Information Processing Letters 35 (1990) 325–328
North-Holland 15 September 1990

ON THE DEPTH COMPLEXITY OF THE COUNTING FUNCTIONS

Andrew CHIN *

Mathematical Institute and Computing Laboratory, University of Oxford, UK

Communicated by F.B. Schneider
Received 18 August 1989
Revised 28 November 1989

We use Karchmer and Wigderson's recent characterization of circuit depth in terms of communication complexity to design shallow Boolean circuits for the counting functions. We show that the MOD_3 counting function on n arguments can be computed by Boolean networks which contain negations and binary OR- and AND-gates in depth $c \log_2 n$, where $c \doteq 2.881$. This is an improvement over the obvious depth upper bound of $3 \log_2 n$. We can also design circuits for the MOD_5 and MOD_{11} functions having depth $3.475 \log_2 n$ and $4.930 \log_2 n$, respectively.

Keywords: Boolean function, circuit depth, complexity, communication complexity, parallel algorithms

1. Introduction

The counting functions $\mathrm{MOD}_{k,r}^{(n)}: \{0, 1\}^n \to \{0, 1\}$ defined by $\mathrm{MOD}_{k,r}^{(n)}(x) = 1$ iff $x_1 + \cdots + x_n = r \bmod k$ been fundamental in the study of Boolean function complexity [3,4,8]. A variety of methods have proved helpful in the construction of short formulas [2,9] and shallow circuits [6] for these functions. In this paper, we show that a recent characterization of circuit depth in terms of communication complexity [5] can be used to design efficient circuits for many of the counting functions.

We will consider circuits over the basis $U_2 = \{\vee, \wedge, \neg\}$. The depth of a U_2-circuit is the maximal number of $\vee$ and $\wedge$ gates in a path from an input gate to the output gate. A "naive" upper bound for the U_2-depth complexity of the counting functions is described by the following

Proposition 1.1. $D_{U_2}(\mathrm{MOD}_{k,r}^{(n)}) \leqslant \lceil 1 + \log_2 k \rceil \cdot \lceil \log_2 n \rceil$.

* Research supported by a National Science Foundation Graduate Fellowship and a Rhodes Scholarship.

Proof. The circuits can be designed recursively by using the identity

$$\mathrm{MOD}_{k,r}^{(n)}(x) = \bigvee_{i=0}^{k-1} \left(\mathrm{MOD}_{k,r+i}^{(\lceil n/2 \rceil)}(x^L) \wedge \mathrm{MOD}_{k,r-i}^{(\lfloor n/2 \rfloor)}(x^R)\right).$$

Recent work by Paterson and Zwick has produced the following global upper bound.

Theorem 1.2 [7]. $D_{U_2}(\mathrm{MOD}_{k,r}^{(n)}) \leqslant c \log_2 n$, *where* $c \leqslant 5.07$.

2. A circuit design tool

With every Boolean function $f: \{0, 1)^n \to \{0, 1\}$, let us associate *mismatch bit problem* $\mathrm{MB}(f)$ involving two players $P1$ and $P2$: $P1$ receives a string $x_1 \in f^{-1}(1)$; $P2$ receives a string $x_2 \in f^{-1}(0)$; their task is to find a coordinate i such that $x_{1,i} \neq x_{2,i}$. Let $\mathrm{CC}(\mathrm{MB}(f))$ denote the minimum number of bits they have to communicate in order for both to agree on such a coordinate. (Unlike standard problems in communication complexity, the task of the players here is to solve a search, rather than a decision, problem.) Then we have

Theorem 2.1 [5]. *For every function* $f:\{0,1\}^n \to \{0,1\}$ *we have* $D_{U_2}(f) = \mathrm{CC}(\mathrm{MB}(f))$.

The elegant proof of this result describes very natural constructions, so that explicit communication protocols yield circuit designs, and vice versa. From a protocol for MB(f), we may build a circuit upward from the output gate, where each internal gate represents one bit of communication (and each path through the circuit represents a communication sequence). The details are found in [5].

3. The protocol for MOD_3

We give an economical communication protocol for $\mathrm{MB}(\mathrm{MOD}_{3,r}^{(n)})$. The basic idea is a divide-and-conquer argument. Our schemes uses messages of different lengths, which correspond to subproblems of different sizes.

Theorem 3.1. *Let* F_i *denote the ith term in the Fibonacci series* 1, 1, 2, 3, 5, 8, 13,... *and let* $r \in \{0,1,2\}$. *Then* $\mathrm{MB}(\mathrm{MOD}_{3,r}^{(F_i)})$ *can be solved in communication* $2i$.

Proof. We give an explicit communication protocol. After $P1$ receives string $x_1 \in (\mathrm{MOD}_{3,r}^{(F_i)})^{-1}(1)$ and $P2$ receives string $x_2 \in (\mathrm{MOD}_{3,r}^{(F_i)})^{-1}(0)$, the processors take turns communicating the *weights* (mod 3) of certain substrings of their inputs. (The weight of a binary string is the number of ones occurring in the string.) The goal is to find corresponding substrings of length 1 for which the weights differ.

More formally, we present the explicit protocol, which uses the integer variables MIN, MAX, TESTMIN, TESTMAX, OLDTESTMIN, OLDTESTMAX, LENGTH and SENDER, and the Boolean variable BALANCE.

Procedure INITIALIZE;
begin
MIN $\leftarrow 1$; MAX $\leftarrow F_i$;
TESTMIN $\leftarrow 1 + F_{i-1}$; TESTMAX $\leftarrow F_i$;
LENGTH $\leftarrow i - 1$;
SENDER $\leftarrow 1$;
$P1$ finds the remainder r of $\sum_{i=1+F_{i-1}}^{F_i} x_{1,i}$ upon division by 3 and transmits the value of r in binary to $P2$. $P2$ evaluates

$$\mathrm{BALANCE} \leftarrow \sum_{i=1+F_{i-1}}^{F_i} x_{1,i} \equiv \sum_{i=1+F_{i-1}}^{F_i} x_{2,i} \pmod 3.$$

end;

Procedure SEND RESULTS;
begin
Player SENDER updates the Boolean variable

$$\mathrm{BALANCE} \leftarrow \sum_{i=\mathrm{OLDTESTMIN}}^{\mathrm{OLDTESTMAX}} x_{1,i} \equiv \sum_{i=\mathrm{OLDTESTMIN}}^{\mathrm{OLDTESTMAX}} x_{2,i} \pmod 3;$$

computes the remainder r of $\sum_{i=\mathrm{TESTMIN}}^{\mathrm{TESTMAX}} x_{\mathrm{SENDER},i}$ upon division by 3; and transmits a message to the other player as indicated in Table 1.
(Note that this is a prefix code.)
end;

Table 1
The MOD_3 code

BALANCE	r	Message
True	0	00
True	1	01
True	2	10
False	0	1100
False	1	1101
False	2	1110

```
Protocol FIND MISMATCH BIT;
begin
   INITIALIZE;
   while LENGTH > 0 do
      begin
(*)      if BALANCE then
            begin
               MAX ← TESTMIN − 1;
               LENGTH ← LENGTH − 1;
            end;
         else
            begin
               MIN ← TESTMIN;
               LENGTH ← LENGTH − 2;
            end;
         OLDTESTMIN ← TESTMIN;
         OLDTESTMAX ← TESTMAX;
         TESTMIN ← MIN + F_LENGTH;
         TESTMAX ← MAX;
         SENDER ← 3 − SENDER;
         SEND RESULTS;
      end;
end (the index of the mismatch is MIN = MAX).
```

Proof of correctness. Use the invariant

$$I \equiv \left(\sum_{i=\text{MIN}}^{\text{MAX}} x_{1,i} \neq \sum_{i=\text{MIN}}^{\text{MAX}} x_{2,i} \right).$$

Note that each time (*) is executed, both processors know the value of BALANCE, so that both processors are able to update MIN, MAX, TESTMIN and TESTMAX.

Proof of complexity. After each execution of the **while–do** loop:

(1) If BALANCE = True, then LENGTH is reduced by 1, and 2 bits of communication are used.

(2) If BALANCE = False, then LENGTH is reduced by 2, and 4 bits of communication are used.

Thus the protocol halts within $2i$ bits of communication.

The asymptotic growth rate of the Fibonacci series yields the improved constant.

Table 2
Upper bounds

Function	Depth
MOD_3	$2.881 \log_2 n$
MOD_5	$3.475 \log_2 n$
MOD_{11}	$4.930 \log_2 n$

Corollary 3.2. *The counting functions* $\text{MOD}_{3,r}^{(n)}$ *may be computed by* U_2*-circuits in depth* $c \log_2 n + O(1)$, *where* $c = 2/(\log_2((1+\sqrt{5})/2)) = 2.881$.

4. Conclusion

By designing the cheapest codes and applying the analogous protocols, the bounds of Section 1 can be improved for the counting functions MOD_5 and MOD_{11} [1] (see Table 2).

These bounds apply to any congruence class with the indicated modulus.

In the case of MOD_5, we are able to use an extremely economical coding scheme (using words of length 3 and 4) and we believe the MOD_5 bound is very close to optimal.

Other bounds seem to contradict our intuition that MOD_p is at least as hard as MOD_q for primes p, q with $p > q$. Let B_2 denote the basis consisting of all the two-variable binary functions. The best upper bound for the formula size of MOD_5 over the basis B_2 is apparently $O(n^3)$. Since there exist B_2-formulas of size $O(n^{2.58})$ for the MOD_7 functions [9], we ask:

Open question. $D_{U_2}(\text{MOD}_5) \leqslant D_{U_2}(\text{MOD}_7)$?

Acknowledgment

I am grateful to my supervisor, Bill McColl, for getting me interested in the Karchmer–Wigderson results and for suggesting changes in an earlier draft of this paper; and to the referees for helpful comments.

References

[1] A. Chin, Shallow circuits for the counting functions, Tech. Rept. PRG-8-90, Computing Laboratory, University of Oxford, Oxford.

[2] M.J. Fischer, A.R. Meyer and M.S. Paterson, $\Omega(n \log n)$ lower bounds on length of Boolean formulas, *SIAM J. Comput.* **11** (1982) 416–427.

[3] M. Furst, J.B. Saxe and M. Sipser, Parity, circuits and the polynomial-time hierarchy, in: *Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science* (1981) 260–270.

[4] J. Håstad, Almost optimal lower bounds for small depth circuits, in: *Proc. 18th Annual ACM Symposium on Theory of Computing* (1986) 6–20.

[5] M. Karchmer and A. Wigderson, Monotone circuits of connectivity require super-logarithmic depth, in: *Proc. 20th Annual ACM Symposium on Theory of Computing* (1988) 539–550.

[6] W.F. McColl, Some results on circuit depth, Ph.D. Thesis, University of Warwick, 1976.

[7] M.S. Paterson and U. Zwick, Improved circuits and formulae for multiple addition, multiplication and symmetric Boolean functions, University of Warwick, in preparation.

[8] R. Smolensky, Algebraic methods in the theory of lower bounds for Boolean function complexity, in: *Proc. 19th Annual ACM Symposium on Theory of Computing* (1987) 77–82.

[9] D.C. Van Leijenhorst, A note on the formula size of the "mod k" functions, *Inform. Process. Lett.* **24** (1987) 223–224.