【4】

(i) ==========

データパスと制御回路の入出力は次のように考えてよいであろう. また, 下の論理変数が定義されている.

データパス入力:	**a** = (a_3 a_2 a_1 a_0) , **b** = (b_3 b_2 b_1 b_0) , sub （加減算を指定する信号）
データパス出力:	シフトレジスタ REG_A の出力,
	D フリップフロップ DFF_{ovf} , DFF_Z , および DFF_C の出力
制御回路入力:	start （演算の開始を指定する信号）
制御回路出力:	done （演算の終了を知らせる信号）
論理変数:	A : シフトレジスタ REG_A の最下位ビット
	B : シフトレジスタ REG_B の最下位ビット
	C : D フリップフロップ DFF_C の状態（桁上げを示す）
	F : D フリップフロップ DFF_{ovf} の状態（オーバーフローを示す）
	Z : D フリップフロップ DFF_Z の状態（全ビット 0 か否かを示す）
	S : D フリップフロップ DFF_{sub} の状態（sub の値を保持している）

加減算を行う際の各桁での演算は, 6.1 節で説明したように, 下記のようになる.

加算（S = 0）の場合: $(A)_2 + (B)_2 + (C)_2$ を計算する. ただし, 最下位ビットのとき C = 0.

減算（S = 1）の場合: $(A)_2 + (\overline{B})_2 + (C)_2$ を計算する. ただし, 最下位ビットのとき C = 1.

これらは,

$$D = S \oplus B = \begin{cases} B & : S = 0 \\ \overline{B} & : S = 1 \end{cases}$$

であることを利用すれば, 全加算器（FA:Full Adder）を用いて実現できる. すなわち, C の初期値を C = S とし, A, D, C を全加算器に入力することにより, S の場合分けをすることなく上の演算を実行できる. そこで, 全加算器の出力を桁上げ X と結果 Y とする. すなわち,

$$(X\ Y)_2 = (A)_2 + (D)_2 + (C)_2$$

とする. そうすると, 次状態において, 桁上げ X を D フリップフロップ DFF_C に入れ, シフトレジスタ REG_A を右シフトする際, Y を REG_A の最上位ビットに入れてやれば, 各桁での演算結果を DFF_C と REG_A の最上位ビットに入れることができる. 従って, この処理を 4 回繰り返すことにより, 逐次処理型の加減算器を実現できる.

次に, オーバーフローを示す D フリップフロップ DFF_{ovf} について考える. オーバーフローは数値ビットからの桁上げと符号ビットからの桁上げが異なるときかつそのときに限り生じているから, 最上位ビットでの演算を行った際, 演算結果の桁上げ X とその前の桁での桁上げ C とを比較し, それらが異なれば 1, 同じであれば 0 を DFF_{ovf} に入れればよい. すなわち, DFF_{ovf} の状態 F の次状態 F’ を

$$F' = X \oplus C$$

で計算すればよい. しかし, 毎回この値を計算し, DFF_{ovf} に蓄えておいても, 最上位ビットに対して加減算を実行すれば全処理は終了するので, 全処理の終了時には, DFF_{ovf} に最上位ビットに対する結果が入ることになる. そこで, 4 回の繰り返し全てにおいてこの式を用いて F' を計算し, DFF_{ovf} に入れておくことにする.

最後に, 演算結果の 4 ビットが全て 0 であるか否かを示す D フリップフロップ DFF_Z について考える. DFF_Z の状態 Z を 1 にするのは, 上に示した演算結果の Y が, 4 回の繰り返しにおいて全て 0 のときかつそのときに限る. 従って, 1 回でも Y が 1 になれば, それ以後 Z = 0 となるようにすればよい. そこで, 演算 i 回目 ($0 \le i \le 3$) の Y を Y_i と書くと, Z の値は次式のように書けるから,

$$Z = \overline{Y_0 + Y_1 + Y_2 + Y_3} = \overline{\left(\left((0 + Y_0) + Y_1\right) + Y_2\right) + Y_3}$$

$Z_0 = 1$, $Z_1 = \overline{\overline{Z_0} + Y_0}$, $Z_2 = \overline{\overline{Z_1} + Y_1}$, $Z_3 = \overline{\overline{Z_2} + Y_2}$, $Z_4 = \overline{\overline{Z_3} + Y_3}$ とすると,

$$Z_4 = \overline{\overline{Z_3} + Y_3} = \overline{(\overline{Z_2} + Y_2) + Y_3} = \overline{\left((\overline{Z_1} + Y_1) + Y_2\right) + Y_3} = \overline{\left(\left((\overline{Z_0} + Y_0) + Y_1\right) + Y_2\right) + Y_3}$$
$$= \overline{\left(\left((0 + Y_0) + Y_1\right) + Y_2\right) + Y_3} = Z$$

を得る. 従って, Z の初期値を 1 とし, 次状態を $Z' = \overline{\overline{Z} + Y}$ なる式で計算してやれば, 4 回の繰り返し後に, DFF_Z に所望の値を入れることができる.

以上より, 逐次処理型の加減算器の手順は次のように書くことができる.

0°: done を 0 とし, start = 0 の間, この操作 0° を繰り返し, start = 1 になったならば, 1° 以降の操作を実行する.
1°: シフトレジスタ REG_A および REG_B にそれぞれ入力 **a** = (a_3 a_2 a_1 a_0) および **b** = (b_3 b_2 b_1 b_0) を入れ, DFF_{sub}, DFF_Z, および DFF_C にそれぞれ sub, 0, および sub を入る.
2°: 以下の操作を 4 回繰り返す. ただし, 2-1° ～2-4° の操作は 1 クロックで行う.
  2-1°: 全加算器(FA)を用いて, $(X\ Y)_2 = (A)_2 + (D)_2 + (C)_2$ を計算する. ここで, D = S⊕B である.
  2-2°: REG_A への入力 I_L を I_L = Y として, REG_A を右に 1 ビットシフトする.
  2-3°: REG_B を右に 1 ビットシフトする. このとき, REG_B への入力 I_L は何でもよい.
  2-4°: DFF_C, DFF_{ovf}, および DFF_Z に, それぞれ Y, X⊕C, および $\overline{\overline{Z} + Y}$ を入れる.
3°: done =1 を出力して, 0° に戻る.

この逐次処理型の加減算器では, 2° を終了するのに 4 クロックを要する. これに対して, 6.2 節で述べた桁上げ伝搬加減算器では, 1 クロックですむ. しかし, 桁上げ伝搬加減算器には, 6 章演習問題【2】に示したように, 信号の伝搬に時間のかかる(遅延が長い)パスが存在するため, 1 クロックの周期をその遅延より短くできない. それに対して, この逐次処理型の加減算器では, 一つの D フリップフロップ(あるいはレジスタ)から別のあるいは同じ D フリップフロップ(あるいはレジスタ)までのパスの遅延が短いため, 1 クロックの周期を短くでき, かつ回路規模も桁上げ伝搬加減算器より小さくできる.

(ii) ==========

データパスには, シフトレジスタ REG_A および REG_B, ならびに D フリップフロップ DFF_{sub}, DFF_C, DFF_{ovf}, および DFF_Z がある. また, 演算器として, 全加算器があり, DFF_{ovf} や DFF_Z の次状態を決定する回路がある. それらの接続関係は, 全加算器に関する式

$$(X\ Y)_2 = (A)_2 + (D)_2 + (C)_2$$

および

$$D = S \oplus B \quad , \qquad F' = X \oplus C \quad , \qquad Z' = \overline{\overline{Z} + Y}$$

から得られる. また, シフトレジスタには S_1 および S_0 の 2 つの制御入力があるが, シフトレジスタ REG_A および REG_B に対する動作は, 上の手順 0°～3° から分かるように, どの時点においても同じである. 従って, REG_A に対する制御信号と REG_B に対する制御信号を別に用意する必要はない.

D フリップフロップの入力には, 処理の時点に応じて入力を選択するためのマルチプレクサが必要となる. そこで, これについて考える.

DFF_{sub}, DFF_C, および DFF_Z では, 外部入力 sub を読み込んだり, 初期値を設定したりしなければならず, データパスの出力を保持する DFF_C, DFF_{ovf}, および DFF_Z では, 計算した値を変化させないようにしておく必要がある. そこで, 初期値を設定するか否かを指定する制御信号 init および値を変化させるか否かを指定する制御信号 En を用意し, 以下のような動作をさせることにする.

En	init	DFF_{sub} (S)	DFF_C (C)	DFF_Z (Z)	DFF_{ovf} (F)
0	*		値を変化させない		
1	0	値を変化させない	X にする	$\overline{\overline{Z} + Y}$ にする	$X \oplus C$ にする
1	1	sub の値にする		1 にする	*（don't care）

そうすると, データパスの入出力は下記のようになる.

データパスの入力

a = $(a_3\ a_2\ a_1\ a_0)$, **b** = $(b_3\ b_2\ b_1\ b_0)$, sub （加減算を指定する信号）

S_1, S_0 : シフトレジスタ用の制御信号

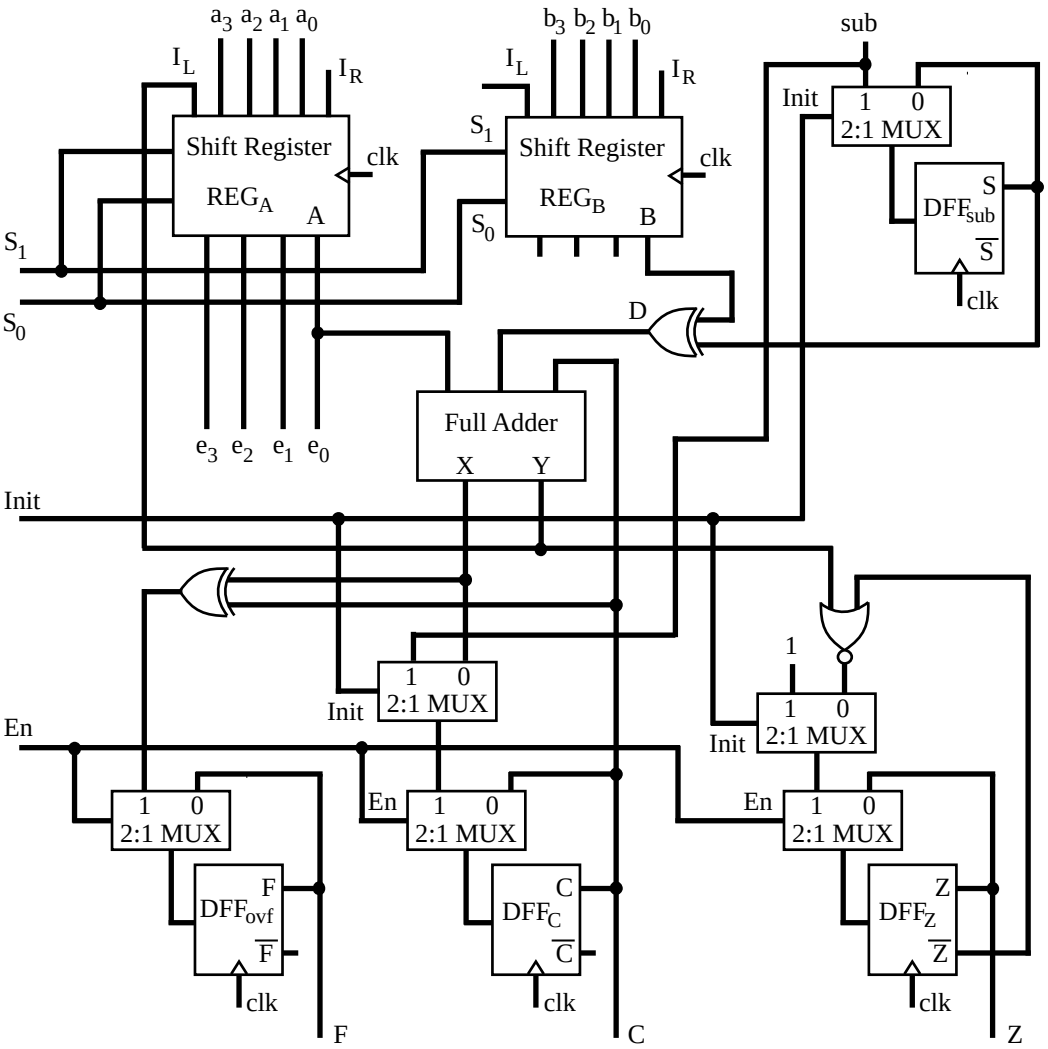
En, init : マルチプレクサ用の制御信号

データパスの出力

シフトレジスタ REG_A の出力

D フリップフロップ DFF_{ovf} , DFF_Z , および DFF_C の出力

これらの入出力が決まれば，上述の 0°～3° の動作をするデータパスを，6 章演習問題【3】のシフトレジスタの回路，全加算器，および 2:1 マルチプレクサを用いて，下図のように構成することができる．



この図で，clk はクロック信号 clock を意味する

(iii) ==========

次に制御回路を設計する.

制御回路は, 上に示したデータパスに, 上記の 0°～3° の操作を実行させねばならないが, その際, データパスでの演算結果によって操作を変える必要はなく, 0°～3°の操作から定められる順序に従って制御信号をデータパスに送り出すだけでよい. 出力すべき制御信号は, init, En, S_1, S_0 ならびに動作の終了を示す done である.

さらに, 操作 2° の繰り返しをアップカウンタで制御することにすると, アップカウンタ用の制御信号 reset および enable が必要となる. ただし, この制御信号はデータパスに向かうのではなく, 制御回路内で利用される. このアップカウンタは, 7 章演習問題【1】で示したように, 2 ビットの状態変数を持ち, 最上位ビットからの桁上げ C_2 が 1 であるか否かによって, 4 回目か否かを判定することにする.

纏めると, 制御回路において出力すべき信号は, 下記のようになる.

マルチプレクサ用 : init , En （これらの意味は上の表を参照）

シフトレジスタ用 : S_1 , S_0 （動作の詳細は下記の表参照）

アップカウンタ用 : reset , enable , （動作の詳細は下記の表参照）

制御入力		動作
S_1	S_0	
0	0	変化なし
0	1	入力取り込み
1	0	左 1 ビットシフト
1	1	右 1 ビットシフト

制御入力		動作
Clear	Enable	
0	0	変化なし
0	1	カウントアップ
1	*	値を 0 にする

上に述べたように, この制御回路では, 入力を用いて出力を決定する必要がないので, 出力が状態だけで決定されるムーア型の順序回路にする. そうすると, 上記の 0°～3° の操作から, 以下の状態を設ければよいことが分かる.

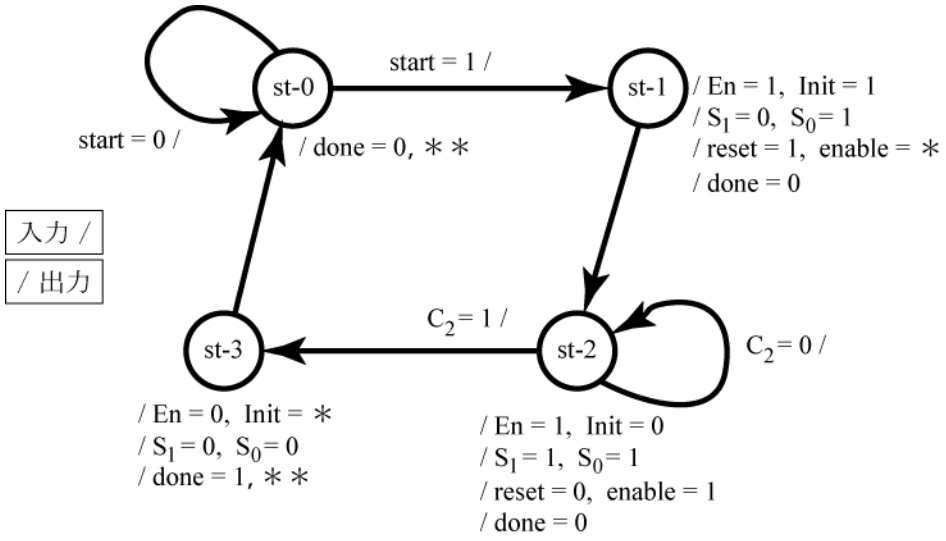
状態 st-0: done = 0 を出力し, 他の出力は 0 でも 1 でもよい(don't care である).
start = 1 であれば, 次は状態 st-1 に行き,
start = 0 であれば, 次は状態 st-0 に戻る.

状態 st-1: シフトレジスタ REG_A および REG_B, ならびに D フリップフロップ DFF_{sub}, DFF_C, DFF_{ovf}, および DFF_Z に外部入力や初期値を入力するため, En = 1, init = 1, S_1 = 0, S_0 = 1 を出力する. また, アップカウンタを reset するため, reset = 1 とする. enable は don't care で, done は 0 である.
次は状態 st-2 に行く.

状態 st-2: 2 つのシフトレジスタを右シフトし, 全 D フリップフロップに演算されて出てきた値を取り込むよう, En = 1, Init = 0, S_1 = 1, S_0 = 1 を出力する. また, アップカウンタをカウントアップするため, reset = 0, enable = 1 とする. done は 0 である.
アップカウンタからの桁上げ C_2 = 1 ならば, 次は状態 st-3 に行き,
C_2 = 0 ならば, 次は状態 st-2 に戻る.

状態 st-3: この状態に来たということは, 状態 st-2 を 4 回繰り返し, 計算が終了したので, done = 1 を出力する. また, シフトレジスタや D フリップフロップを変化させないよう, En = 0, S_1 = 0, S_0 = 0 を出力する. init, clear, enable は don't care である.
次は状態 st-0 に戻る.

このような状態を考えたときの状態遷移図は下図のようになる. ここで, 出力は入力に依存しないので, 各状態の下に書いてあり, 次状態が入力によって変わらない場合には, 矢印に入力条件を書いていない. また, ＊は don't care を, ＊＊はそこに書かれていない出力が don't care であることを示す.



(iv) ==========

4 つの状態が定義されたので, 状態変数 x_1, x_2 を用意し, 下の表の左端のように状態割当を行うと, 下のような状態遷移表が書ける.

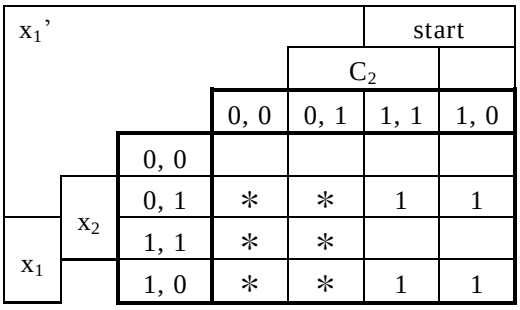
現状態: x_1, x_2		入力	次状態: x_1', x_2'							
			start = 0				start = 1			
			C_2 = 0		C_2 = 1		C_2 = 0		C_2 = 1	
st-0	0, 0		st-0	0, 0	st-0	0, 0	st-1	0, 1	st-1	0, 1
st-1	0, 1		＊	＊, ＊	＊	＊, ＊	st-2	1, 0	st-2	1, 0
st-2	1, 0		＊	＊, ＊	＊	＊, ＊	st-2	1, 0	st-3	1, 1
st-3	1, 1		＊	＊, ＊	＊	＊, ＊	st-0	0, 0	st-0	0, 0

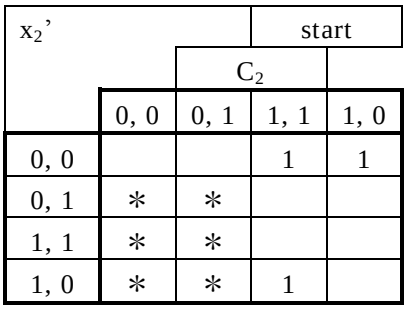
また, 出力表は下記のようになる. Moore 型回路の場合には, 出力は入力に依存しない.

現状態: x_1, x_2		出力						
		done	En	init	S_1	S_0	reset	enable
st-0	0, 0	0	＊	＊	＊	＊	＊	＊
st-1	0, 1	0	1	1	0	1	1	＊
st-2	1, 0	0	1	0	1	1	0	1
st-3	1, 1	1	0	＊	0	0	＊	＊

これらより, ムーア型の制御回路を以下のように構成することができる.

x_1' および x_2' は, 下図のようなカルノー図となる.





従って, 次式を得る.

$$x_1' = x_1 \cdot \overline{x_2} + \overline{x_1} \cdot x_2 = x_1 \oplus x_2$$

$$x_2' = \overline{x_1} \cdot \overline{x_2} \cdot \mathrm{start} + x_1 \cdot \overline{x_2} \cdot C_2$$

done は, 上の出力表から分かるように, 状態 st-3, すなわち $(x_1, x_2) = (1, 1)$ の場合に 1 にすればよいから, 次式で書ける.

$$\mathrm{done} = x_1 \cdot x_2$$

En および init は，下図のようなカルノー図となる．

En		x_2	
		0	1
	0	＊	1
x_1	1	1	

init		x_2	
		0	1
	0	＊	1
x_1	1		＊

従って，次式を得る．

$$En = \overline{x_1} + \overline{x_2} \quad , \qquad Init = \overline{x_1} = x_2$$

S_1 および S_0 は，下図のようなカルノー図となる．

S_1		x_2	
		0	1
	0	＊	
x_1	1	1	

S_0		x_2	
		0	1
	0	＊	1
x_1	1	1	

従って，次式を得る．

$$S_1 = \overline{x_2} \quad , \qquad S_0 = \overline{x_1} + \overline{x_2}$$

reset および enable は，下図のようなカルノー図となる．

reset		x_2	
		0	1
	0	＊	1
x_1	1		＊

enable		x_2	
		0	1
	0	＊	＊
x_1	1	1	＊

従って，次式を得る．

$$reset = \overline{x_1} = x_2 \quad , \qquad enable = 1$$

以上より，

$$En = \overline{x_1} + \overline{x_2} = \overline{x_1 \cdot x_2} = \overline{done} \quad , \qquad S_0 = \overline{x_1} + \overline{x_2} = \overline{x_1 \cdot x_2} = \overline{done}$$

であることを用い，下記の式を用いることにすれば，

$x_1' = x_1 \oplus x_2$ ，　$x_2' = \overline{x_1} \cdot \overline{x_2} \cdot start + x_1 \cdot \overline{x_2} \cdot C_2$ ，　$done = x_1 \cdot x_2$

$En = \overline{done}$ ，　$init = x_2$

$S_1 = \overline{x_2}$ ，　$S_0 = \overline{done}$

$reset = x_2$ ，　$enable = 1$

下図のような制御回路を構築できる．なお，ここでは初期状態を状態 st-0 にできるよう，dff-reset 付き D フリップフロップを用い，dff-reset 信号により状態を $x_1 = 0$ および $x_2 = 0$ にできるようにしている．