On the Time Required to Perform Addition

S. WINOGRAD

IBM Corporation, Yorktown Heights, New York*

Abstract. The time required to perform a group operation using logical circuitry is investigated. A lower bound on this time is derived, and in the case that the group is abelian it is shown that the lower bound can be approached as the complexity of the elements used increases. In particular, if the group operation is adding integers modulo μ, it is shown that the lower bound behaves as log log $\alpha(\mu)$, where $\alpha(\mu)$ is the largest power of a prime which divides μ.

I. *Introduction*

A study is made in this paper of the amount of time required to add two integers using logical circuitry. Each of the addends ranges over the set $Z_N = \{0, 1, \cdots, N-1\}$, and the addition is performed modulo N. This problem was considered by U. Ofman [1], who studied it in the case that $N = 2^n$; and the input signals representation of the addends as well as the output signals representation of the sum correspond to the radix 2 representation of integers. He showed that under these conditions the addition can be performed requiring an amount of time which grows as $\log_2 \log_2 N = \log_2 n$; while the amount of hardware required grows as $\log_2 N = n$. Another addition scheme which also requires an amount of time which grows as $\log_2 n$, under the above mentioned conditions, was reported in [2]. An evaluation of the relative merits of various schemes for addition can be found in [6].

In this paper only the amount of time required to perform addition is investigated, disregarding the amount of hardware required for the implementation of the addition schemes. A lower bound on the amount of time is derived, and it is shown that this lower bound can be approached as the complexity of the logical elements used increases.

In Section II, a more precise formulation of the problem is to finite abelian groups. It is then shown that for finite abelian groups the lower bound can be approached as the complexity of the logical elements used increases.

In Section II, a more precise formulation of the problem is given; the lower bound is derived in Section III; and Section IV describes schemes which require an amount of time which is close to the lower bound.

II. *Formulation of the Problem*

This section is devoted to giving a more precise definition of a logical circuit, as well as its ability to add integers or perform any other group operation in a given amount of time.

Definition 1. A *d-values logical element* is an object with r input lines and one output line. The output line and each of the input lines can be in one of d distinct states. The state of the output line at time $t + 1$, is a function of the states of the input lines at time t.

* Thomas J. Watson Research Center.

Journal of the Association for Computing Machinery, Vol. 12, No. 2 (April, 1965), pp. 277–285

Definition 2. A *d-values logical circuit* is a finite set of d-values logical elements and a rule of interconnections which partitions the set of lines of the logical elements and identifies (connects) two lines if and only if they are in the same class. This rule of interconnections is subject to the restriction that no two output lines are to be in the same class. The classes which do not include an output line are called the *inputs of the circuit*, and a designated set of classes which do include an output line is called the *outputs of the circuit*.

Let C be a logical circuit. We designate by S_c the (finite) set of all possible configurations of the states of the output lines of the logical elements of C (S_c is termed the set of *internal states*); we designate by I_c the (finite) set of all possible configurations of states of the inputs of C (I_c is termed the set of *input states*), and we designate by O_c the (finite) set of all possible configurations of the states of the outputs of C (O_c is termed the set of *output states*). Note that since the outputs of C are a subset of the set of output lines of elements of C, the output state is determined by the internal state, we use $o: S_c \to O_c$ to designate this function.

The *behavior* of a logical circuit C is described in terms of a function $f: S_c \times I_c \to S_c$, where $f(s, i)$ $(s \in S_c, i \in I_c)$ is determined by setting the state of each input line of an element of C at time t equal to the state of the output line or input of the circuit with which it is associated, as determined by s and i, and taking the resulting state of the circuit at time $t + 1$ as $f(s, i)$. The function $f(s, i)$ can be extended to $f^*: S_c \times I_c \times N \to S_c$ where N is the set of natural integers, and where $f^*(s, i, 1) = f(s, i)$ and $f^*(s, i, n + 1) = f(f^*(s, i, n), i)$. Finally we define $c: S_c \times I_c \times N$ by $c(s, i, n) = o(f^*(s, i, n))$.

Definition 3. Let $\varphi: \prod_{i=1}^{t} X_i \to Y$ be any finite function. A *circuit C is said to be capable of computing the function φ in time τ*, if there exists a state $s_0 \in S_c$, a partition of the inputs of C into t equivalence classes, a set of t functions $g_j : X_j \to I_{c,j}$ $(j = 1, 2, \cdots, t)$ where $I_{c,j}$ is the set of configurations of states of the inputs of the jth equivalence class, a function $h: Y \to O_c$ such that for each $(x_1, x_2, \cdots, x_t) \in X_1 \times X_2 \cdots \times X_t c(s_0, (g_1(x_1), g_2(x_2), \cdots, g_t(x_t)), \tau) = h(\varphi(x_1, x_2, \cdots, x_t))$. We also say that a *circuit C is capable of computing the function φ* if there exists an integer τ such that C is capable of computing φ in time τ.

The notion of a circuit computing a function as described in definition 3 essentially consists of the following.

(i) The internal state of the circuit is set to s_0 at time 0.

(ii) Each of the arguments of the functions is coded and applied as inputs to the circuit.

(iii) The inputs are held fixed until time τ, in which time the output of the circuit is "sampled" to yield the result of the computation (in a coded form).

In Section III we derive a lower bound on the time required to compute the function $\varphi: G \times G \to G$, where G is a finite group, and φ is the group operation.

III. *The Lower Bound*

Let C be a logical circuit. We use $c_j(s, i, \tau)$ to denote the jth component of $c(s, i, \tau)$, i.e. if the state of output o_j is the jth component of the output state, then $c_j(s, i, \tau)$ is the state of o_j at time τ if the circuit "started" in internal state s and the input is held fixed as i.

Definition 4. An output line o_j is said to be s *independent* ($s \in S_c$) *on input line* l_k *in time* τ, if for all $i_1 \in I_c$ and $i_2 \in I_c$ differing only in the state of l_k $c_j(s, i_1, \tau) \neq c_j(s, i_2, \tau)$. If an output line is not s-independent on line l_k in time τ, it is said to be s-*dependent on* l_k *in time* τ. Note that if output o_j is s-dependent on input l_k there exists a "path" from l_k to o_j if the circuit is viewed as a directed graph.

LEMMA 1. *Let C be a logical circuit; each of whose elements has at most $f(>1)$ input lines. For all output lines o_j of C, if o_j is s_0-dependent on t inputs of C at time τ, then $\tau \geqq [\log_r t]$.* ($[x]$ denotes the smallest integer not smaller than x.)

PROOF. We prove the lemma by induction on t. If $t \leqq r$, the inequality obviously holds.

Suppose o_j depends on $t > k$ inputs of C, and o_j is the output line of element e_j. Let O be the set of input lines of e_j, at least one of those lines is an output of another element of C which depends on at least $[t/r]$ inputs of C at time $\tau - 1$. By induction hypothesis we obtain

$$\tau - 1 \geqq \left[\log_r \left[\frac{t}{r}\right]\right] \geqq \left[\log_r \frac{t}{r}\right] = [\log_r t] - 1,$$

so $\tau \geqq [\log_r t]$.

Let C be a d-values logical circuit which computes $\varphi: G \times G \to G$, where G is a finite group, and φ is the group operation. In the remainder of the section we use $g_1 \cdot g_2$ to denote $\varphi(g_1, g_2)$. Since C computes φ, there exists a function $h: G \to O_c$. We use $h_j(g)$ to denote the state of the jth component of $h(g)$.

Definition 5. An element $a \in G$ is said to *have property* E_j *with respect to a circuit C which computes φ* if for all $b \in gp(a)$, $h_j(b) = h_j(e)$, where $gp(a)$ denotes the group generated by a, and e denotes the identity of G. We denote the fact that $a \in G$ has property E_j by $E_j(a)$.

LEMMA 2. *Suppose $E_j(a)$ does not hold, and $b, c \in G$ are such that $g_1(b)$ and $g_1(c)$ ($g_2(b)$ and $g_2(c)$) differ only in inputs on which o_j does not depend, then $a \notin gp(b \cdot c^{-1})$ ($a \notin gp(b^{-1} \cdot c)$).* Here g_1 and g_2 are as in definition 3.

PROOF. We prove here only one half of the assertion, the other half is provable in an analogous way. By hypothesis, for any element $x \in G$

$$h_j(b \cdot x) = c_j(s_0, (g_1(b), g_2(x)), \tau) = c_j(s_0, (g_1(c), g_2(x)), \tau) = h_j(c \cdot x)$$

(since o_j was independent of the input lines in which $g_1(b)$ and $g_1(c)$ differ). Thus $h_j((b \cdot c^{-1}) \cdot x) = h_j(b \cdot (c^{-1} \cdot x)) = h_j(c \cdot (c^{-1} \cdot x)) = h_j(x)$, for all $x \in G$. Setting $x = e$ we obtain $h_j(b \cdot c^{-1}) = h_j(e)$, setting $x = b \cdot c^{-1}$ we obtain $h_j(e) = h_j(b \cdot c^{-1}) = h_j((b \cdot c^{-1})^2)$, by setting $x = (b \cdot c^{-1})^s$ for all integers s, we obtain that $E_j(b \cdot c^{-1})$ holds. But if $a \in gp(b \cdot c^{-1})$ then $E_j(a)$ would hold as well, so $a \notin gp(b \cdot c^{-1})$.

Definition 6. Let $e \neq a \in H \leqq G$, we say that $P(a, H)$ holds if and only if $\{e\} \neq H_1 \leqq H \Rightarrow a \in H_1$. We say, that property $P(H)$ holds for $H \leqq G$ if either $H = \{e\}$, or there exists $e \neq a \in H$ such that $P(a, H)$ holds.

LEMMA 3. *Let C be a d-values logical circuit which computes $\varphi: G \times G \to G$. Let $e \neq a \in H \leqq G$ be such that $P(a, H)$ holds, and let o_j be such that $h_j(a) \neq h_j(e)$, then o_j depends on at least $[\log_d |H|]$ input lines in each equivalence class of the inputs of C.* ($|H|$ denotes the order of H.)

PROOF. Assume o_j depended only on $q < [\log_d |H|]$ of the inputs lines of the first equivalence class, then there are $b, c \in H$ such that $g_1(b)$ and $g_1(c)$ differ only in input lines on which o_j does not depend. By lemma 2, $a \notin gp(b \cdot c^{-1})$

contradiction. Similar arguments establish the result for the second equivalence class of the input lines.

Let $\alpha(G) = \max \{|H| \mid H \leqq G \text{ and } P(H)\}$. Note that if $G \neq \{e\}$, then $\alpha(G) > 1$, because if $e \neq a \in G$, then let the order of a be $p^i q$ (where q is relatively prime to p); observe that $P(a^{p^{i-1}q}, gp(a^q))$ holds and that $| gp(a^q) | = p^i > 1$.

THEOREM 1. *Let C be a d-values logical circuit, each of whose elements has at most r input lines, which computes $\varphi: G \times G \to G$ in time τ, then if $G \neq \{e\}$, $\tau \geqq [\log_r 2[\log_d \alpha(G)]]$.*

PROOF. Let $H \leqq G$ be such that $P(H)$ holds and that $| H | = \alpha(G)$. Let $a \in H$ be such that $P(a, H)$ hold. There exists o_j such that $h_j(a) \neq h_j(e)$, and therefore, by lemma 3, o_j depends on at least $2[\log_d | H |] = 2[\log_d \alpha(G)]$ input lines of C. Applying Lemma 1 we obtain $\tau \geqq [\log_r 2[\log_d \alpha(G)]]$.

We see, in Theorem 1, that the lower bound on the time required to compute $\varphi: G \times G \to G$ by a logical circuit depends on $\alpha(G)$. It might be of interest to further investigate $\alpha(G)$, and the subgroups H such that $P(H)$ holds.

Remark 1. If H is such that $P(H)$ holds and $b, c \in H$ are of orders p^i, q^j, respectively (where p and q are prime numbers), then $p = q$; for let $e \neq a \in H$ be such that $P(a, H)$ holds, then the order of a divides p^i and q^i. Therefore if $P(H)$ holds, H has to be a p-group, i.e. the order of H as well as the order of each element of H is a power of p for some prime number p.

Remark 2. If $H = H_1 \times H_2$, then $P(H)$ cannot hold, for $P(H)$ holds if and only if $\bigcap_{\{e\} \neq H' \leqq H} H' \neq \{e\}$, but $H = H_1 \times H_2$ if and only if $H_1 \cap H_2 = \{e\}$.

Remark 3. If H is abelian, then $P(H)$ holds if and only if H is a cyclic group of order p^i for some prime number p.

Since any abelian group can be written uniquely as the product of cyclic p-groups, we obtain that if G is abelian. $\alpha(G)$ is the order of the largest cyclic p-subgroup of G.

In particular if $G = Z_\mu = \{0, 1, 2, \cdots, \mu - 1\}$ where the group operation φ is defined by $\varphi(z_1, z_2) = z_1 + z_2 \bmod \mu$, we obtain that $\alpha(G) = \alpha(\mu)$, where $\alpha(\mu)$ is defined as $\alpha(\mu) = \max \{p_i^n \mid p_i \text{ is a prime, } n \text{ is positive integer, and } p_i^n \mid \mu\}$.

Before ending this section we will give two immediate corollaries of Theorem 1.

COROLLARY 1. *If C is a d-values logical circuit, each of whose element has at most r input lines, which computes $\varphi': G \times G \to G$ in time τ, where $\varphi'(a, b) = a \cdot b^{-1}$, then $\tau \geqq [\log_r 2[\log_d \alpha(G)]]$.*

PROOF. Let g_1, g_2 and h be the decoding and encoding functions. Consider the functions $g_1' = g_1$, $g_2'(a) = g_2(a^{-1})$ and $h' = h$. It is clear that with these functions C computes $\varphi: G \times G \to G$ where $\varphi(a, b) = a \cdot b$ in *time* τ, *therefore* $\tau \geqq [\log_r 2[\log_d \alpha(G)]]$.

COROLLARY 2. *Let C be a d-values logical circuit, each of whose elements has at most r inputs, which computes*

$$\varphi_k : \overbrace{G \times G \times \cdots \times G}^{k} \to G,$$

where $\varphi(a_1, a_2, \cdots, a_k) = a_1 \cdot a_2 \cdots a_k$, in time τ, then $\tau \geqq [\log_r k[\log_d \alpha(G)]]$.

PROOF. In case $k = 2$, then we have the statement of Theorem 1. It is clear that Lemmas 2 and 3 hold also for circuits which compute φ_k, then by the same argument used to prove Theorem 1 the result of the corollary follows.

IV. *Computation Scheme for Abelian Groups*

In this section we describe a scheme for computing the group operation for abelian groups. This scheme requires time which is close to the lower bound obtained in the last section, and actually approaches the lower bound as the number of inputs of the elements comprising the circuit increases.

We first describe the scheme for cyclic groups of order p^i using d-values circuits, and finally we consider the general case of any finite abelian group using d-values circuit.

In considering cyclic groups of order μ it will be convenient to describe the group operation as addition modulo μ, and use terms like "a carry." In the case of cyclic groups we "encode" and "decode" the elements of the group using the regular radix d representation of the positive integers.

Before describing the scheme we digress to show that if we use p-values logical circuits to compute φ for a cyclic group of order p^i, then the radix p representation of integers for both arguments and result yields as little dependence of output lines on input lines as possible. In talking about cyclic groups of order p^n we consider $Z_{p^n} = \{0, 1, \cdots, p^n - 1\}$ as the carrier and addition modulo p^n as the group operation. If we use radix p representation of both summands and the result, then one output line depends on n inputs of each summand, two outputs depend on $n - 1$ inputs of each summand, and in general k outputs depend on $n - k + 1$ inputs of each summand.

The following proposition shows the minimality of this dependence.

PROPOSITION. *Let C be a p-values logical circuit which computes $Z_{p^n} + Z_{p^n} \bmod p^n$, then for each $1 \leqq k \leqq n$ there are at least k outputs which depend on at least $n - k + 1$ inputs in each equivalence class.*

PROOF. Consider the set of outputs $O_k = \{o_j | \exists t \cdot p^{n-k} \in Z_{p^n}$ such that $h_j(t \cdot p^{n-k}) \neq h_j(0)\}$. It is clear that since there are $p^k - 1$ elements of Z_{p^n} of the form $t \cdot p^{n-k}$ that the cardinality of O_k is at least k. We show here that each output of O_k depends on at least $n - k + 1$ input lines of each equivalence class and thus establish the result. Consider the set $Q_k = \{0, 1, 2, \cdots, p^{n-k+1} - 1\}$, if $o_j \in O_k$ depended only on $q < n - k + 1$ input lines of the first equivalence class, then there would be two elements $a, b \in Q_k$ such that $g_1(a)$ and $g_1(b)$ will differ only in input lines on which o_j does not depend. But then since $t \cdot p^{n-k} \in gp(a - b)$ we obtain a contradiction to Lemma 2. Similar arguments hold for the second addend.

LEMMA 10. *For each $r \geqq 3$, there exists a d-values logical circuit with elements which have at most r* input lines, which computes $Z_{d^t} + Z_{d^t} \bmod d^t$, *where $t = \langle r/2 \rangle \langle (r+1)/2 \rangle^s$ in time $\tau = s + 1$, where $\langle x \rangle$ denotes the integer part of x.*

PROOF. We prove the theorem by induction on s. *In the proof, the addends as well as the sum are represented by their radix d* representation.

In case $s = 0$, we can represent any number in Z_{d^t} by $\langle r/2 \rangle$ digits, and thus we can compute the sum in time $\tau = 1$ using logical elements with up to r inputs. (Actually only one such element is needed per digit of the output.) Moreover, in time $\tau = 1$ we can determine whether, for any two elements $b, c \in Z_{d^t}$, $b + c \geqq d^t$ or not, i.e. we can generate the carry as well. Notice also, that in this case we could compute $1 + Z_{d^t} + Z_{d^t} \bmod d^t$ as well as generate the carry in time $\tau = 1$.

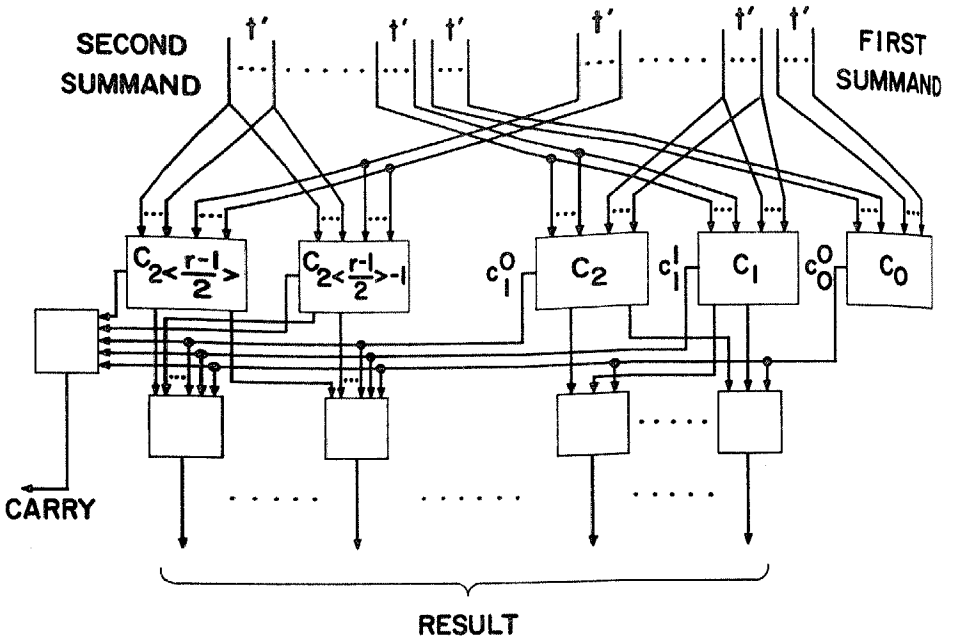

FIG. 1

Assume it is true that for $s = m$ we can compute $Z_{d^t} + Z_{d^t} \bmod d$ and $1 + Z_{d^t} + Z_{d^t} \bmod d^t$ as well as generate the carries in both cases in time $\tau = m + 1$. Let $s = m + 1$. We may build the desired circuit from $2\langle(r + 1)/2\rangle - 1$ circuits each of which will compute $a + Z_{d^t} + Z_{d^t} \bmod d^t$ for $s = m$ ($a = 0$ or 1), and then obtain the desired result by combining the outputs of those circuits in one unit of time, and thus obtain the result in time $\tau = m + 1 + 1 = m + 2 = s + 1$.

Let t' denote $\langle r/2\rangle\langle(r + 1)/2\rangle^m$. Using radix d representation of the elements of d^t (for $s = m + 1$), we can consider the digits of the representation of Z_{d^t} as comprised of $\langle(r + 1)/2\rangle$ batches each of which consists of t' digits, the first batch consisting of the lowest t' digits, the second batch consisting of digits in positions of $d^{2t'-1}$ to $d^{t'}$ etc. Let $C_0, C_1, \cdots, C_{2\langle(r-1)/2\rangle}$ be the $2\langle(r + 1)/2\rangle - 1$ circuits such that C_j will compute $Z_{d^t} + Z_{d^t} \bmod d^t$ if j is even as well as generate a carry, and C_j will compute $1 + Z_{d^t} + Z_{d^t} \bmod d^t$ if j is odd as well as generate a carry. The inputs to circuit C_j are the kth batch of each of the addends, where $k = [j/2] + 1$. It is clear that circuit C_{2k} adds the $(k + 1)$-th batches as if no carry "propagated from the right," while the circuit C_{2k+1} adds the $(k + 1)$-th batches as if a carry did "propagate from the right." Let c_k^0 denote the carry generated by C_{2k} and c_k^1 denote the carry generated by C_{2k-1}. It is clear that $c_0^0, c_1^0, c_1^1, \cdots, c_j^0, c_j^1$ determine whether in adding the $(j + 2)$-th batch (when the actual result of adding modulo d^t is desired) we have a carry "propagated from the right" or not. Thus the number of input lines needed to determine the actual result as well as the carry does not exceed $2\langle(r + 1)/2\rangle - 1 \leqq r$, and therefore requires only one extra unit of time. Thus the time required to compute $Z_{d^t} + Z_{d^t} \bmod d^t$ as well as generate the carry is $m + 2$. By modifying the proof by requiring C_0 to compute $1 + Z_{d^{t'}} + Z_{d^{t'}} \bmod d^{t'}$, we obtain that

$1 + Z_{d^t} + Z_{d^t} \bmod d^t$ as well as generating the carry can also be computed in time $\tau = m + 2$. The scheme used to prove Lemma 4 is based on the scheme reported in [2], and is illustrated in Figure 1.

The following two corollaries are obtained as an easy consequence of Lemma 4.

COROLLARY 3. *For each $r \geqq 3$ there exists a d-values logical circuit, which have at most r input lines to each of its elements, which computes $Z_{d^n} + Z_{d^n} \bmod d^n$ in time $\tau = 1 + [\log_{\langle (r+1)/2 \rangle} (n/\langle r/2 \rangle)]$.*

COROLLARY 4. *For each $r \geqq 3$ and for each $a \in Z_{d^n}$ there exists a d-values circuit, whose elements have at most r input lines, which compute $a + Z_d{}^n + Z_d{}^n \bmod d^n$ in time $\tau = 1 + [\log_{\langle (r+1)/2 \rangle} (n/\langle r/2 \rangle)]$.*

LEMMA 5. *For each $r \geqq 3$, there exists a d-values logical circuit, each of whose elements has at most r input lines, which can compute $Z_{p^n} + Z_{p^n} \bmod p^n$ in time $\tau = 2 + [\log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d p^n]]$.*

PROOF. Let $s = [\log_d p^n]$. Observe that if $a + b < p^n$ then $a + b \bmod p^n = a + b \bmod d^s$, and if $a + b \geqq p^n$ then $a + b + (d^s - p^n) \bmod d^s = a + b \bmod p^n$; moreover $a + b \geqq p^n$ if and only if $a + b + (d^s - p^n) \geqq d^s$. Thus we first construct two adders, one which adds $a + b \bmod d^s$ and one which adds $a + b + (d^s - p^n) \bmod d^s$. By Corollary 4, those two adders can compute their respective functions in time $\tau = 1 + [\log_{\langle (r+1)/2 \rangle} (s/\langle r/2 \rangle)]$. By choosing the output of the appropriate adder according to whether the second one has a carry or not we obtain $a + b \bmod p^n$. Note that the selection of the appropriate output can be performed in one unit of time since $r \geqq 3$. Thus the total time to perform $a + b \bmod p^n$ is

$$\tau = 2 + [\log_{\langle (r+1)/2 \rangle} (s/\langle r/2 \rangle)] = 2 + [\log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d p^n]].$$

THEOREM 2. *Let G be a finite abelian group. For each $r \geqq 3$ there exists a d-values logical circuit, whose elements have at most r input lines, which computes $\varphi: G \times G \to G$ in time*

$$\tau = 2 + [\log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d \alpha(G)]].$$

PROOF. Since G is a finite abelian group, G can be decomposed as $G = G_1 \times G_2 \times \cdots \times G_m$ where each G_i is a cyclic group of order a power of a prime. Thus we can construct m circuits which compute $\varphi_i : G_i \times G_i \to G_i$, each of which will require $\tau_i = 2 + [\log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d | G_i |]]$ to compute its function. Thus the whole circuit can compute its function in time

$$\tau = \max_{1 \leqq i \leqq m} \tau_i = 2 + [\log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d \alpha(G)]].$$

(The idea of the proof is the same as the scheme of [3].)

Comparison of the result of Theorem 2, with those of Theorem 1, shows that as r increases the time required to compute $\varphi: G \times G \to G$ for finite abelian groups approaches the lower bound. More specifically, let τ_{act} denote the time obtained in Theorem 2, and $\tau_{\min}$ denote the lower bound obtained in Theorem 1, then:

$$\tau_{\text{act}} = 2 + [\log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d \alpha(G)]]$$

$$\sim 2 + \log_{\langle (r+1)/2 \rangle} (1/\langle r/2 \rangle) [\log_d \alpha(G)]$$

$$\sim 2 - \log_{\langle (r+1)/2 \rangle} \langle r/2 \rangle + [\log_r \log_d \alpha(G)]/\log_{\langle (r+1)/2 \rangle} r.$$

So for large enough r such that $\log_{\langle (r+1)/2 \rangle} \langle r/2 \rangle \sim 1$ and $\log_{\langle (r+1)/2 \rangle} r \sim 1$ we obtain $\tau_{act} \sim \tau_{min} + 1$.

V. *Discussion*

In Section III we saw that a lower bound of the amount of time required to compute the group operation of a finite group G depends on the logarithm of the logarithm of the order of a certain p-subgroup of G. In the case that G is an abelian group (and in particular if G is Z_μ where the group operation is addition modulo μ), this p-subgroup is the largest cyclic subgroup with order a power of a prime. In Section II we saw that this lower bound can be approached as the number of input lines to the logical elements used to construct the circuit increases.

These results depend on the particular definition used for "a logical circuit C is capable of computing the function φ." In the definition we required that the inputs of the circuit be partitioned, and each equivalence class would correspond to one argument of the function to be computed. This was done to insure that the function φ will be actually computed by the circuit and therefore that the inputs will carry information on the arguments only and not on the way they are to combine to yield the result. The reader can readily convince himself that if we had replaced the requirement on g_j in definition 3 by requiring the existence of $g_j' : I_{c,j}' \subset I_{c,j} \to X_j$, we would have an equivalent definition.

The requirement on the existence of a function $h: Y \to O_c$ is equivalent to requiring the existence of $h' : O_c' \subset O_c \xrightarrow{1:1} Y$. The reason for h' being 1:1 is to avoid the possibility of having the outputs carry just the same information as the inputs, and having the function h' "actually" perform the computation. Of course there are alternative definitions of computation which will still conform to our intuitive notion. In [4] an addition scheme was described in which the requirement on h' was relaxed, instead it was required that the same code will be used to code each of the addends as well as the result.

Another feature of definition 3 was that there is a fixed time τ when the output is sampled. An alternative approach is to consider the time the circuits "settle down," which will of course depend on the particular values the arguments take, and then consider the average time for the circuit to "settle down" as the computation time (see [5]). It is conjectured that this average time is still the same order of magnitude as the lower bound, but we could not prove this conjecture.

Another avenue of investigation is to consider both the time of computation and the number of logical elements required to construct the circuit (see [1]). This approach might even be coupled with relaxing the assumption that the inputs carry all the information about the arguments at time $\tau = 0$, and allowing the inputs to be fed into the circuit "sequentially."

Acknowledgments. The author is grateful to M. O. Rabin for suggesting the problem of computation time for addition under the restriction of r-inputs components as well as raising the question whether this time can be improved if one does not use radix representation for the numbers. Many thanks are also due to C. C. Elgot for his help in clarifying many points and to J. B. Wright for his illuminating comments.

RECEIVED MAY, 1964

REFERENCES

1. OFMAN, U. On the algorithmic complexity of discrete functions. *Dokladi 145*, 1 (1962), 48–51.
2. MACSORELY, O. L. High speed arithmetic in binary computers. Proc. IRE, 1961, 67–91.
3. GARNER, H. L. The residue number system. *IRE Trans. EC-8*, (June 1959), 140–147.
4. AUIZIENIS, A. Signed sigit number representation for fast parallel arithmetic. *IRE Trans. EC-10* (Sept. 1961), 389–400.
5. GILCHRIST, B., POMERENE, J. H., AND WONG, S. Y. Fast carry logic of digital computers. *IRE Trans. EC-4*, 4 (Dec. 1955), 133–136.
6. SKLANSKY, J. An evaluation of several two-summand binary adders. *IRE Trans. EC-9* (June 1960), 213–226.