Upper bounds on the computational power of an optical model of computation

Damien Woods

Boole Centre for Research in Informatics and School of Mathematics,
University College Cork, Ireland. http://www.bcri.ucc.ie/~dw5

Abstract. We present upper bounds on the computational power of an optical model of computation called the $\mathcal{C}_2$-CSM. We show that $\mathcal{C}_2$-CSM time is no more powerful than sequential space, thus giving one of the two inclusions that are necessary to show that the model verifies the parallel computation thesis. Furthermore we show that $\mathcal{C}_2$-CSMs that simultaneously use polynomial space and polylogarithmic time decide no more than the class NC.

1 Introduction

The computational model we study is relatively new and is called the continuous space machine (CSM) [11–13, 20–23]. The original definition of the model was by Naughton [11, 12]. The CSM is inspired by classical Fourier optics and uses complex-valued images, arranged in a grid structure, for data storage. The program also resides in images. The CSM has the ability to perform Fourier transformation, complex conjugation, multiplication, addition, thresholding and resizing of images. It has simple control flow operations and is deterministic. We analyse the model in terms of seven complexity measures inspired by real-world resources.

A rather general variant of the model was previously shown [23] to decide the membership problem for all recursively enumerable languages, and as such is unreasonable in terms of implementation. Also, the growth in resource usage was shown for each CSM operation, which in some cases was unreasonably large [21]. This work motivated the definition of the $\mathcal{C}_2$-CSM, a more realistic and restricted CSM.

Recently [22] we have given lower bounds on the computational power of the $\mathcal{C}_2$-CSM by showing that it is at least as powerful as models that verify the parallel computation thesis. This thesis [4, 6] states that parallel time corresponds, within a polynomial, to sequential space for reasonable parallel models. See, for example, [18, 14, 8, 7] for details. Furthermore we have shown that $\mathcal{C}_2$-CSMs that simultaneously use polynomial SPACE and polylogarithmic TIME accept at least the class NC [22].

Here we present the other of the two inclusions that are necessary in order to verify the parallel computation thesis; we show that $\mathcal{C}_2$-CSMs computing in TIME $T(n)$ accept at most the languages accepted by deterministic Turing machines in $O(T^2(n))$ space: $\mathcal{C}_2$-CSM-TIME$(T(n)) \subseteq$ DSPACE$(O(T^2(n)))$. Also we show that $\mathcal{C}_2$-CSMs that simultaneously use polynomial SPACE and polylogarithmic TIME accept at most the class NC: $\mathcal{C}_2$-CSM-SPACE, TIME$(n^{O(1)}, \log^{O(1)} n) \subseteq$ NC. These inclusions are established via $\mathcal{C}_2$-CSM simulation by a polynomial sized, log depth, logspace uniform circuit.

2 The CSM

We begin by informally describing the model, this brief overview is not intended to be complete more details are to be found in [23, 20].

A complex-valued image (or simply, image) is a function $f : [0,1) \times [0,1) \to \mathbb{C}$, where $[0,1)$ is the half-open real unit interval. We let $\mathcal{I}$ denote the set of complex-valued images. Let $\mathbb{N}^+ = \{1,2,3,\ldots\}$, $\mathbb{N} = \mathbb{N}^+ \cup \{0\}$, and for a given CSM M let $\mathcal{N}$ be a countable set of images that encode M's addresses. Additionally, for a given M there is an *address encoding function* $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ such that $\mathfrak{E}$ is Turing machine decidable, under some *reasonable* representation of images as words. An address is an element of $\mathbb{N} \times \mathbb{N}$.

Definition 1 (CSM). *A CSM is a quintuple $M = (\mathfrak{E}, L, I, P, O)$, where*
$\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ *is the address encoding function*
$L = ((s_\xi, s_\eta), (a_\xi, a_\eta), (b_\xi, b_\eta))$ *are the addresses: sta, a, and b,*
$I = ((\iota_{1_\xi}, \iota_{1_\eta}), \ldots, (\iota_{k_\xi}, \iota_{k_\eta}))$ *are the addresses of the k input images,*
$P = \{(\zeta_1, p_{1_\xi}, p_{1_\eta}), \ldots, (\zeta_r, p_{r_\xi}, p_{r_\eta})\}$ *are the r programming symbols and their addresses where* $\zeta_j \in (\{h, v, *, \cdot, +, \rho, st, ld, br, hlt\} \cup \mathcal{N}) \subset \mathcal{I}$,
$O = ((o_{1_\xi}, o_{1_\eta}), \ldots, (o_{l_\xi}, o_{l_\eta}))$ *are the addresses of the l output images.*
Each address is an element from $\{0,1,\ldots,\Xi-1\} \times \{0,1,\ldots,\mathcal{Y}-1\}$ *where* $\Xi, \mathcal{Y} \in \mathbb{N}^+$. *Addresses a and b are distinct.*

Addresses whose contents are not specified by P in a CSM definition are assumed to contain the constant image $f(x,y) = 0$. We interpret this definition to mean that M is (initially) defined on a grid of images bounded by the constants Ξ and $\mathcal{Y}$, in the horizontal and vertical directions respectively.

In our grid notation the first and second elements of an address tuple refer to the horizontal and vertical axes of the grid respectively, and image $(0,0)$ is at the lower left-hand corner of the grid. The images have the same orientation as the grid. Figure 1 gives the CSM operations in this grid notation. Configurations are defined in a straightforward way as a tuple $\langle c, e \rangle$ where c is an address called the control and e represents the grid contents. For a more thourough introduction see [20, 23].

Next we define some CSM complexity measures. All resource bounding functions map from $\mathbb{N}$ into $\mathbb{N}$ and are assumed to have the usual properties [1].

Definition 2. *The* TIME *complexity of a CSM M is the number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.*

Definition 3. *The* GRID *complexity of a CSM M is the minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.*

$\boxed{\mathrm{h}}$: replace image in a with its horizontal 1D Fourier transform (FT).
$\boxed{\mathrm{v}}$: replace image in a with its vertical 1D FT.
$\boxed{*}$: replace image in a with its complex conjugate.
$\boxed{\cdot}$: multiply (point by point) the two images in a and b. Store result in a.
$\boxed{+}$: perform a complex (point by point) addition of a and b. Store result in a.
$\boxed{\rho}\boxed{z_\mathrm{l}}\boxed{z_\mathrm{u}}$: $z_\mathrm{l}, z_\mathrm{u} \in \mathcal{I}$; filter the image in a by amplitude using z_l and z_u as lower and upper amplitude threshold images, respectively.
$\boxed{\mathrm{st}}\boxed{\xi_1}\boxed{\xi_2}\boxed{\eta_1}\boxed{\eta_2}$: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy the image in a into the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and whose top right-hand corner address is (ξ_2, η_2).
$\boxed{\mathrm{ld}}\boxed{\xi_1}\boxed{\xi_2}\boxed{\eta_1}\boxed{\eta_2}$: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy into a the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and top right-hand corner address is (ξ_2, η_2).
$\boxed{\mathrm{br}}\boxed{\xi}\boxed{\eta}$: $\xi, \eta \in \mathbb{N}$; unconditionally branch to the image at address (ξ, η).
$\boxed{\mathrm{hlt}}$: halt.

Fig. 1. The set of CSM operations, given in our grid notation.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ constant-valued pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$. If we choose a reasonable and realistic S then the details of S are not important.

Definition 4. *The* SPATIALRES *complexity of a CSM M is the minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.*

Definition 5. *The* DYRANGE *complexity of a CSM M is the ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.*

In earlier treatments [21, 23] we defined the complexity measures AMPLRES, PHASERES and FREQ. AMPLRES and PHASERES are measures of the cardinality of discrete amplitude and phase values of the complex numbers in the range of CSM images. In the present work AMPLRES and PHASERES both have constant value of 2 which means that all images are of the form $f : [0, 1) \times [0, 1) \to \{0, \pm\frac{1}{2}, \pm 1, \pm\frac{3}{2}, \ldots\}$. Furthermore we are studying a restricted CSM to which FREQ does not apply.

Often we wish to make analogies between space on some well-known model and CSM 'space-like' resources. Thus we define the following convenient term.

Definition 6. *The* SPACE *complexity of a CSM M is the product of all of M's complexity measures except* TIME.

We have defined the complexity of a computation (sequence of configurations) for each measure. We extend this definition to the complexity of a (possibly non-final) configuration in the obvious way. Also, we sometimes talk about the complexity of an image, this is simply the complexity of the configuration that the image is in.

In [21, 20] we defined the $\mathcal{C}_2$-CSM, a restricted and more realistic class of CSM.

Definition 7 **($\mathcal{C}_2$-CSM).** *A $\mathcal{C}_2$-CSM is a CSM whose computation* TIME *is defined for $t \in \{1, 2, \ldots, T(n)\}$ and has the following restrictions:*

- *For all* TIME *t both* AMPLRES *and* PHASERES *have constant value of 2.*
- *For all* TIME *t each of* GRID*,* SPATIALRES *and* DYRANGE *is $O(2^t)$ and* SPACE *is redefined to be the product of all complexity measures except* TIME *and* FREQ.
- *Operations h and v compute the discrete FT (DFT) in the horizontal and vertical directions respectively.*
- *Given some* reasonable *binary word representation of the set of addresses $\mathcal{N}$, the address encoding function $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ is decidable by a logspace Turing machine.*

3 Circuits and representation

In this work we are using logspace uniform circuits over the complete basis $\wedge$, $\vee$ and $\neg$ with the usual complexity measures of size and depth [1, 15]. For convenience we frequently write "uniform" instead of "logspace uniform". Given a circuit c_n, its encoding $\overline{c}_n$ is a string of 4-tuples, where each tuple encodes a single gate and is of the form $(g, b, g_l, g_r) \in (\{0,1\}^+, \{\wedge, \vee, \neg\}, \{0,1\}^+ \cup \varnothing, \{0,1\}^+ \cup \varnothing)$. The tuple specifies the gate label g, the operation b that the gate computes, and the inputs g_l and g_r. For $\neg$ gates exactly one of g_l or g_r is the null symbol $\varnothing$.

Let U-SIZE,DEPTH(size(n), depth(n)) be the class of languages recognised by logspace uniform bounded fan-in circuits of size size(n) and depth depth(n), respectively. The equality $\mathrm{NSPACE}(S^{O(1)}(n)) = \text{U-SIZE}, \mathrm{DEPTH}(2^{n^{O(1)}}, S^{O(1)}(n))$ is well-known [3, 8]. Circuit depth is a measure of parallel time, hence logspace uniform circuits verify the parallel computation thesis. Specifically we are interested in the inclusion [1, 8]

$$\text{U-SIZE}, \mathrm{DEPTH}(2^{S(n)}, S(n)) \subseteq \mathrm{DSPACE}(O(S(n))). \qquad (1)$$

Suppose we are simulating a $\mathcal{C}_2$-CSM M that has TIME, GRID, SPATIALRES, DYRANGE and SPACE of $T(n)$, $G(n)$, $R_{\mathrm{S}}(n)$, $R_{\mathrm{D}}(n)$ and $S(n)$ respectively. From Definition 7, $S(n) = G(n)R_{\mathrm{S}}(n)R_{\mathrm{D}}(n)4 \leqslant c_G 2^{T(n)} c_{R_{\mathrm{S}}} 2^{T(n)} c_{R_{\mathrm{D}}} 2^{T(n)}$ for constants $c_G, c_{R_{\mathrm{S}}}, c_{R_{\mathrm{D}}}$. To simplify things (by removing the constants) we redefine the the TIME of M by increasing it by a constant, $T'(n) = \lceil \log(c2^{T(n)}) \rceil$ where $c = \max(c_G, c_{R_{\mathrm{S}}}, c_{R_{\mathrm{D}}})$. In the sequel we write $T'(n)$ as $T(n)$. SPACE is now bounded above by $2^{T(n)}2^{T(n)}2^{T(n)}$, specifically each of GRID, SPATIALRES and DYRANGE is now bounded above by $2^{T(n)}$. Next let $G(n) = G_x(n)G_y(n)$ and $R_{\mathrm{S}}(n) = R_{\mathrm{S}_x}(n)R_{\mathrm{S}_y}(n)$, where $G_x(n)$ and $G_y(n)$ are the number of grid images in the horizontal and vertical directions respectively, and where $R_{\mathrm{S}_x}(n)$ and $R_{\mathrm{S}_y}(n)$ are the number of image pixels in the horizontal and vertical directions respectively. Define $G_x(n) = G_y(n) = R_{\mathrm{S}_x}(n) = R_{\mathrm{S}_y}(n) = 2^{T(n)}$. Thus $G(n) = R_{\mathrm{S}}(n) = 2^{2T(n)}$ and $R_{\mathrm{D}}(n) = 2^{T(n)}$ and we get our final upper bound on M's SPACE: $S(n) \leqslant 2^{2T(n)}2^{2T(n)}2^{T(n)} = 2^{5T(n)}$. These adjustments do not affect M's computation, we have simply defined M to be more TIME and SPACE inefficient.

M's grid of images are represented by a single binary word $\mathcal{G}$. The word $\mathcal{G}$ is composed of $G(n)$ *image subwords* of equal length such that for each image i in M there is an image subword $\mathcal{G}_i$, and vice versa. Specifically $\mathcal{G} = \mathcal{G}_0\mathcal{G}_1 \cdots \mathcal{G}_{G(n)-1}$. We order M's images first horizontally and then vertically; beginning with the lower leftmost grid image, proceeding left to right and then bottom to top.

Next we show how each pixel in image i is represented in the image subword $\mathcal{G}_i$. This representation scheme is analogous to the previous representation of images as subwords. The image subword $\mathcal{G}_i$ is composed of $R_{\mathrm{S}}(n)$ *pixel subwords* of equal length, $\mathcal{G}_i = \mathcal{G}_i[0]\mathcal{G}_i[1] \cdots \mathcal{G}_i[R_{\mathrm{S}}(n)-1]$. For each pixel j in image i there is a pixel subword $\mathcal{G}_i[j]$ and vice versa. Analogous to the image ordering, we order the pixels in each image first by the horizontal and then by the vertical direction, beginning with the lower leftmost pixel.

Given DYRANGE of $R_{\mathrm{D}}(n)$ it follows directly that the value (or range) of each pixel in a $\mathcal{C}_2$-CSM configuration is from the set $\{0, \pm\frac{1}{2}, \pm 1, \pm\frac{3}{2}, \ldots, \pm R_{\mathrm{D}}(n)\}$. To represent this set as a set of binary words we use the 2's complement binary representation of integers, with a slight modification: the binary sequence is shifted by one place to take care of the halves.

At this point we have defined the entire structure of the word $\mathcal{G}$ representing M's grid. The length of $\mathcal{G}$ is $|\mathcal{G}| = R_{\mathrm{S}}(n)G(n)\lceil\log(4R_{\mathrm{D}}(n)+1)\rceil$. Hence if M's SPACE complexity is $O(S(n))$ then $|\mathcal{G}|$ is also $O(S(n))$. We substitute to define $|\mathcal{G}|$ in terms of TIME, $|\mathcal{G}| = 2^{2T(n)}2^{2T(n)}\lceil\log(4\cdot 2^{T(n)}+1)\rceil$. Specifically the length of each pixel subword is $|\mathcal{G}_i[j]| = \lceil\log(4\cdot 2^{T(n)}+1)\rceil$ and the length of each image subword is $|\mathcal{G}_i| = 2^{2T(n)}\lceil\log(4\cdot 2^{T(n)}+1)\rceil$. These expressions are useful for giving bounds on circuit complexity in terms of TIME $T(n)$.

A $\mathcal{C}_2$-CSM configuration $\langle c, e\rangle$ is represented as a binary word ctrl$\mathcal{G}$, which we write as $(\mathrm{ctrl}, \mathcal{G})$, where ctrl is of length $2T(n)$ and $\mathcal{G}$ is as given above. (We interpret the 'instruction pointer' ctrl as a number that indexes the location of the next $\mathcal{C}_2$-CSM operation).

4 Circuit simulation of $\mathcal{C}_2$-CSM

To simulate the $T(n)$ computation steps of the $\mathcal{C}_2$-CSM M we will design a logspace uniform circuit c_M that is of size $S^{O(1)}(n)$ and depth $O(T^2(n))$. We assume that M is a language deciding $\mathcal{C}_2$-CSM [23, 20], and that the input word w is of length n. M is simulated by the circuit c_M in the following way. At the first step of the simulation the circuit c_M is presented with the input word $(\mathrm{ctrl}_{\mathrm{sta}}, \mathcal{G}_{\mathrm{sta}})$ that represents M's initial configuration (including M's input). The circuit c_M has $T(n)$ identical layers numbered 0 (the input layer) to $T(n)-1$ (the output layer). A layer contains a circuit c_{op} for each $\mathcal{C}_2$-CSM operation op, where the circuit encoding function $1^n \rightarrow \overline{c}_{op}$ is computable by a transducer Turing machine in workspace $\log \mathrm{size}(\overline{c}_{op})$.

4.1 Circuits computing $+$, $\cdot$, $*$, ρ, h and v

We begin by simulating the $+$ operation. Addition of two nonnegative integers written in binary is computable by an unbounded fan-in circuit of size $O(m^2)$ and constant depth and so is an AC^0 problem (e.g. use the well-known carry look-ahead algorithm, for example see [8]). Hence this problem is also in NC^1. Krapchenko [9], and Ladner and Fischer [10] give tighter NC^1 adders that have depth $O(\log m)$ and lower the size bound to $O(m)$.

Theorem 1 (circuit simulation of $+$). *The $\mathcal{C}_2$-CSM operation $+$ is simulated by a logspace uniform circuit $c_{a:=a+b}$ of size $O(2^{2T(n)}T(n))$ and depth $O(\log T(n))$.*

Proof. Circuit: To add two pixel words we use Ladner and Fischer's NC^1 addition algorithm [10]. Extending this algorithm to work for the 2's complement binary representation is straightforward. Recall that the addition operation adds images a and b in a parallel point by point fashion and places the result in a. The circuit $c_{a:=a+b}$ has one adder subcircuit for each pixel j in a. Under the representation scheme described above, pixel word $\mathcal{G}_a[j]$ is added to pixel word $\mathcal{G}_b[j]$, resulting in a new word $\mathcal{G}_{a'}[j]$. There are $R_\mathrm{S} = 2^{2T(n)}$ pixels in each of a and b thus the circuit $c_{a:=a+b}$ consists of $2^{2T(n)}$ parallel adders. Each adder has depth $O(\log p)$ and size $O(p)$ where $p = \lceil \log(4 \cdot 2^{T(n)} + 1) \rceil$ is pixel word length. Since each adder has size $O(T(n))$, the circuit has size $O(2^{2T(n)}T(n))$. The circuit has depth $O(\log p) = O(\log T(n))$.

Uniformity: We show that $1^n \rightarrow \overline{c}_{a:=a+b}$ is computable by a transducer that uses at most log workspace. At any computation step the transducer will have at most the encoding of the current gate and a constant number of counters on its worktapes. The circuit has $O(2^{2T(n)}T(n))$ gates, thus each gate label has length $O(T(n))$ and is computable in space $\log |\overline{c}_{a:=a+b}| = O(T(n))$. There are counters for the index of the current gate and for the current adder circuit. Each adder circuit is logspace uniform and hence a constant number of [length $O(T(n))$] counters, is sufficient to construct each one. All gates and counters are computable in space $\log |\overline{c}_{a:=a+b}|$. $\square$

As with addition there are NC^1 circuits for multiplication of binary numbers [8], for example Schönage and Strassen's circuit [16], which uses the DFT, has size $O(m \log m \log\log m)$ and depth $O(\log m)$. Unlike addition, Furst, Saxe and Sipser [5] showed that multiplication is not in AC^0, by showing that parity is not in AC^0 [2]. The proof of the following theorem is identical to the previous one except that we use a polynomial sized NC^1 multiplication circuit as opposed to the linear sized NC^1 adder used above.

Theorem 2 (circuit simulation of $\cdot$). *The $\mathcal{C}_2$-CSM operation $\cdot$ is simulated by a logspace uniform circuit $c_{a:=a\cdot b}$ of size $O(2^{2T(n)}T^2(n))$ and depth $O(\log T(n))$.*

Each pixel in a $\mathcal{C}_2$-CSM is rational valued, hence the complex conjugation operation $*$ is the identity and is simulated by the identity circuit $(\neg\neg a)$.

Theorem 3 (circuit simulation of $*$). *The* $\mathcal{C}_2$-CSM *operation $*$ is simulated by a logspace uniform circuit $c_{a:=a^*}$ of size $O(2^{2T(n)})$ and constant depth.*

Theorem 4 (circuit simulation of ρ). *The* $\mathcal{C}_2$-CSM *operation ρ is simulated by a logspace uniform circuit $c_{a:=\rho(a,z_1,z_u)}$ of size $O(2^{2T(n)}T^2(n))$ and depth $O(\log T(n))$.*

Proof. Circuit: First we build a circuit $c_{u>v}$ that tests if the number represented by one pixel word is greater than another. It is straightforward to give constant size and depth circuits that tell if two bits $b_1, b' \in \{0,1\}$ are equal and if one is greater than the other: The circuits $c_{b\equiv b'}$ and $c_{b>b'}$ respectively compute the $\equiv$ and $>$ expressions $b \equiv b' = (b \wedge b') \vee (\neg b \wedge \neg b')$ and $b > b' = b \wedge \neg b'$. On pixel words u and v, we define the Boolean expression $u > v = \bigvee_{m=0}^{|u-1|}((u_m > v_m) \wedge (\bigwedge_{k=0}^{m-1}(u_k \equiv v_k)))$. We build $c_{u>v}$ as follows. For each m the circuit computing $\bigwedge_{k=0}^{m-1}(u_k \equiv v_k)$ is realised as a $O(m)$ size, $O(\log m)$ depth tree. There are $|u|$ such trees, the root of each has a $\wedge$ test with the constant depth circuit for $u_m > v_m$. We take the OR of these ANDs using a $\log|u|$ depth OR tree. The circuit $c_{u>v}$ has size $O(|u|^2)$ and depth $O(\log|u|)$.

Using a similar construction we build the circuit $c_{u<v}$, this has the same complexity as $c_{u>v}$. Moreover these circuits can be extended to work on the 2's complement representation with only a multiplicative constant increase in size and additive constant increase in depth. We combine these circuits to create the pixel thresholding circuit $c_{v:=\rho(v,l,u)}$ that evaluates to l if $c_{v<l} \equiv 1$, to u if $c_{v>u} \equiv 1$, and to v otherwise.

Recall that the operation $\rho(a, z_l, z_u)$ thresholds image a in a parallel point by point fashion and places the result in a. The circuit $c_{a:=\rho(a,z_l,z_u)}$ has one pixel-thresholding subcircuit for each pixel j in a. Pixel word $\mathcal{G}_a[j]$ is thresholded below by pixel word $\mathcal{G}_{z_l}[j]$ and above by pixel word $\mathcal{G}_{z_u}[j]$ resulting in a new word $\mathcal{G}_{a'}[j]$. The circuit $c_{a:=\rho(a,z_l,z_u)}$ consists of $2^{2T(n)}$ parallel pixel thresholding subcircuits. Each pixel thresholding subcircuit has $O(\log p)$ depth and size $O(p^2)$ where $p = \lceil\log(4\cdot 2^{T(n)}+1)\rceil$ is pixel word length. Since there are $2^{2T(n)}$ pixel thresholding subcircuits (each has size $O(T^2(n))$) the circuit has size $O(2^{2T(n)}T^2(n))$. The entire circuit has depth $O(\log p) = O\left(\log\lceil\log(4\cdot 2^{T(n)}+1)\rceil\right) = O(\log T(n))$.

Uniformity: By stepping through the construction and applying the arguments given in Theorem 1 it is seen that the length of each gate label is $O(T(n))$ and a constant number of variables is sufficient to construct the encoding. □

Next we give simulations of the DFT operations via logspace uniform fast Fourier transform (FFT) circuits. On input length m the FFT circuit has size bounded above by $2m\log m$ and depth bounded above by $2\log m$ [15].

Theorem 5 (circuit simulation of h and v). *The* $\mathcal{C}_2$-CSM *horizontal (respectively, vertical) DFT operation h (respectively, v) is simulated by a logspace uniform circuit $c_{a:=h(a)}$ of size $O(2^{2T(n)}T(n))$ and depth $O(T(n))$.*

Proof. For each row of pixel words in image word a, the circuit $c_{a:=h(a)}$ has a single FFT subcircuit. The output is an image word a' such that each row in a' is the DFT of the same row in a. It is straightforward to verify the size, depth and uniformity. The circuit $c_{a:=v(a)}$ that simulates the $\mathcal{C}_2$-CSM vertical DFT operation v is constructed similarly to $c_{a:=h(a)}$, except we replace the word "row" with "column". □

4.2 Circuits computing *ld* and *st*

The simulations of ld and st are much more involved than the previous constructions. We briefly sketch the simulations, details are to be found in [20].

Theorem 6 (circuit simulation of ld by $c_{a:=[(\xi_1',\eta_1'),(\xi_2',\eta_2')]}$). *The $\mathcal{C}_2$-CSM operation $ld\,(\xi_1',\,\eta_1',\,\xi_2',\,\eta_1')$ is simulated by the uniform circuit $c_{a:=[(\xi_1',\eta_1'),(\xi_2',\eta_2')]}$ of size $O(2^{6T(n)}T(n))$ and depth $O(T(n))$. This circuit takes as input the image words ξ_1', η_1', ξ_2' and η_2', and outputs an image word a.*

Proof (sketch). The address image words ξ_1', η_1', ξ_2', η_2' are decoded into four binary number words by four circuits that respectively compute $\mathfrak{E}^{-1}_{2^{T(n)}}(\xi_1') = \xi_1$, $\mathfrak{E}^{-1}_{2^{T(n)}}(\eta_1') = \eta_1$, $\mathfrak{E}^{-1}_{2^{T(n)}}(\xi_2') = \xi_2$, $\mathfrak{E}^{-1}_{2^{T(n)}}(\eta_2') = \eta_2$, where $\mathfrak{E}^{-1}_{2^{T(n)}}$ is a function that encodes the inverse of the logspace computable address encoding function $\mathfrak{E}$ from Definition 1.

In the next step we wish to select the 'rectangle' of image words that are to be loaded to image word a. Suppose that the rectangle contains more than one image, then naïvely copying the entire rectangle to the image word a would cause problems. (The rectangle to be loaded may contain up to $2^{4T(n)} = R_{\mathrm{S}}(n)G(n)$ pixel words whereas the image word a should contain exactly $2^{2T(n)}$ pixel words). However, observe that spatialRes is bounded above by $2^{2T(n)}$, thus the rectangle to be loaded contains at most $2^{2T(n)}$ regions of distinct value (otherwise image a would contain $> 2^{2T(n)}$ pixels after execution of ld). Hence we have to select only $2^{2T(n)}$ representative pixel words out of the total of $2^{2T(n)}(\xi_2 - \xi_1 + 1)(\eta_2 - \eta_1 + 1) \leqslant 2^{4T(n)}$ pixels. We choose the pixel with the lowest index in each (possibly) distinct area. For each pixel word i in image word a, the pixel j to be loaded is defined as $j = \mathrm{col}(i) + \mathrm{row}(i)R_{\mathrm{S}_x}G_x$ where $\mathrm{col}(i) = (i \bmod R_{\mathrm{S}_x})(\xi_2 - \xi_1 + 1) + R_{\mathrm{S}_x}\xi_1$ and $\mathrm{row}(i) = \lfloor \frac{i}{R_{\mathrm{S}_x}} \rfloor (\eta_2 - \eta_1 + 1) + R_{\mathrm{S}_y}\eta_1$ and as usual $R_{\mathrm{S}_y} = R_{\mathrm{S}_y} = G_x = 2^{T(n)}$.

For each i, a circuit computes the binary number j, the index of the pixel we want to 'load', j is then passed to another circuit which selects pixel word j from the grid word $\mathcal{G}$. The output of the i^{th} circuit represents the i^{th} pixel in image a after a ld operation. □

So far each simulated operation affects only image a. The simulation of st differs in that a *rectangle* of images defined by the coordinates (ξ_1, η_1) and (ξ_2, η_2) is affected.

Theorem 7 (circuit simulation of st by $c_{[(\xi_1',\eta_1'),(\xi_2',\eta_2')]:=a}$). *The $\mathcal{C}_2$-CSM operation $st\,(\xi_1',\,\eta_1',\,\xi_2',\,\eta_1')$ is simulated by a logspace uniform circuit $c_{[(\xi_1',\eta_1'),(\xi_2',\eta_2')]:=a}$*

of size $O(2^{12T(n)}T^6(n))$ and depth $O(T(n))$. This circuit takes as input the image words ξ_1', η_1', ξ_2' and η_2'. It outputs a word of length $|\mathcal{G}|$ that contains the rectangle (defined by $(\xi_1',\eta_1'),(\xi_2',\eta_2')$) of image words to be stored and zeros at all other positions.

Proof (sketch). Let i be the index of a pixel word in image word a. For each i, the index j of each pixel word that will be stored to, satisfies $j = (\mathrm{col}(i) + u) + (\mathrm{row}(i) + v)R_{\mathrm{s}_x}G_x$ for $0 \leqslant u \leqslant \xi_2 - \xi_1$ and $0 \leqslant v \leqslant \eta_2 - \eta_1$, where $R_{\mathrm{s}_x} = G_x = 2^{T(n)}$ and $\mathrm{col}(i)$ and $\mathrm{row}(i)$ were given in the proof of Theorem 6.

The address image words ξ_1', η_1', ξ_2', η_2' are decoded into the binary numbers ξ_1, η_1, ξ_2, η_2 (as in Theorem 6). Then for each $i \in \{0,\ldots,2^{2T(n)}\}$ there is a circuit $c_{st\mathrm{Pixel}(i)}$. The circuit $c_{st\mathrm{Pixel}(i)}$ consists of $2^{4T(n)}$ subcircuits, each tests $j = j'$ for a unique pixel word j' in $\mathcal{G}$. Essentially $c_{st\mathrm{Pixel}(i)}$ generates a 'mask' word m, $|m| = 2^{4T(n)}$. For the given i, mask m has the property that $m_{j'} = 1$ if and only if j' is the index of a pixel word that is to be overwritten with pixel word i from image word a.

Next, the circuit $c_{[(\xi_1',\eta_1'),(\xi_2',\eta_2')]:=a}$ uses i subcircuits as follows. For each pixel word i in image word a: Subcircuit i ANDs the k^{th} symbol in pixel word i with each of the $2^{4T(n)}$ outputs of $c_{st\mathrm{Pixel}(i)}$. At this point we have i grid words; the i^{th} grid word is 0 everywhere except for the 'rectangular' part of the grid that pixel i is stored to. These i grid words are ORed using an OR tree, giving the final output grid word as defined in the theorem statement. This final output word is a mask that contains the 'stored rectangle' and all other pixel words contain only zeros. □

4.3 Control flow and main results

$\mathcal{C}_2$-CSM control flow is straightforward to simulate. Recall from Section 3 that the binary word ctrl represents the $\mathcal{C}_2$-CSM control (or instruction pointer). Simulating *br* involves finding a new value for ctrl from the *br* parameters.

Theorem 8 (circuit simulation of *br* by $c_{br\,(\xi',\eta')}$). *The $\mathcal{C}_2$-CSM branch operation $br\,(\xi',\eta')$ is simulated by a logspace uniform circuit $c_{br\,(\xi',\eta')}$ of size $O(2^{2T(n)}T(n))$ and depth $O(T(n))$.*

Proof. The circuit $c_{br\,(\xi',\eta')}$ decodes its address image word parameters into the binary numbers ξ and η (as in Theorem 6) which are then translated into an image word index i by evaluating $\xi + \eta G_x = i$. Index i points to the image word that encodes the next operation to be executed. □

Let $C_{(i)}$ be an arbitrary $\mathcal{C}_2$-CSM configuration and let $\vdash_M$ be a relation on configurations that defines the operational semantics of $\mathcal{C}_2$-CSM M [23, 20]. The configuration $C_{(i)}$ is encoded as $(\mathrm{ctrl}_{(i)}, \mathcal{G}_{(i)})$ as described in Section 3.

Theorem 9 (circuit simulation of $C_{(i)} \vdash_M C_{(i+1)}$ by c_{step}). *Let M be a $\mathcal{C}_2$-CSM. The uniform circuit c_{step} simulates $C_{(i)} \vdash_M C_{(i+1)}$ and is of size $O(2^{12T(n)}T^6(n))$ and depth $O(T(n))$.*

Proof (sketch). The circuit c_{step} computes $(\text{ctrl}_{(i)}, \mathcal{G}_{(i)}) \to (\text{ctrl}_{(i+1)}, \mathcal{G}_{(i+1)})$. A control flow simulating circuit updates $\text{ctrl}_{(i)}$ using either $c_{br\,(\xi',\eta')}$ or a circuit that simulates sequential control flow by incrementing $\text{ctrl}_{(i)}$ by one of $\{0,1,3,5\}$ depending on the current operation. Another circuit updates $\mathcal{G}_{(i)}$ by making use of the circuits that were given earlier and special mask words to simulate whatever $\mathcal{C}_2$-CSM operation is pointed to by $\text{ctrl}_{(i)}$. □

Next we give the resource use for our circuit simulation of a $\mathcal{C}_2$-CSM.

Theorem 10 (circuit simulation of M by c_M). *Let M be a $\mathcal{C}_2$-CSM that computes for* TIME *$T(n)$. The logspace uniform circuit c_M simulates M and is of size $O(2^{12T(n)}T^7(n))$ and depth $O(T^2(n))$.*

Proof. Circuit c_M is the composition of $T(n)$ instances of c_{step} from Theorem 9. The circuit is given the initial configuration $(\text{ctrl}_{\text{sta}}, \mathcal{G}_{\text{sta}})$ of M as input. After $O(T^2(n))$ parallel timesteps c_M outputs a word representing the final configuration of M. □

The size bound in the previous theorem seems quite high, however one should keep in mind that M has SPACE of $S(n) = O(2^{3T(n)})$. (This was the original SPACE bound on M before we redefined SPACE to suit our simulations).

If M is a language deciding $\mathcal{C}_2$-CSM [20, 23] we augment c_M so that it ORs the bits of the relevant output image word, thus computing a $\{0,1\}$-valued function. The resulting circuit has only a constant factor overhead in the size and depth of c_M. From this we state:

Corollary 1. $\mathcal{C}_2$-CSM-TIME$(T(n)) \subseteq$ U-SIZE,DEPTH$(O(2^{12T(n)}T^7(n)), O(T^2(n)))$

Let $T(n), S(n) = \Omega(\log n)$. From the inclusion given by Equation (1) we state:

Corollary 2. $\mathcal{C}_2$-CSM-TIME$(T(n)) \subseteq$ DSPACE$(O(T^2(n)))$

Combining the above result with the converse inclusion (given in [20, 22]) gives a relationship between nondeterministic (sequential) space, $\mathcal{C}_2$-CSM TIME and deterministic space.

Corollary 3. NSPACE$(S(n)) \subseteq \mathcal{C}_2$-CSM-TIME$(O(S(n) + \log n)^4)$
$\subseteq$ DSPACE$(O(S(n) + \log n)^8))$

To summarise, the $\mathcal{C}_2$-CSM satisfies the parallel computation thesis:

Corollary 4. NSPACE$(S^{O(1)}(n)) = \mathcal{C}_2$-CSM-TIME$(S^{O(1)}(n))$

This establishes a link between space bounded sequential computation and TIME bounded $\mathcal{C}_2$-CSM computation, e.g. $\mathcal{C}_2$-CSM-TIME$(n^{O(1)}) = \mathsf{PSPACE}$.

The thesis relates parallel time to sequential space, however in our simulations we explicitly gave *all* resource bounds. As a final result we show that the class of $\mathcal{C}_2$-CSMs that simultaneously use polynomial SPACE and polylogarithmic TIME decide at most the languages in NC. Let $\mathcal{C}_2$-CSM-SPACE, TIME$(S(n), T(n))$ be the class of languages decided by $\mathcal{C}_2$-CSMs that use SPACE $S(n)$ and TIME $T(n)$.

For uniform circuits it is known [8] that $\text{U-SIZE}, \text{DEPTH}(n^{O(1)}, \log^{O(1)} n) = \mathsf{NC}$. From the resource overheads in our simulations:

$$\mathcal{C}_2\text{-CSM-SPACE}, \text{TIME}(2^{O(T(n))}, T(n)) \subseteq \text{U-SIZE}, \text{DEPTH}(2^{O(T(n))}, T^{O(1)}(n)).$$

For the case of $T(n) = \log^{O(1)} n$ we have our final result.

Corollary 5. $\mathcal{C}_2\text{-CSM-SPACE}, \text{TIME}(n^{O(1)}, \log^{O(1)} n) \subseteq \mathsf{NC}$

In [20, 22] it was shown that the converse inclusion also holds. Thus $\mathcal{C}_2$-CSMs that simultaneously use both polynomial SPACE and polylogarithmic TIME exactly characterise NC.

5 Discussion

We have given upper bounds on $\mathcal{C}_2$-CSM power in terms of uniform circuits. Combining this with previously shown lower bounds [22]; the $\mathcal{C}_2$-CSM verifies the parallel computation thesis and $\mathcal{C}_2$-CSMs with polynomial SPACE and polylogarithmic TIME decide exactly the languages in NC.

Our simulations could probably be improved to get a tighter relationship between $\mathcal{C}_2$-CSM TIME and sequential space. For example, the size bounds on the circuit simulation of *ld* and *st* could be improved with the aim of reducing the degree of the polynomials in Corollary 3, maybe even to a quadratic. Any improvement beyond that would be difficult, since it would imply an improvement to the quadratic bound in Savitch's theorem. Such improvements would enable us define a tighter bound on simultaneous resource usage between the $\mathcal{C}_2$-CSM and (say) uniform circuits, in the hope of exactly characterising NC^k by varying a parameter k to the model.

By how much can we generalise the $\mathcal{C}_2$-CSM definition and still preserve the upper bounds presented here? For example, in analogy with Simon's result for vector machines [17], we believe it should be possible to remove the $O(2^{3t})$ SPACE restriction from the $\mathcal{C}_2$-CSM definition.

Our results are the first to establish a general relationship (upper and lower bounds) between optically inspired computation and standard complexity classes. On a related note, our results show that the kind of optics modelled by the $\mathcal{C}_2$-CSM is simulated in reasonable time on any of the parallel architectures that verify the parallel computation thesis.

References

1. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity, vols I and II.* EATCS Monographs on Theoretical Computer Science. Springer, Berlin, 1988.
2. R. B. Boppana and M. Sipser. *The complexity of finite functions.* Volume A of van Leeuwen [19], 1990.
3. A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, Dec. 1977.

4. A. K. Chandra and L. J. Stockmeyer. Alternation. In *FOCS 1976*, pages 98–108, Houston, Texas, Oct. IEEE.
5. M. L. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
6. L. M. Goldschlager. *Synchronous parallel computation.* PhD thesis, University of Toronto, Computer Science Department, Dec. 1977.
7. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness theory.* Oxford university Press, Oxford, 1995.
8. R. M. Karp and V. Ramachandran. *Parallel algorithms for shared memory machines.* Volume A of van Leeuwen [19], 1990.
9. V. Krapchenko. Asymptotic estimation of addition time of a parallel adder. *Syst. Theory Res.*, 19:105–222, 1970.
10. R. E. Ladner and M. J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, 1980.
11. T. J. Naughton. Continuous-space model of computation is Turing universal. In *Critical Technologies for the Future of Computing*, Proc. SPIE vol. 4109, pages 121–128, Aug. 2000.
12. T. J. Naughton. A model of computation for Fourier optical processors. In *Optics in Computing 2000*, Proc. SPIE vol. 4089, pages 24–34, Quebec, June 2000.
13. T. J. Naughton and D. Woods. On the computational power of a continuous-space optical model of computation. In *Machines, Computations and Universality: 3*rd *Int. Conference*, volume 2055 of *LNCS*, pages 288–299, Chişinău, 2001. Springer.
14. I. Parberry. *Parallel complexity theory.* Wiley, 1987.
15. J. E. Savage. *Models of computation: Exploring the power of computing.* Addison Wesley, 1998.
16. A. Schönhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7(3-4):281–292, 1971.
17. J. Simon. On feasible numbers. In *Proc. 9th Annual ACM Symposium on Theory of Computing*, pages 195–207. ACM, 1977.
18. P. van Emde Boas. *Machine models and simulations.* Volume A of van Leeuwen [19], 1990.
19. J. van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume A. Elsevier, Amsterdam, 1990.
20. D. Woods. *Computational complexity of an optical model of computation.* PhD thesis, National University of Ireland, Maynooth, 2005.
21. D. Woods and J. P. Gibson. Complexity of continuous space machine operations. In *New Computational Paradigms, First Conference on Computability in Europe*, volume 3526 of *LNCS*, pages 540–551, Amsterdam, June 2005. Springer.
22. D. Woods and J. P. Gibson. Lower bounds on the computational power of an optical model of computation. In *Fourth International Conference on Unconventional Computation*, volume 3699 of *LNCS*, pages 237–250, Sevilla, Oct. 2005. Springer.
23. D. Woods and T. J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1–3):227–258, Apr. 2005.