ハードウェア構成法　テスト答案 (一部)

440622B　安達知也

2. $(A+1)\times(B+1)-1$ の回路

2.1 前準備

まず、式を簡単にする。$\overline{A}=-A-1$ に注意すると、

$$\begin{aligned}(A+1)\times(B+1)-1 &= (-A-1)\times(-B-1)-1\\ &= \overline{A}\times\overline{B}-1\end{aligned}$$

というように、単純な 4bit 符号つき整数の積算に帰着できる。-1 の項は、符号なし整数演算との変換に利用する Offset に加味してやればよい。

次にこの式を筆算に直す。符号つき整数を符号なし整数に変換して演算するために最上位の符号 bit を NOT し、あとで Offset を加えて計算結果を合わせる。二重に NOT していて見にくいところもあるが、以下のようになる。

			$\overline{a_3}$	$\overline{a_2}$	$\overline{a_1}$	$\overline{a_0}$
		$\times$	$\overline{b_3}$	$\overline{b_2}$	$\overline{b_1}$	$\overline{b_0}$
			$\overline{\overline{a_3}\,\overline{b_0}}$	$\overline{a_2}\,\overline{b_0}$	$\overline{a_1}\,\overline{b_0}$	$\overline{a_0}\,\overline{b_0}$
		$\overline{\overline{a_3}\,\overline{b_1}}$	$\overline{a_2}\,\overline{b_1}$	$\overline{a_1}\,\overline{b_1}$	$\overline{a_0}\,\overline{b_1}$	
	$\overline{\overline{a_3}\,\overline{b_2}}$	$\overline{a_2}\,\overline{b_2}$	$\overline{a_1}\,\overline{b_2}$	$\overline{a_0}\,\overline{b_2}$		
$\overline{a_3}\,\overline{b_3}$	$\overline{\overline{a_2}\,\overline{b_3}}$	$\overline{\overline{a_1}\,\overline{b_3}}$	$\overline{\overline{a_0}\,\overline{b_3}}$			
0	0	1	0	0	0	0
						-1
			$\overline{\overline{a_3}\,\overline{b_0}}$	$\overline{a_2}\,\overline{b_0}$	$\overline{a_1}\,\overline{b_0}$	$\overline{a_0}\,\overline{b_0}$
		$\overline{\overline{a_3}\,\overline{b_1}}$	$\overline{a_2}\,\overline{b_1}$	$\overline{a_1}\,\overline{b_1}$	$\overline{a_0}\,\overline{b_1}$	
	$\overline{\overline{a_3}\,\overline{b_2}}$	$\overline{a_2}\,\overline{b_2}$	$\overline{a_1}\,\overline{b_2}$	$\overline{a_0}\,\overline{b_2}$		
$\overline{a_3}\,\overline{b_3}$	$\overline{\overline{a_2}\,\overline{b_3}}$	$\overline{\overline{a_1}\,\overline{b_3}}$	$\overline{\overline{a_0}\,\overline{b_3}}$			
0	0	0	1	1	1	1

出力は最小値 -57、最大値 63 で、7bit に収まっているので、Offset も 7bit 分だけ考慮すればよい。さらに、結果の最上位 bit は符号 bit なので、式の特性上、下からの Carry を待つことなく計算できる (後述)。

残りの 6bit は普通に掛け算するしかないようなので、さらに簡約を進めていく。まず、見た目の煩わしさを解消するため、以降は $\overline{a_i}\,\overline{b_j}$ を N_{ij}、$\overline{\overline{a_i}\,\overline{b_j}}$ を O_{ij} と表す。回路上では $\overline{a_i}\,\overline{b_j}$ は NOR ゲート、$\overline{\overline{a_i}\,\overline{b_j}}$ は OR ゲートで実現できるので、その頭文字をとった。

1bit の+1 回路 (Fig.1) を利用すると、下の状態まで進む。4bit めの 1 はこの方法で上げても速くならない気がするので、1 を足す Full Adder(Fig.2) に渡すことにした。

		N_{02}	N_{01}	N_{00}	
		O_{30}	N_{20}	N_{10}	O_{00}
	O_{31}	N_{21}	N_{11}	O_{01}	
O_{32}	N_{22}	N_{12}	O_{02}		
O_{23}	O_{13}	O_{03}			
		1			

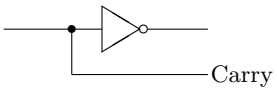

Fig.1 1bit +1 **回路**

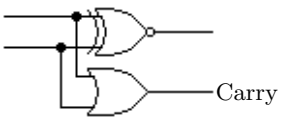

Fig.2 +1 Full Adder

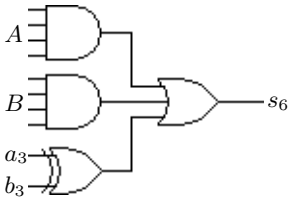

Fig.3 **最上位**bit **を求める回路**

2.2 回路

2.2.1 符号 bit

元の式に戻って、$(A+1)\times(B+1)<1$ となる条件を考えると、

- $(A=-1)$ OR $(B=-1)$
- $(A<-1)$ XOR $(B<-1)$

であることがわかる。これは Fig.3 に示すような簡単な回路で実現できる。

2.2.2 残りの bit

前項で導出した筆算をもとに、Full Adder と Half Adder を使ったブロック図にしたものが Fig.4 である。s_5 の bit から出た Carry は本来最上位へ上げるべきものだったが、今回は s_6 を別に求めてしまったので使わない。

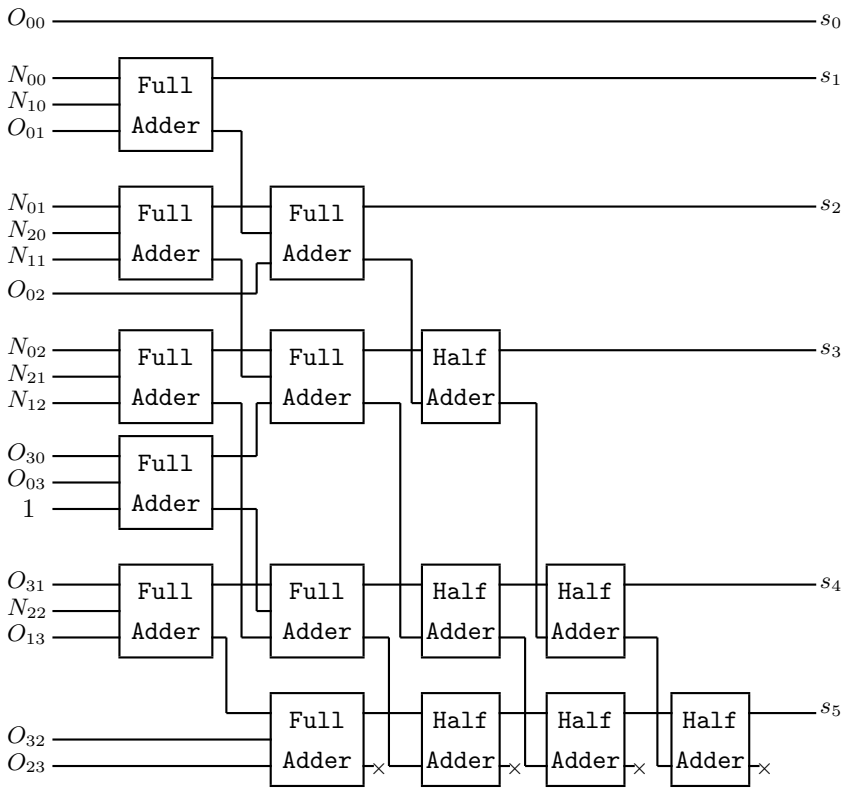

Fig.4 **加算ブロック図**

2.2.3 全回路図

Full Adder と Half Adder を展開し、全体の回路を作る。ただし、s_5 の bit の Majority Gate は省く。また、シミュレータに ExNOR が用意されていなかった都合で、+1 Full Adder の ExNOR を ExOR に変更し、前段の OR の一つを NOR にした。それに伴い、その部分の足し算の Carry の計算は 4 入力 OR で代用することとなった。Fig.5 に全回路図を示す。

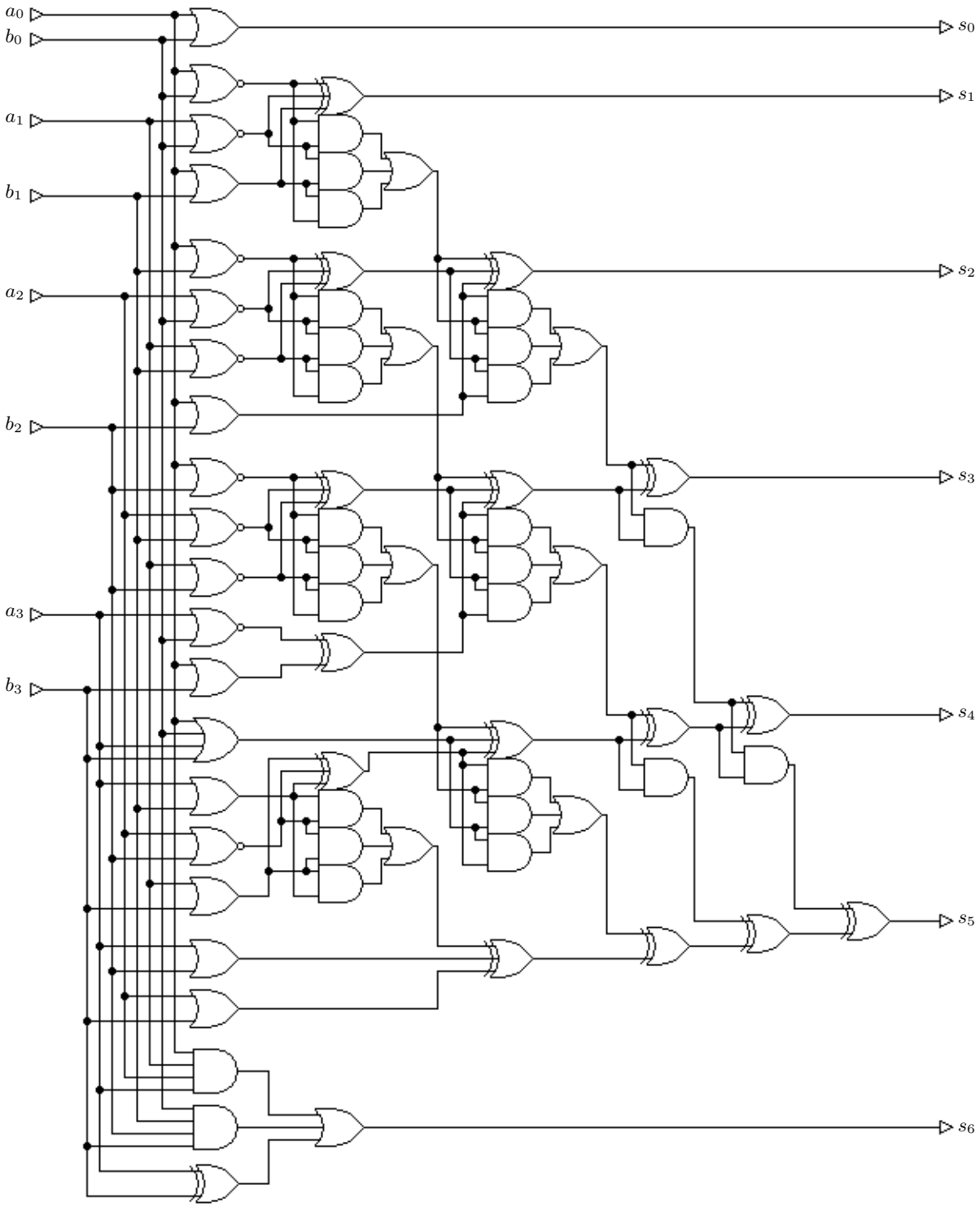

Fig.5 **全回路図**

2.3 シミュレーション

題意を満たす回路になっているか確かめるため、全通りの入力を加えて正しい答えが出るかのテストを行った。A と B それぞれを -8 から 7 まで変えていったときの入出力波形を Fig.6 に示す。最初は $A = B = -8$ で、そのときの出力は 48、次は $A = -7$、$B = -8$ で、出力は 41…というように、正しい結果が出ていることがわかる。

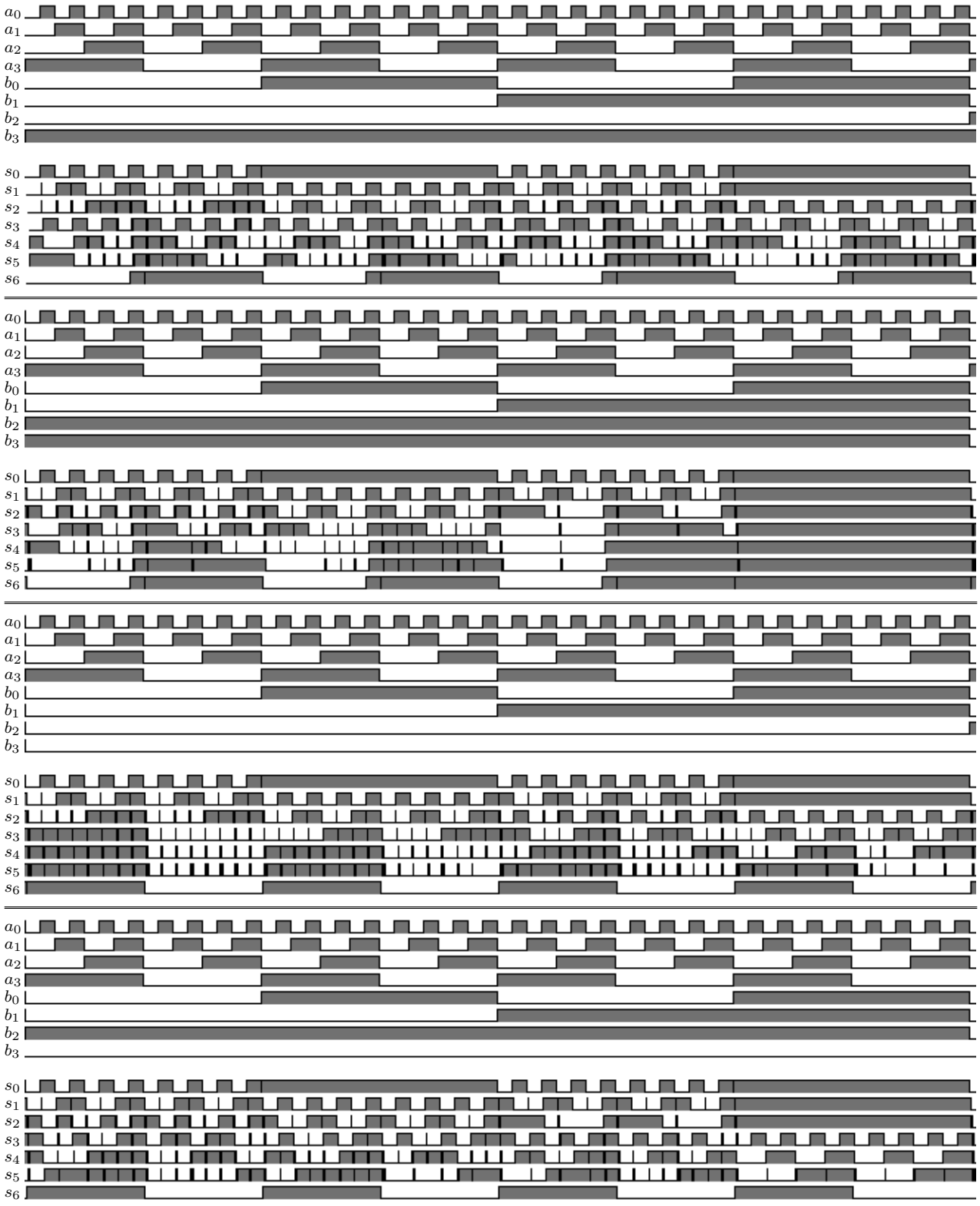

Fig.6 **入出力波形**

また、最悪の場合でも 8 段分の遅延で答えが出ることも確認できた。これは回路図から読み取れる理論値と同じである。あくまでもシミュレータの出力なので、実際の基板でどうなのかは全く想像もつかないが、おそらく実戦的にも速い回路であろうと思われる。

3. 講義へのコメント

演算回路も順序回路も、回路の動作自体は考えればわかるものなので大丈夫なのですが、演算回路は使い道がはっきりしているのに対し、順序回路はどのような場面で使うのかがどうにもつかめずにいます。機械全体をどのように構成していくのかを説明していただけると、個々の回路の使い方もよりよくわかると思うのですが…