10^9$	47609	639	0.03

Wordlength. While Table 1 shows the parameters for $L_w = 64$, values for different wordlengths can be easily obtained from our equations, given the appropriate value for $\alpha_{\rm F}(L_w)$. Simple arguments on the structure of the equations show that, for a fixed ℓ, when the wordlength is halved, then F is slightly more than doubled if $\alpha_{\rm F}$ grows linearly with L_w, while F is slightly less than quadrupled if $\alpha_{\rm F}$ grows quadratically with L_w. Thus, operating with the smallest wordlength that guarantees the numerical properties of the QCD algorithms, probably somewhere between 32 and 64 bits, may lead to nonnegligible savings with respect to the case for 64 bits.

4.1 Chip organization

In the preceding analysis, by setting $\eta = 1$ in Eq. (5), we have ignored the area requirements due to control structures and intra-chip data and instruction transfers. To show how these requirements can be kept small (within 10%, corresponding to $\eta \geq 0.9$), we sketch a chip organization for machines tailored to the large-memory regimen and to suitable chip size, say $\ell \geq 10^5$.

Letting $p^2 = F/8$, we consider a chip organized as a $p \times p$ two-dimensional mesh of small processing elements (SPEs). Each SPE is endowed with m/p^2 bits of local off-chip memory and with 8 floating point units (which naturally exploit the 8 flop of the complex multiply-and-add operation, very abundant in QCD codes).

Controller. We envisage a SIMD organization with a single centralized structure in charge of flow control broadcasting control words to all SPEs. Without entering into details, we estimate its complexity to be similar to that of one functional unit and its layout to fit in an $\ell_c \times \ell_c$ region, with $\ell_c = 10^4$. A region of the same shape and size can be also budgeted for a program memory (2 Mbit of embedded DRAM). Thus, controller and program memory can be accommodated in a centrally placed (say) vertical layout strip of width $\ell_c = 10^4$.

Control distribution. To distribute control words, and to support various reduction operations, we make provision for a binary tree rooted at the controller, with p^2 leaves at the SPEs, and with edge bandwidth $b_t = 128$ (in bit/cycle). Adopting an H-layout [37], the tree requires $(p-1)b_t < \sqrt{F/8}b_t$ bandwidth, both vertically and horizontally.

Data transfers. For the parameter ranges we are considering, the node's sublattice can be mapped so that neighboring lattice points are assigned either to the same SPE or to two near-neighbor SPEs. Inter-node data can then be routed by p row busses and p column busses, with overall bandwidth $b/2$, both in the vertical and in the horizontal direction.

In order to translate the bandwidth requirements into area occupancy, we need to estimate the width $1/\beta_0$ (in units of λ) of a connection carrying one bit/cycle. Based on an exercise carried out on currently available 130 nm technology, we set $1/\beta_0 = 5$.

In summary, we obtain an $\ell_h \times \ell_v$ layout, where $\ell_v = \ell + (1/\beta_0)(\sqrt{F/8}b_t + b/2)$ and $\ell_h = \ell_v + \ell_c$. Considering that $F < \ell^2/\alpha_{\rm F}$, $\alpha_{\rm F} = 10^8$, $b = 4\beta\ell$, $1/\beta_0 = 5$, $\beta = 1/3000$, $b_t = 128$, $\ell_c = 10^4$, and $l \geq 10^5$, we can derive that $\ell_v = (1+\epsilon)\ell$, with $\epsilon = (b_t/\sqrt{8\alpha_{\rm F}} + 2\beta)/\beta_0 \leq 0.026$, whence $\eta = \ell^2/\ell_h\ell_v = [(1+\epsilon^2)+(1+\epsilon)\ell_c/\ell)]^{-1} \geq 0.9$.

Table 2. Relevant parameters of current processors (see Section 4.2 for the definition of the ratio ξ). In the table, ξ_{LM} and ξ_{MM} refer, respectively, to the large and to the medium memory regimens.

	apeNEXT	**BG/L**	**Cell**	**CSX600**	**ITANIUM2**	**QCDOC**
frequency	200Mhz	700Mhz	3.2Ghz	250Mhz	1.6Ghz	500Mhz
λ	180nm	130nm	90nm	130nm	90nm	130nm
L_w	64	32/64	32/64	64	64	64
F	8	4	64/8	192	4	2
m	32kb	32Mb	20Mb	4.5Mb	72Mb	32Mb
b_{lc}	128	62.85	64	102.4	x	41.6
b_{nb}	48	24	192	256	$32 - x$	21.8
ξ_{LM}	n.a.	6.07/2.28	2.27/6.06	0.17	4.04	4.13
ξ_{MM}	0.76	9.53/4.77	0.61/2.42	0.16	2.20	6.30

4.2 Current processors

It might be instructive to place some recent processors in the (b_{lc}, b_{nb}, m, F) space. Ideally, given b_{lc}, b_{nb} and m, the number of flop/cycle that could be sustained for the Dirac computation is $F^* = w(n)/\max(I_{lc}(n,m)/b_{lc}, I_{nb}(n,m)/b_{nb})$. Thus, the ratio $\xi = f^*/F$ can be viewed as a measure of how well the machine is balanced (for QCD). If $\xi = 1$, then the balance is perfect. If $\xi < 1$, the bandwidth and memory resources are sufficient to sustain only a fraction ξ of the available performance. If $\xi > 1$, the bandwidth and memory resources would be sufficient to sustain a multiple ξ of the available performance. Note however that ξ is only a measure of balance of one of the architectural parameters (b_{lc}, b_{nb}, m, F), once the remaining ones have been fixed, based on some criteria other than our methodology: ξ alone does not capture how suitable a given architecture is for QCD computing. Six processors [35, 6, 24, 16, 22, 15] are listed in Table 2, with relevant features and the corresponding ξ metric, for both the large and the medium memory regimens. Many observations could be made, keeping in mind that it would not be appropriate to consider ξ as a figure of merit for the corresponding design, which is likely to have been optimized for different technologies (e.g., SRAM vs DRAM) and for applications different from QCD or, in the case of apeNEXT and QCDOC, for different memory regimens as well as for different codes (which do not necessarily use a schedule that minimizes the information exchange). We leave most of these observations to the interested reader, except for noting that we start to see the appearance of compute-intensive architectures and that at least in one case (the Cell processor, especially in single precision) the design parameters are remarkably close to the design space that we have identified in the previous subsections.

5 Conclusions

In this paper, we have developed an approach to analyze tradeoffs between bandwidth, memory and processing for a given computation, providing quantitative guidelines to evaluate or design a machine for such a computation. By this approach, we have shown

that, broadly speaking, chips where a substantial fraction of the silicon is used for functional units could deliver from hundreds to thousands of flops per cycle on QCD computations, over the next decade.

The models of this paper only provide for a first-order analysis of the resource tradeoffs. More accurate models and analyses would obviously be needed to provide sound guidance in an actual design. In particular, wire length and corresponding delays become increasingly critical as feature size shrinks. In the chip organization sketched in Section 4.1, while near-neighbour connections should not pose a serious problem, the long wires of the tree for control distribution require further attention (pipelining the instruction stream might be sufficient for QCD codes, which have few control dependences; multiple controllers on the same chip can be another avenue). A careful analysis of these issues is a prerequisite to any credible estimate of achievable clock frequencies. Power consumption is another increasingly relevant issue, completely neglected in this preliminary study. Here, we simply observe that as static power accounts for an increasing fraction of energy consumption, as soon as area is converted into transistor a price is paid. Therefore, architectures as the one we have outlined, which maximize the percentage of area that does useful processing, become increasingly attractive from the power perspective.

At a broader level, the proposed approach and its refinements could be used to study other domains that can potentially take advantage of on-chip supercomputing.

Acknowledgments Helpful and constructive discussions with Giacomo Marchiori, Jose Moreira, and Pratap Pattnaik are gratefully acknowledged.

References

1. H. Abelson and P. Andreae. Information transfer and area-time tradeoffs for VLSI multiplication. *Communications of the ACM*, 23(1):20–23, 1980.
2. A. Aggarwal, A.K. Chandra, and M. Snir. Hierarchical memory with block transfer. In *Proc. of the 28th IEEE Symp. on Foundations of Computer Science*, pages 204–216, 1987.
3. A. Aggarwal and J.S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.
4. M. Albanese et al. The APE Computer: an Array Processor Optimized for Lattice gauge Theory Simulations. *Comput. Phys. Commun.* 45:345, 1987.
5. F. Allen et al. Blue Gene: a vision for protein science using a petaflop supercomputer. IBM Systems Journal, 40(2)310-327, 2001.
6. G. Almasi et al. Design and implementation of message passing services for the Blue Gene/L supercomputer. *IBM J. Res. Develop.*, 49(2/3), 2005.
7. B. Alpern, L. Carter, E. Feig, and T. Selker. The uniform memory hierarchy model of computation. *Algorithmica*, 12(2/3):72–109, 1994.
8. C. Battista et al. The APE-100 Computer:(I) the Architecture. *Int. J. High Speed Computing* 5:637, 1993.
9. J. Beetem, M. Denneau, and D. Weingarten. The GF11 supercomputer. In *Proc.of 12th Int. Symposium on Computer Architecture*, pages 108-115, 1985.
10. G. Bilardi, A. Pietracaprina, and P. D'Alberto. On the space and access complexity of computation dags. In *Proc. of 26th Workshop on Graph-Theoretic Concepts in Computer Science*, LNCS 1928, pages 47–58, 2000.

11. G. Bilardi and F.P. Preparata. Area-time lower-bound techniques with application to sorting. *Algorithmica*, 1(1):65–91, 1986.
12. G. Bilardi and F.P. Preparata. Processor-time tradeoffs under bounded-speed message propagation: Part II, lower bounds. *Theory of Computing Systems*, 32:531–559, 1999.
13. G. Bilardi and M. Sarrafzadeh. Optimal VLSI circuits for the discrete Fourier transform. *Advances in Computing Research* 4:87–101, JAI Press, Greenwich Connecticut, 1987.
14. R.P. Brent and H.T. Kung. The chip complexity of binary arithmetic. *J. Ass. Comp. Mach.* 28(3):521–534, 1981.
15. D.Chen et al. QCDOC: A 10-teraflops scale computer for lattice QCD. In *Proc. of 18th Intl. Symposium on Lattice Field Theory (Lattice 2000)*, Bangalore, India, August 2000.
16. ClearSpeed Site: `www.clearspeed.com`
17. J. Clouser et al A 600-MHz superscalar floating-point processor. *IEEE Journal on Solid-State Circuits*, 34(7):1026-1029, July 1999.
18. D.E. Culler, J.P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, San Mateo, CA, 1999.
19. R. Cypher, Theoretical aspects of VLSI PIN limitations, SIAM J. Comput., Vol. 2, No.2, pp. 356-378, April 1993.
20. C. Fantozzi, A. Pietracaprina, and G. Pucci. Seamless integration of parallelism and memory hierarchy. In *Proc. of 29th Int. Colloquium on Automata, Languages and Programming*, LNCS 2380, pages 856–867, July 2002.
21. J.W. Hong and H.T. Kung. I/O complexity: The red-blue pebble game. In *Proc. of the 13th ACM Symp. on Theory of Computing*, pages 326–333, 1981.
22. Intel Itanium2 Site: `www.intel.com/products/processor/itanium2/`
23. Y. Iwasaki. Computers for lattice field theories. *Nuclear Physics (Proc. Suppl.)* 34:78, 1994.
24. J. Kahle, M. Suzuoki, Y. Masubuchi, Cell Microprocessor Briefing, San Francisco, February 7, 2005.
25. F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.
26. S. Mueller et al The vector floating-point unit in a synergistic processor element of a Cell processor. In *Proc. 17th IEEE Int. Symp. on Computer Arithmetic*, June 2005. To Appear.
27. R.D. Mawhinney, The 1 Teraflops QCDSP Computer, Parallel Computing 25(10–11):1281-1296, 1999.
28. *Parallel Computing*, 25(10–11), 1999. Special Issue on High Performance Computing in LQCD.
29. M. Snir, I/O Limitations on multi-chip VLSI systems, Proc. 19th Allerton Conference on Communications, Control, and Computing, Monticello, IL, 1981, pp. 224-233.
30. S.M. Sze, editor. *VLSI Technology*. McGraw-Hill, New York NY, 2nd edition, 1988.
31. C.D. Thompson. *A complexity theory for VLSI*. PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, Aug. 1980. Tech. Rep. CMU-CS-80-140.
32. The Top 500 Supercomputer Sites: `http://www.top500.org`.
33. R. Tripiccione. APEmille. *Parallel Computing*, 25(10–11):1297–1309, 1999.
34. R. Tripiccione. LGT simulations on APEmachines. *Computer Physics Communications* 139:55, 2001.
35. R. Tripiccione. Strategies for dedicated computing for lattice gauge theories. *Computer Physics Communications* 169:442-448, 2005.
36. TRIPS: Tera-op Reliable Intelligently adaptive Processing System. `www.cs.utexas.edu/users/cart/trips/`.
37. J.D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, Rockville MD, 1984.
38. A.C.C. Yao. Some complexity questions related to distributive computing. In *Proc. of the 11th ACM Symp. on Theory of Comp.*, pages 209–213, 1979.