計算機論理設計
9. 算術論理演算器設計

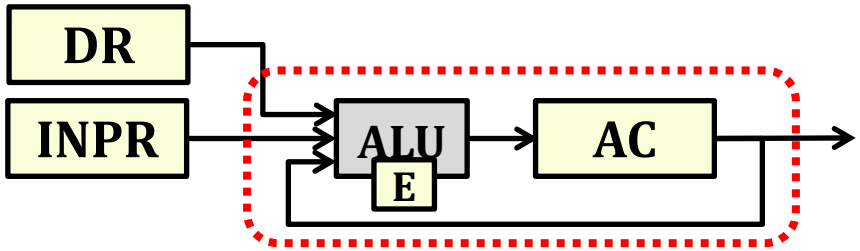
一色 剛
工学院情報通信系
isshiki@ict.e.titech.ac.jp

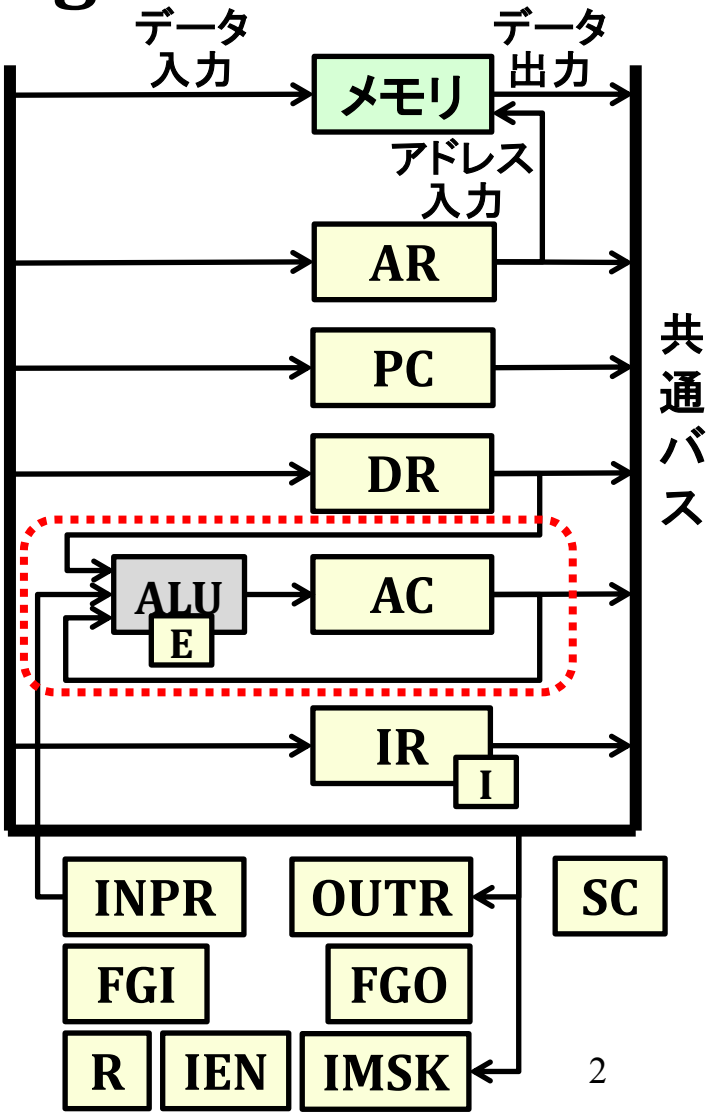
算術論理演算器
（ALU : Arithmetic Logic Unit）

■ **ALUの機能**

- 算術演算 : ADD, INC
- 論理演算 : AND, CMA, CME
- シフト演算 : CIL, CIR
- データ転送 : LDA, INP
- その他 : CLA, CLE

→ ALU回路設計とAC/Eレジスタ制御を同時に考える



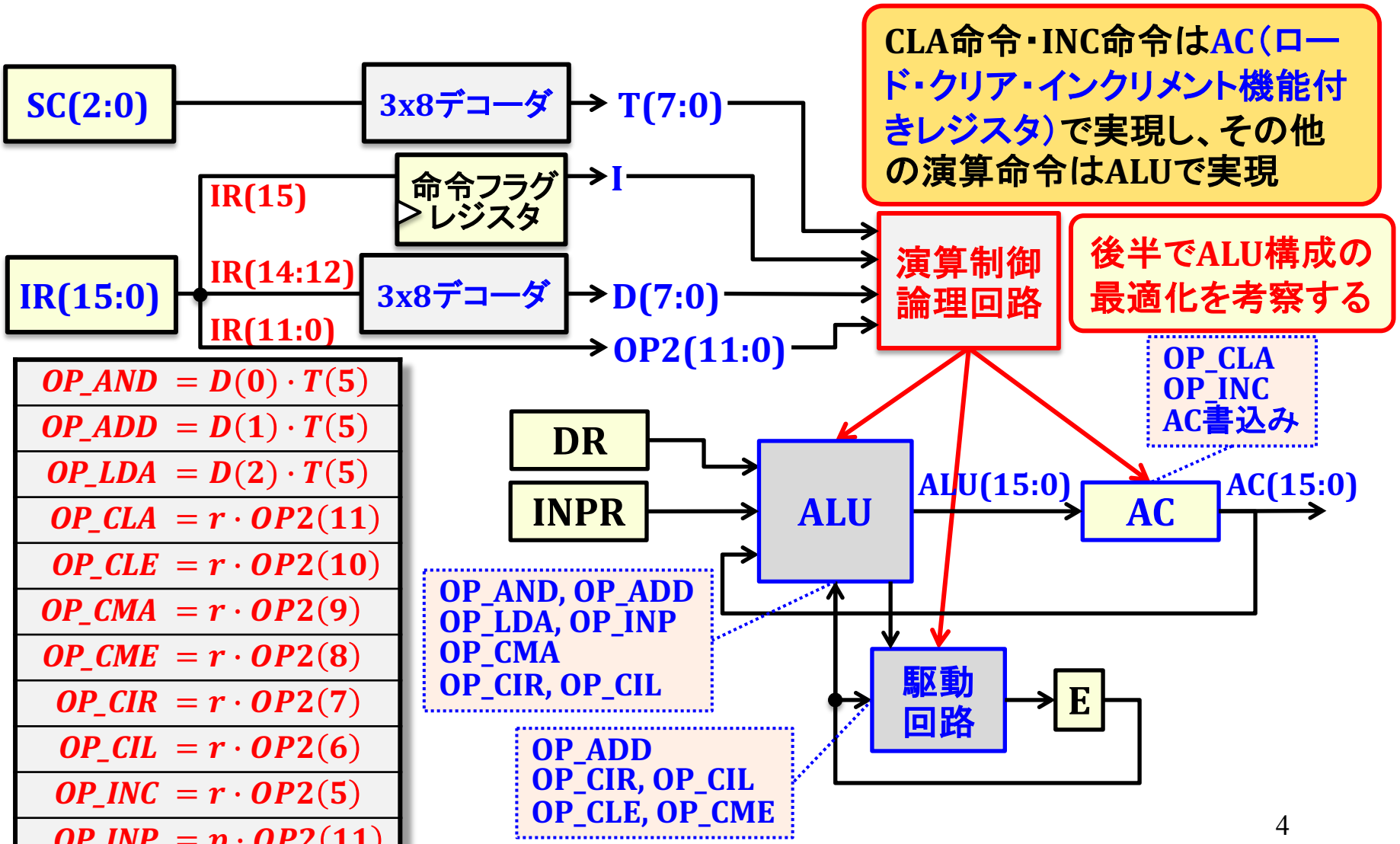


AC/Eレジスタを変更する命令実行動作

$$r = D(7) \cdot \bar{I} \cdot T(3)\,, p = D(7) \cdot I \cdot T(3)$$

命令	制御信号（=実行条件）	レジスタ転送式
AND	$OP_AND = D(0) \cdot T(5):$	$AC \leftarrow AC \,\&\, DR$
ADD	$OP_ADD = D(1) \cdot T(5):$	$AC \leftarrow AC + DR,\ E \leftarrow C_{out}(16)$
LDA	$OP_LDA = D(2) \cdot T(5):$	$AC \leftarrow DR$
CLA	$OP_CLA = r \cdot OP2(11):$	$AC \leftarrow 0$
CLE	$OP_CLE = r \cdot OP2(10):$	$E \leftarrow 0$
CMA	$OP_CMA = r \cdot OP2(9):$	$AC \leftarrow \overline{AC}$
CME	$OP_CME = r \cdot OP2(8):$	$E \leftarrow \overline{E}$
CIR	$OP_CIR = r \cdot OP2(7):$	$AC(14{:}0) \leftarrow AC(15{:}1),$ $AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$OP_CIL = r \cdot OP2(6):$	$AC(15{:}1) \leftarrow AC(14{:}0),$ $AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$OP_INC = r \cdot OP2(5):$	$AC \leftarrow AC + 1$
INP	$OP_INP = p \cdot OP2(11):$	$AC(7{:}0) \leftarrow INPR$

演算制御論理設計



$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CLE = r \cdot OP2(10)$
$OP_CMA = r \cdot OP2(9)$
$OP_CME = r \cdot OP2(8)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$

AC制御論理設計

$$AC_LD = OP_AND + OP_ADD + OP_LDA + OP_CMA + OP_CIR + OP_CIL + OP_INP$$

$$AC_CLR = OP_CLA$$

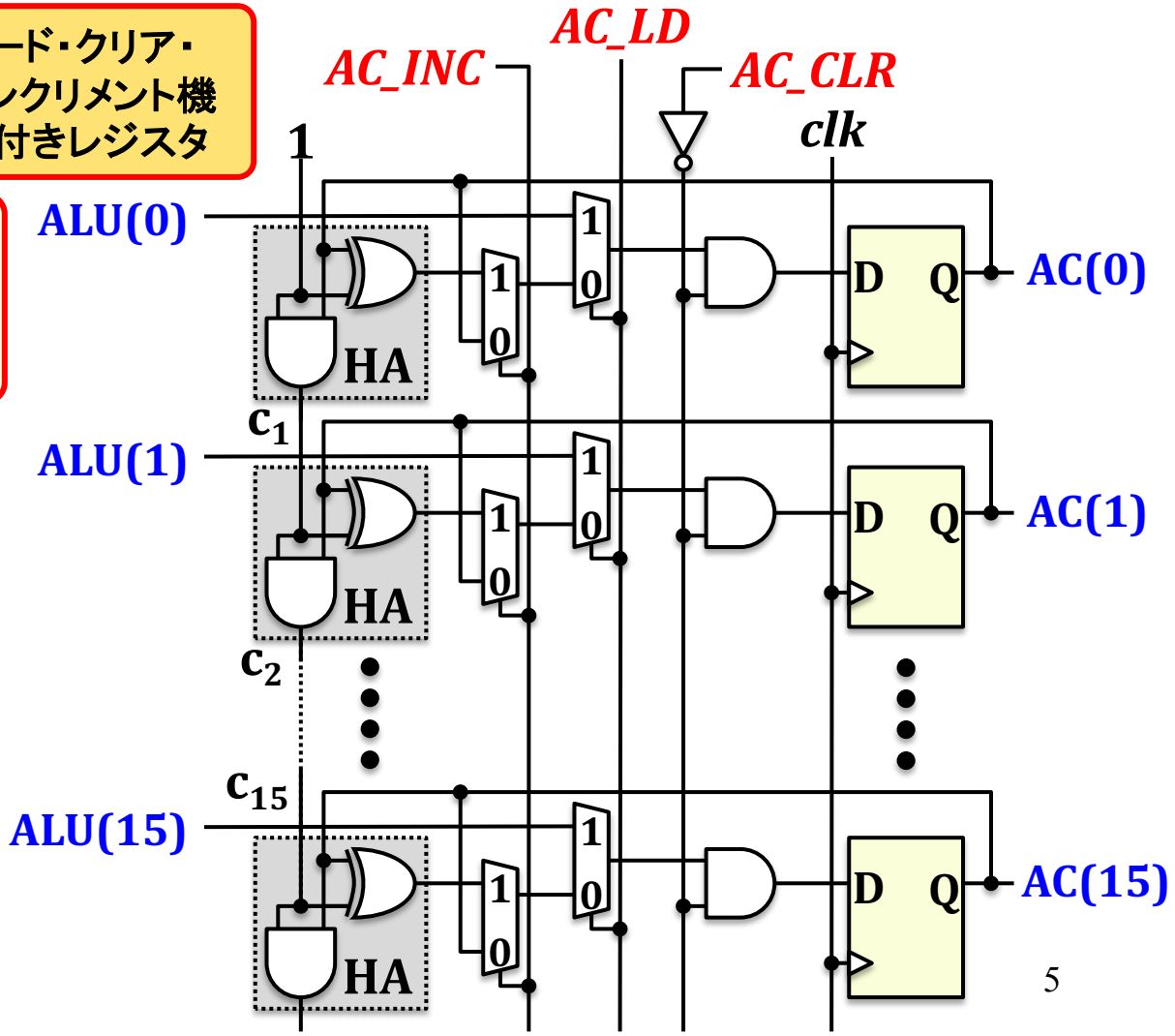
$$AC_INC = OP_INC$$

ロード・クリア・インクリメント機能付きレジスタ

AC_LD : CLA命令・INC命令以外のAC変更動作はALU出力から供給される

ACを変更する命令動作

$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CMA = r \cdot OP2(9)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$



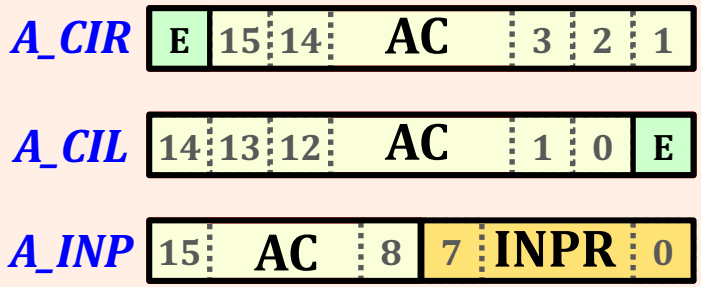
ALU機能と回路構成

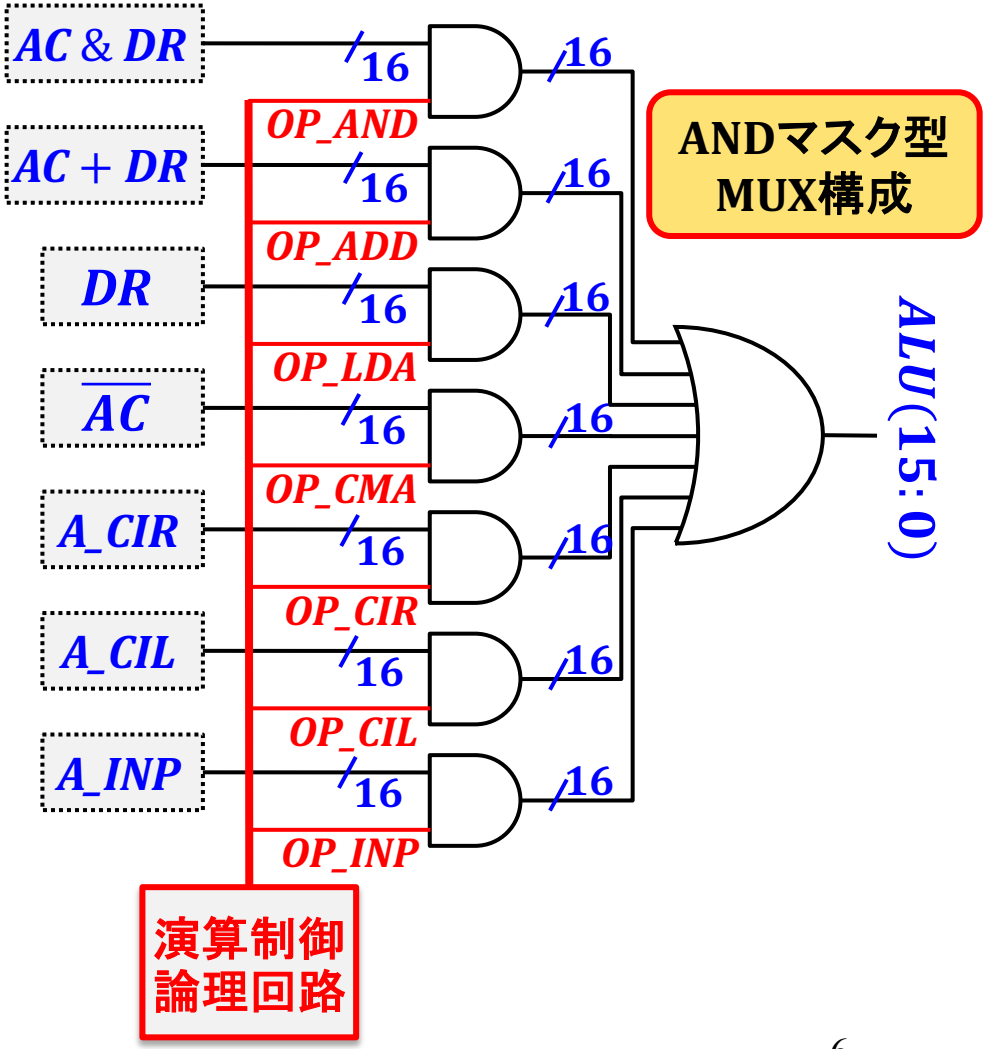
ALUで実現する機能

OP_AND	$AC \,\&\, DR$
OP_ADD	$AC + DR$
OP_LDA	DR
OP_CMA	$\overline{AC}$
OP_CIR	$A_CIR = \{E, AC(15:1)\}$
OP_CIL	$A_CIL = \{AC(14:0), E\}$
OP_INP	$A_INP = \{AC(15:8), INPR\}$

7つの出力を
MUXで選択

{X, Y}：ビット連結





ALUビットスライス回路構成

ALUで実現する機能

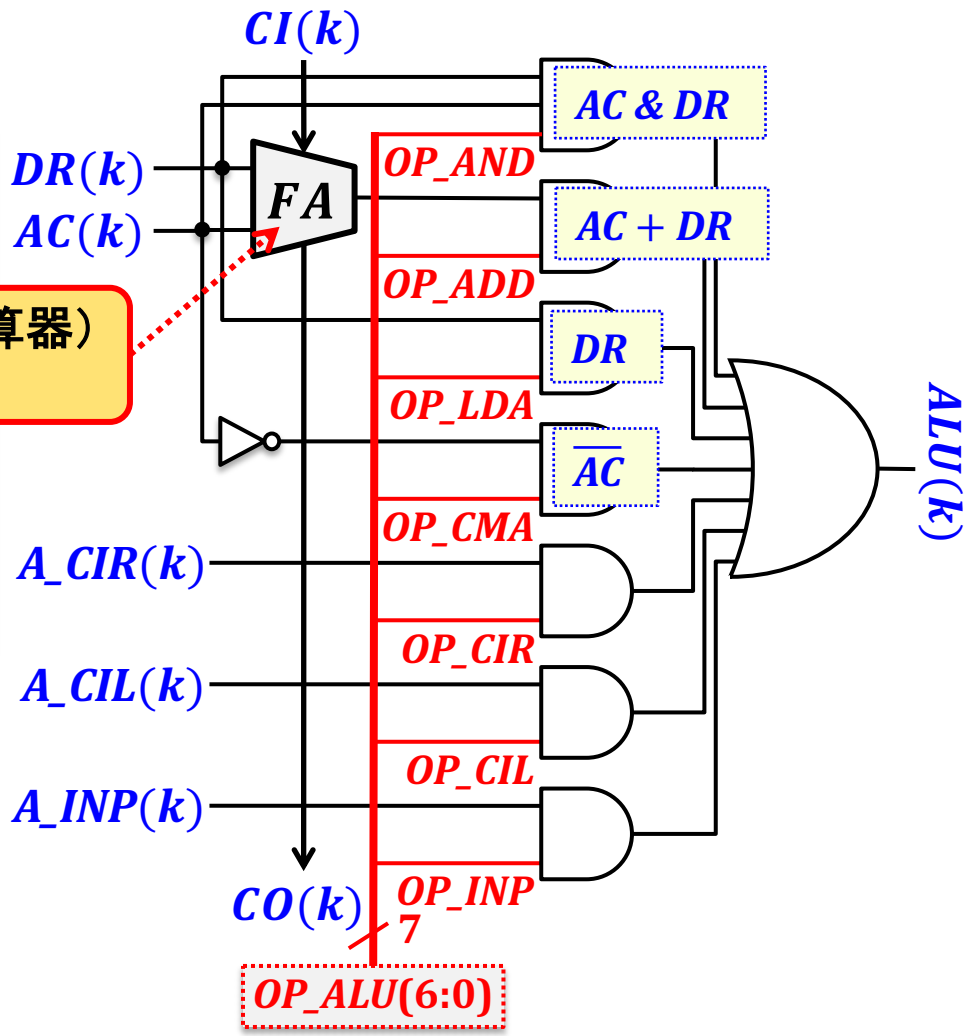
OP_AND	$AC \,\&\, DR$
OP_ADD	$AC + DR$
OP_LDA	DR
OP_CMA	$\overline{AC}$
OP_CIR	$A_CIR = \{E, AC(15:1)\}$
OP_CIL	$A_CIL = \{AC(14:0), E\}$
OP_INP	$A_INP = \{AC(15:8), INPR\}$

FA : Full-Adder（全加算器）
講義資料1 :P10 参照

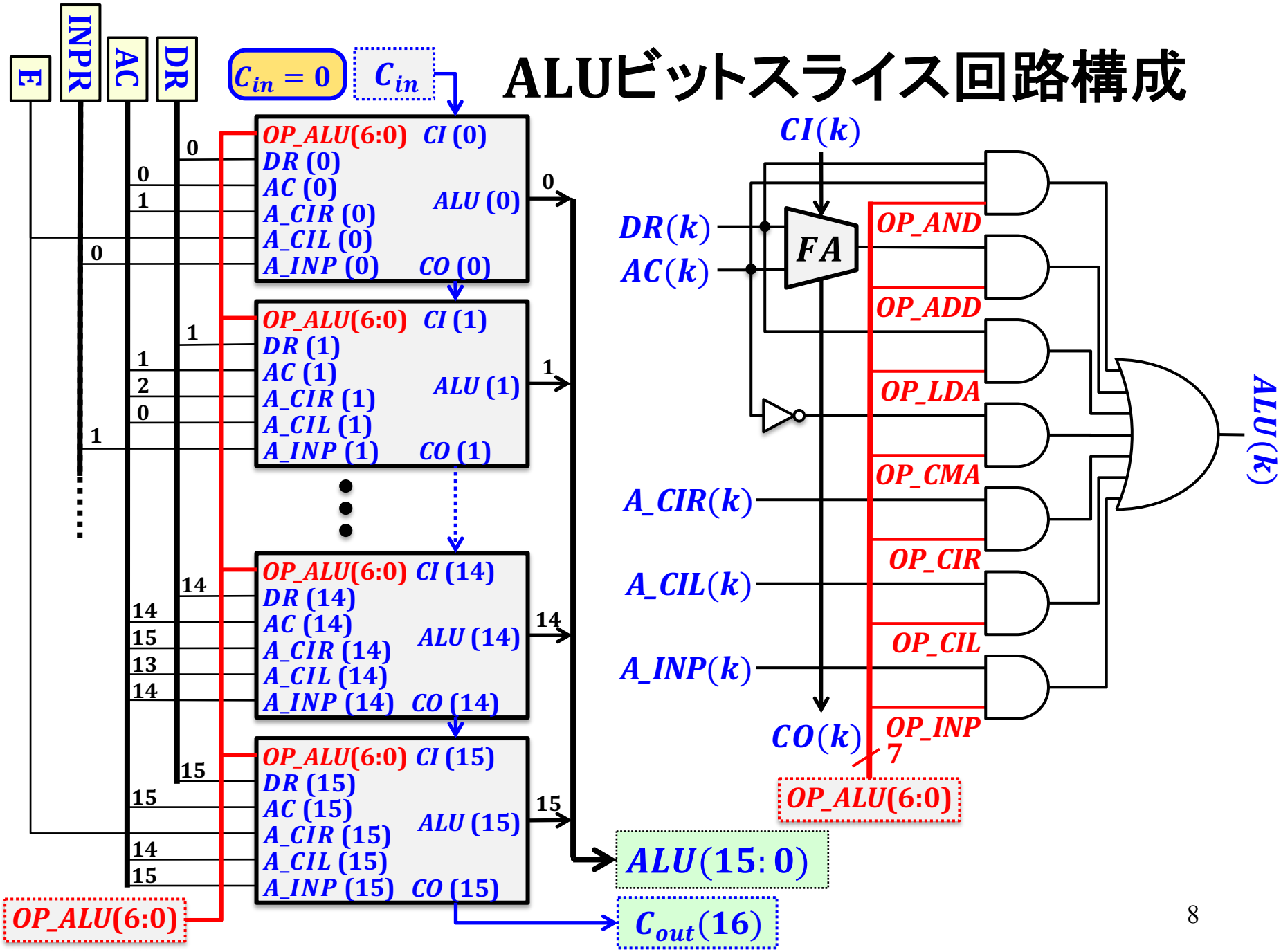
$$A_CIR(k) = \begin{cases} E & (k = 15) \\ AC(k+1) & (k \neq 15) \end{cases}$$

$$A_CIL(k) = \begin{cases} AC(k-1) & (k \neq 0) \\ E & (k = 0) \end{cases}$$

$$A_INP(k) = \begin{cases} INPR(k) & (k < 8) \\ AC(k) & (k \geq 8) \end{cases}$$



ALUビットスライス回路構成



Eレジスタ駆動回路設計

$r = D(7) \cdot \overline{I} \cdot T(3)$

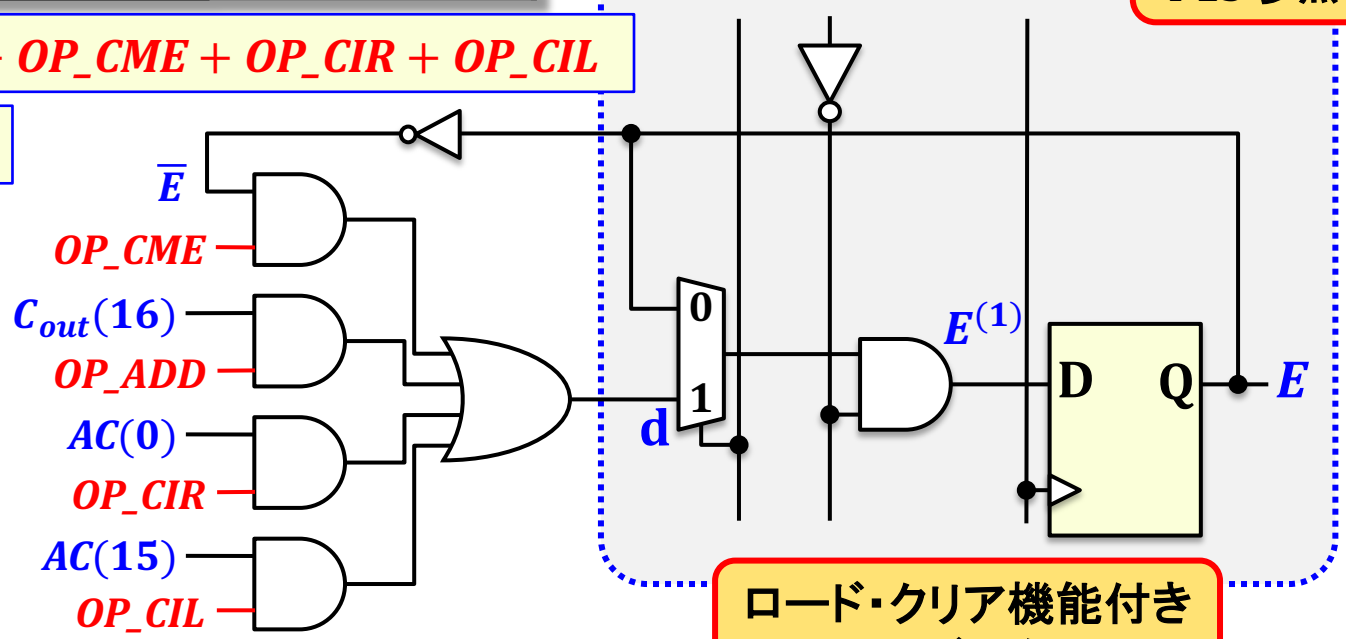
$OP_ADD = D(1) \cdot T(5)$:	$E \leftarrow C_{out}(16)$
$OP_CLE = r \cdot OP2(10)$:	$E \leftarrow 0$
$OP_CME = r \cdot OP2(8)$:	$E \leftarrow \overline{E}$
$OP_CIR = r \cdot OP2(7)$:	$E \leftarrow AC(0)$
$OP_CIL = r \cdot OP2(6)$:	$E \leftarrow AC(15)$

ロード・クリア制御回路の簡単化を考える → 次ページ

講義資料7：P13 参照

$E_LD = OP_ADD + OP_CME + OP_CIR + OP_CIL$

$E_CLR = OP_CLE$



ロード・クリア機能付きレジスタ

Eレジスタ駆動回路設計

$$OP_NE = \overline{OP_ADD + OP_CLE + OP_CME + OP_CIR + OP_CIL}$$

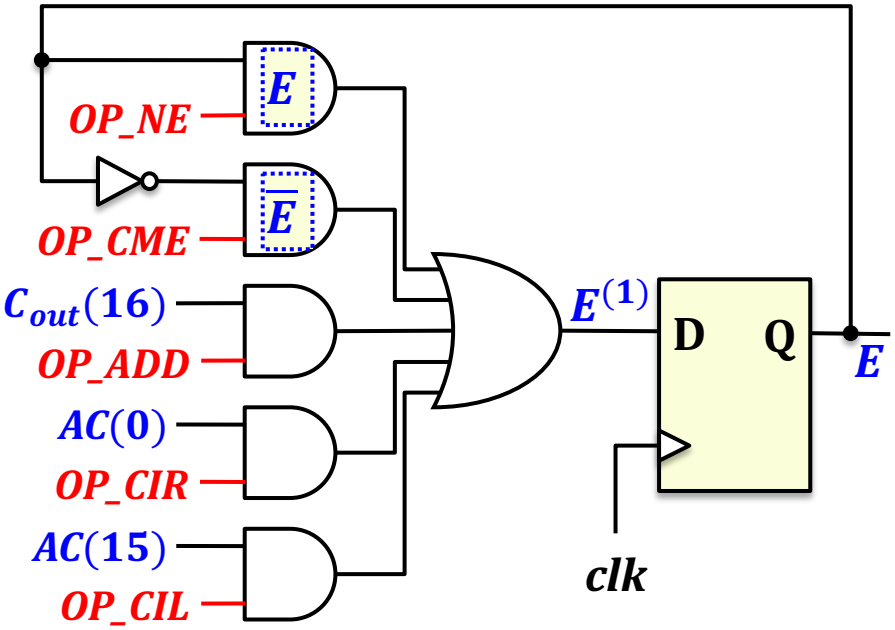
	OP_NE	OP_ADD	OP_CME	OP_CIR	OP_CIL
ADD	0	1	0	0	0
CLE	0	0	0	0	0
CME	0	0	1	0	0
CIR	0	0	0	1	0
CIL	0	0	0	0	1
E保持	1	0	0	0	0

OP_NE：Eレジスタの保持（非更新）条件

OP_CLE：クリア動作は、5つの制御信号が不活性（全て0）の時に実現される

$$r = D(7) \cdot \overline{I} \cdot T(3)$$

$OP_ADD = D(1) \cdot T(5)$：	$E \leftarrow C_{out}(16)$
$OP_CLE = r \cdot OP2(10)$：	$E \leftarrow 0$
$OP_CME = r \cdot OP2(8)$：	$E \leftarrow \overline{E}$
$OP_CIR = r \cdot OP2(7)$：	$E \leftarrow AC(0)$
$OP_CIL = r \cdot OP2(6)$：	$E \leftarrow AC(15)$
上記以外（OP_NE）：	E保持

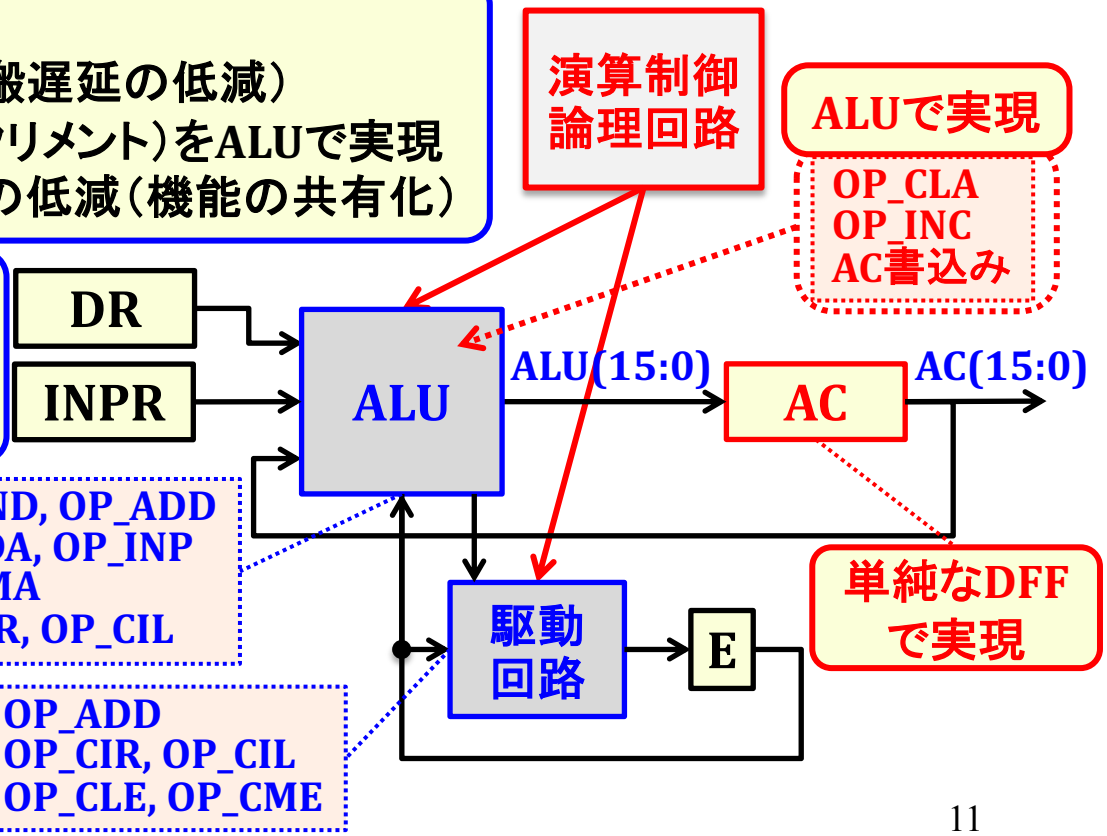


算術論理演算器設計の最適化

回路設計の「最適化」： 同一の機能を実現する回路構成は複数存在し、その中から「最適」な回路構成を導出する → 「最適性」の評価は一般に難しい問題であるため、様々なトレードオフを考慮しながら設計改良を繰り返す

設計改良ポイント：
- 加算器の高速化（キャリー伝搬遅延の低減）
- ACレジスタ機能（クリア・インクリメント）をALUで実現
- ALU内部MUXの選択入力数の低減（機能の共有化）

設計改良の目的：
- 回路遅延の低減（高速動作）
- 回路規模の低減（小面積）



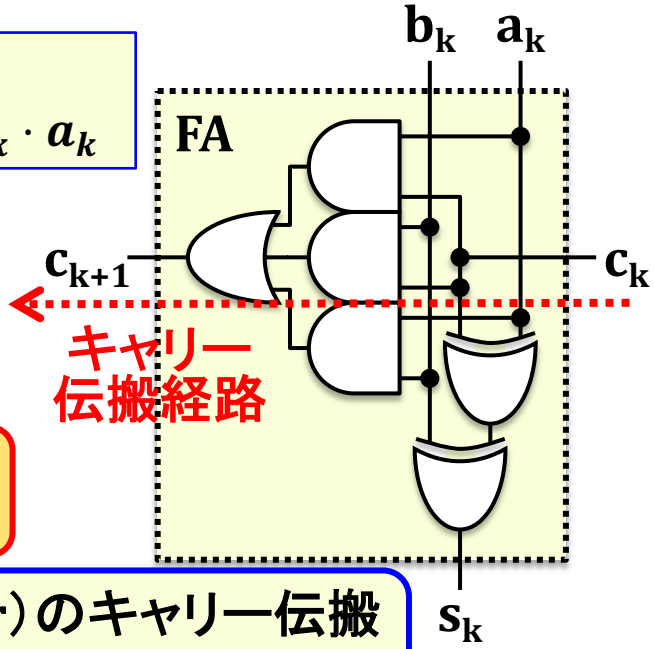
加算器構成 : Ripple-Carry Adder

a_k	b_k	c_k	c_{k+1}	s_k
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$s_k = a_k \oplus b_k \oplus c_k$$
$$c_{k+1} = a_k \cdot b_k + b_k \cdot c_k + c_k \cdot a_k$$

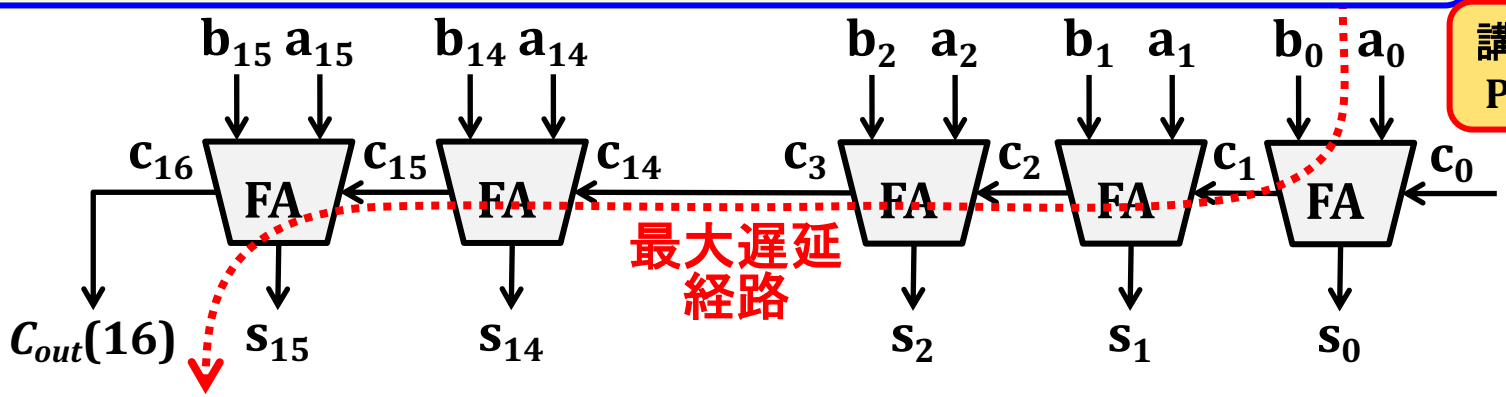
a_k , b_k : データ入力
c_k : キャリー入力
s_k : データ出力
c_{k+1} : キャリー出力

3ビット入力の「1」の個数を
2ビットで出力する論理関数



Ripple-Carry Adder : 全加算器（FA : Full Adder）のキャリー伝搬経路を直列接続 → ALU回路遅延はキャリー伝搬遅延が支配的

講義資料1 : P10 参照



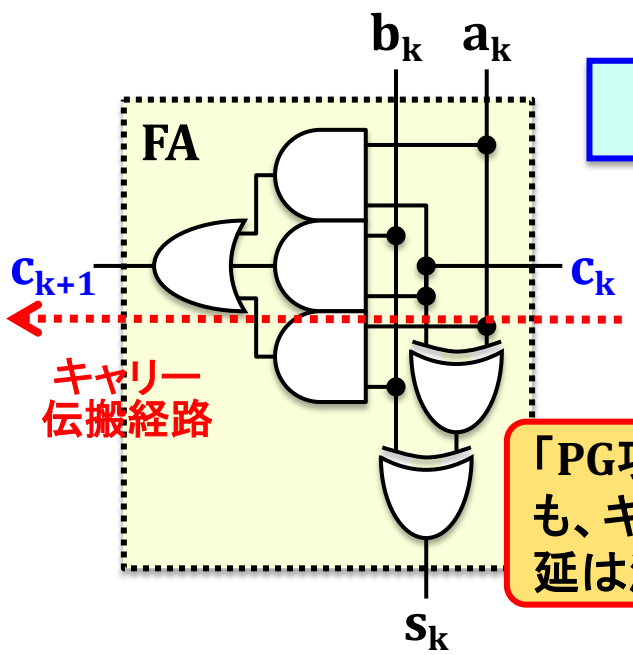
全加算器のPropagate-Generate(PG)項

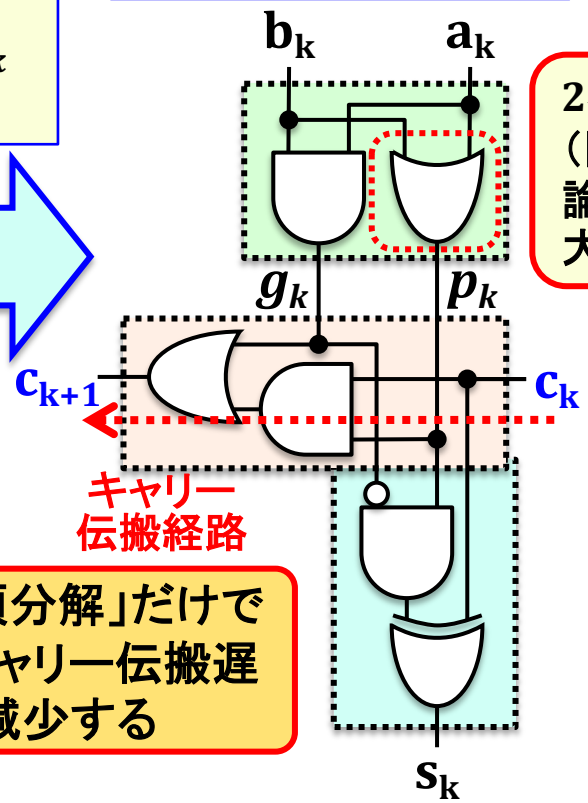
P(propagate)項：キャリー伝搬条件
G(generate)項：キャリー生成条件

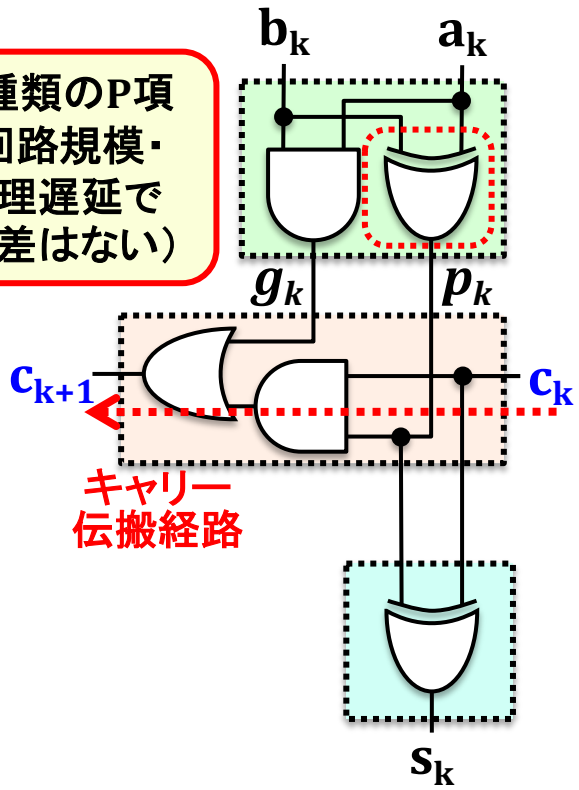
$$s_k = (a_k \oplus b_k) \oplus c_k$$
$$= \left((a_k + b_k) \cdot \overline{(a_k \cdot b_k)}\right) \oplus c_k$$
$$c_{k+1} = a_k \cdot b_k + (a_k + b_k) \cdot c_k$$
$$= a_k \cdot b_k + (a_k \oplus b_k) \cdot c_k$$

$$p_k = a_k + b_k$$
$$g_k = a_k \cdot b_k$$
$$s_k = (p_k \cdot \overline{g_k}) \oplus c_k$$
$$c_{k+1} = g_k + p_k \cdot c_k$$

$$p_k = a_k \oplus b_k$$
$$g_k = a_k \cdot b_k$$
$$s_k = p_k \oplus k$$
$$c_{k+1} = g_k + p_k \cdot c_k$$



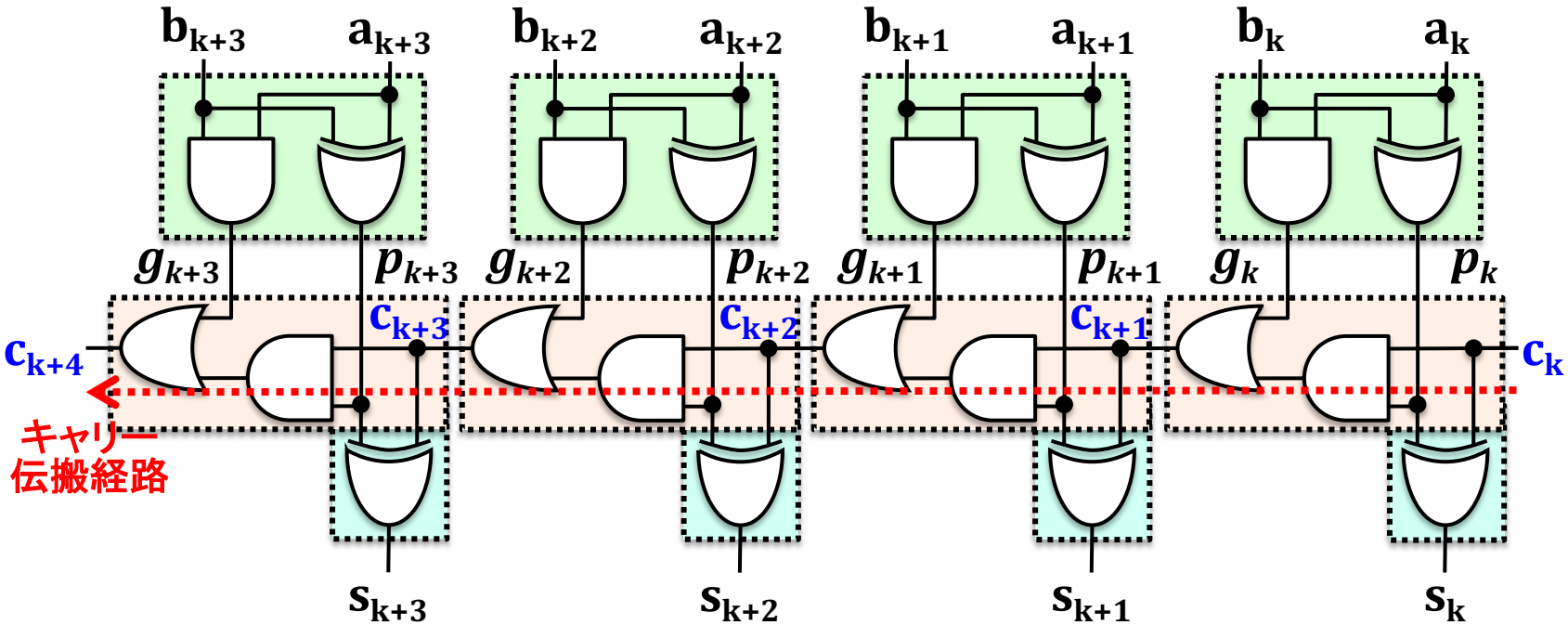




2種類のP項
（回路規模・
論理遅延で
大差はない）

「PG項分解」だけでも、キャリー伝搬遅延は減少する

PG項によるキャリー伝搬経路



$$c_{k+4} = g_{k+3} + p_{k+3} \cdot \left(g_{k+2} + p_{k+2} \cdot \left(g_{k+1} + p_{k+1} \cdot (g_k + p_k \cdot c_k)\right)\right)$$

$$c_{k+1} = g_k + p_k \cdot c_k$$

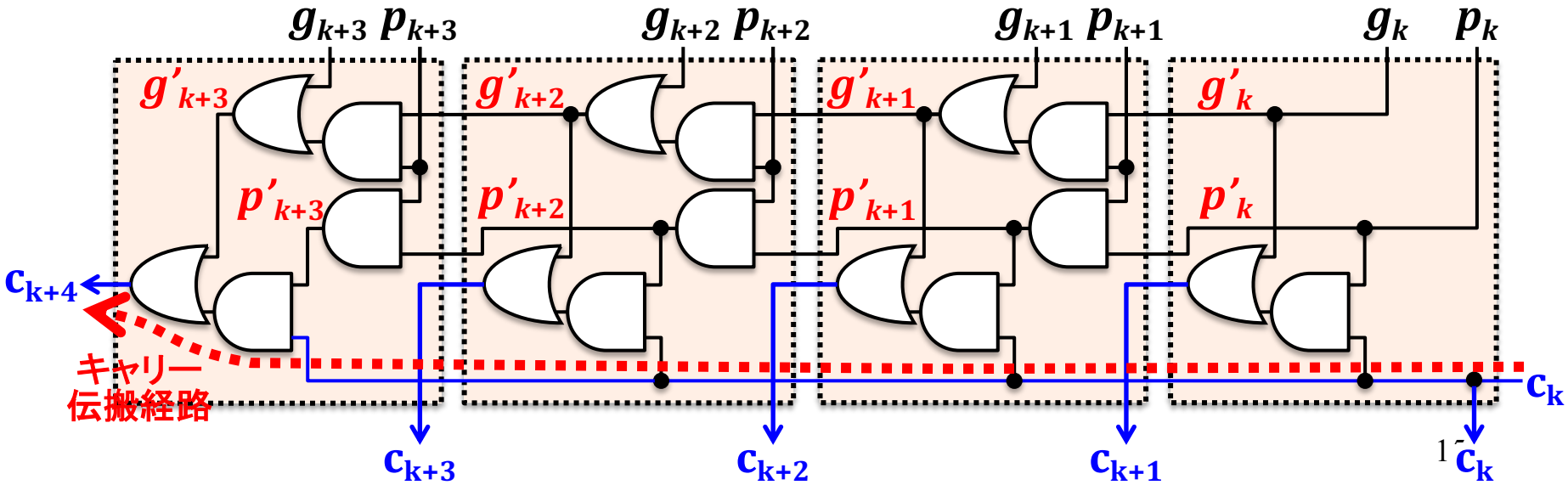
PG項の入替によりキャリー伝搬遅延をさらに低減する
→ 4-bit Carry Lookahead Generator

$$c_{k+4} = g_{k+3} + p_{k+3} \cdot \left(g_{k+2} + p_{k+2} \cdot (g_{k+1} + p_{k+1} \cdot g_k)\right) + (p_{k+3} \cdot p_{k+2} \cdot p_{k+1} \cdot p_k) \cdot c_k$$

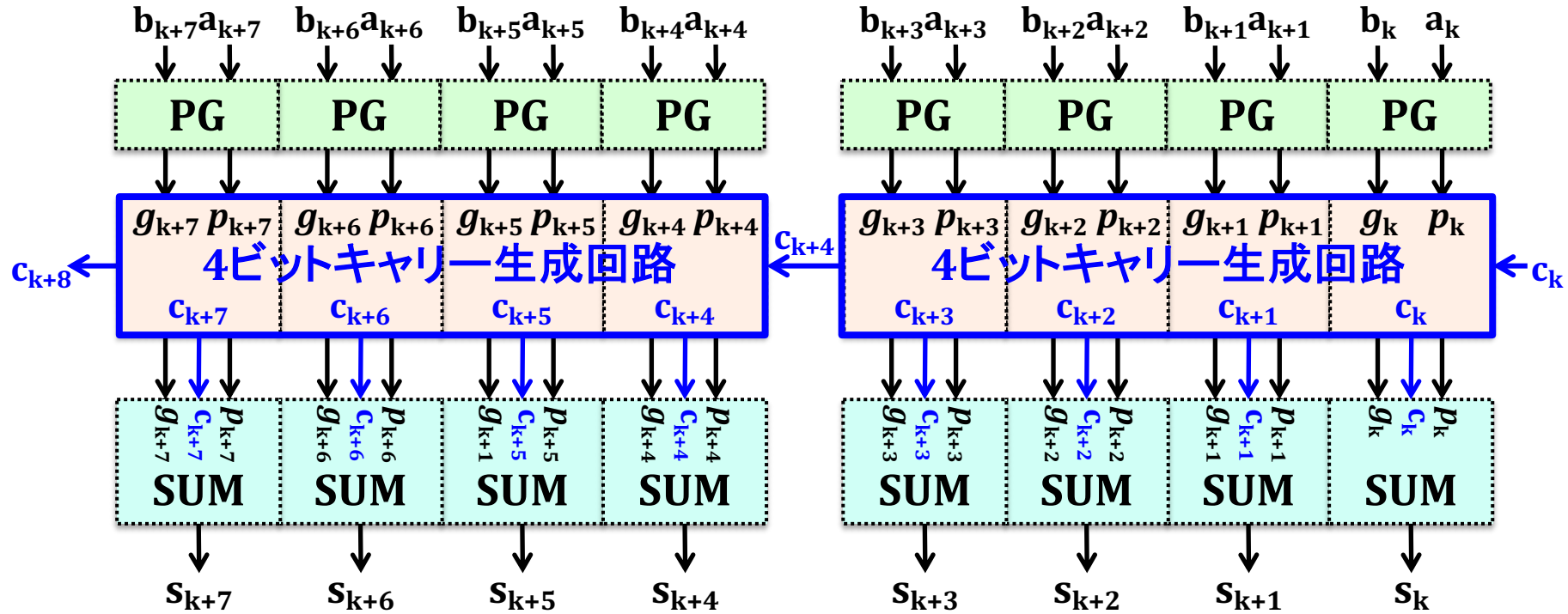
4-Bit Carry Lookahead Generator

$g'_k = g_k$	$p'_k = p_k$	$c_{k+1} = g'_k + p'_k \cdot c_k$
$g'_{k+1} = g_{k+1} + p_{k+1} \cdot g'_k$	$p'_{k+1} = p_{k+1} \cdot p'_k$	$c_{k+2} = g'_{k+1} + p'_{k+1} \cdot c_k$
$g'_{k+2} = g_{k+2} + p_{k+2} \cdot g'_{k+1}$	$p'_{k+2} = p_{k+2} \cdot p'_{k+1}$	$c_{k+3} = g'_{k+2} + p'_{k+2} \cdot c_k$
$g'_{k+3} = g_{k+3} + p_{k+3} \cdot g'_{k+2}$	$p'_{k+3} = p_{k+3} \cdot p'_{k+2}$	$c_{k+4} = g'_{k+3} + p'_{k+3} \cdot c_k$

$$c_{k+4} = \underbrace{g_{k+3} + p_{k+3} \cdot (g_{k+2} + p_{k+2} \cdot (g_{k+1} + p_{k+1} \cdot g_k))}_{g'_{i+3}} + \underbrace{(p_{k+3} \cdot p_{k+2} \cdot p_{k+1} \cdot p_k)}_{p'_{i+3}} \cdot c_k$$



Carry Lookahead Adder構成



PG

$$p_k = a_k + b_k$$
$$g_k = a_k \cdot b_k$$

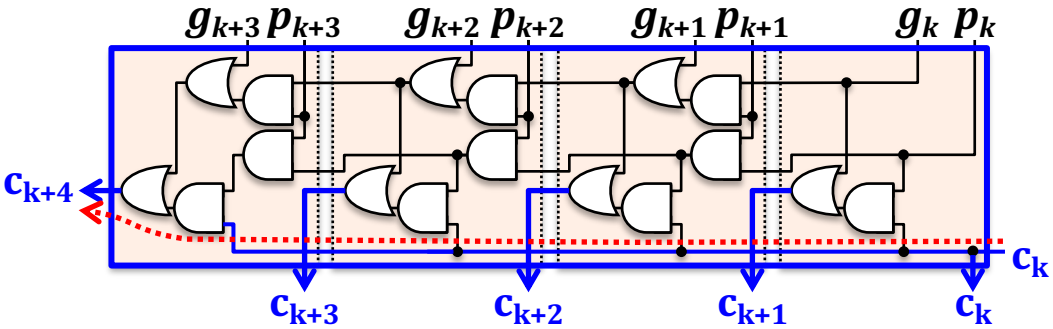
$$p_k = a_k \oplus b_k$$
$$g_k = a_k \cdot b_k$$

SUM

$$s_k = (p_k \cdot \overline{g_k}) \oplus c_k$$

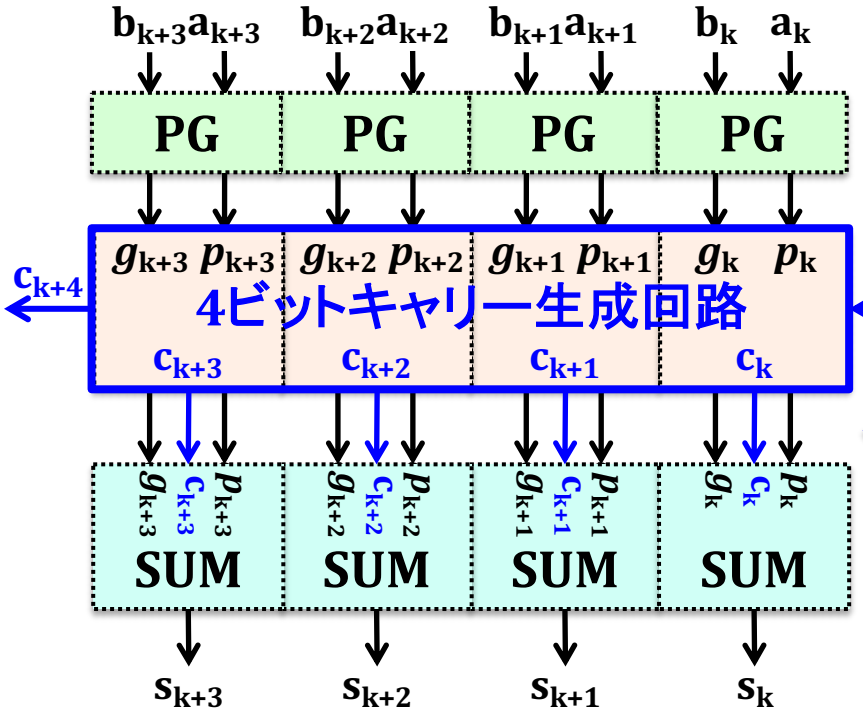
$$s_k = p_k \oplus c_k$$

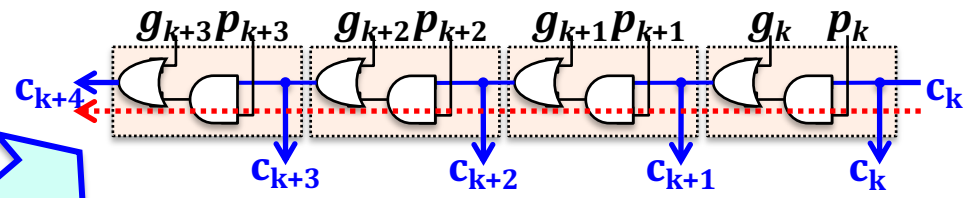
2種類のP項



Carry Lookahead型キャリー生成回路

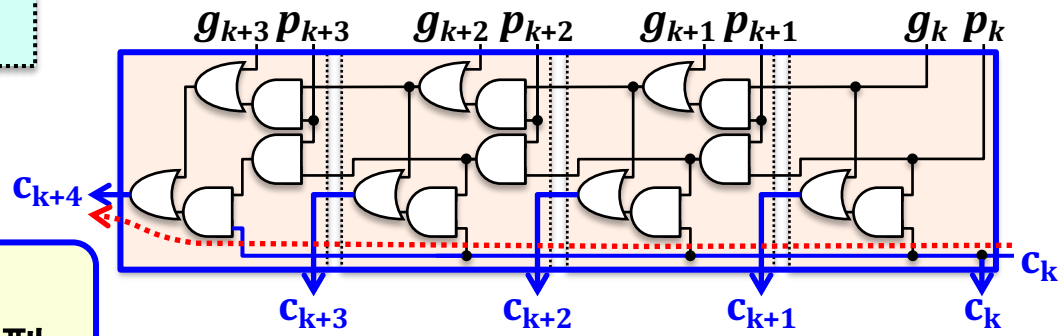
加算器構成比較





Ripple-carry型キャリー生成回路
（回路面積小、キャリー伝搬遅延大）

トレードオフ関係



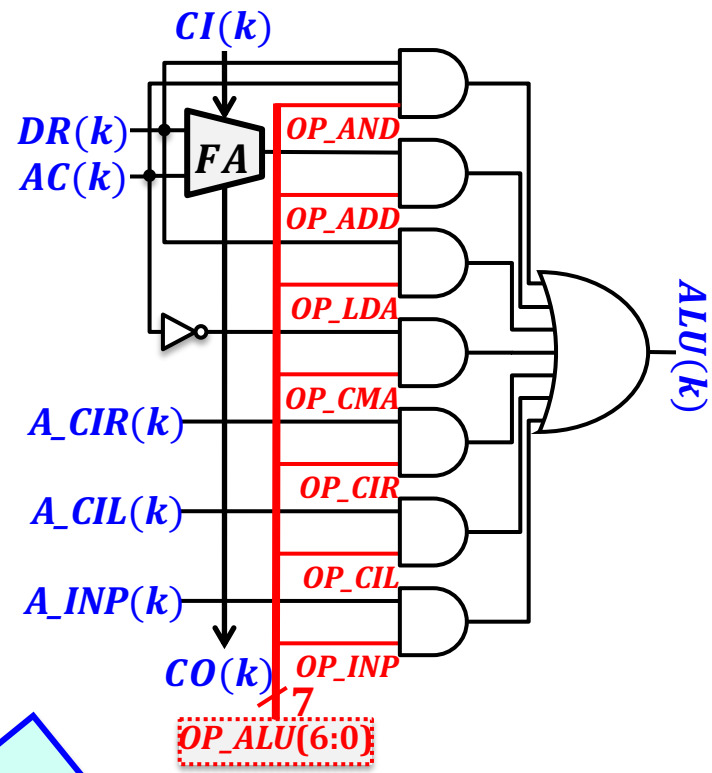
Carry Lookahead型キャリー生成回路
（回路面積大、キャリー伝搬遅延小）

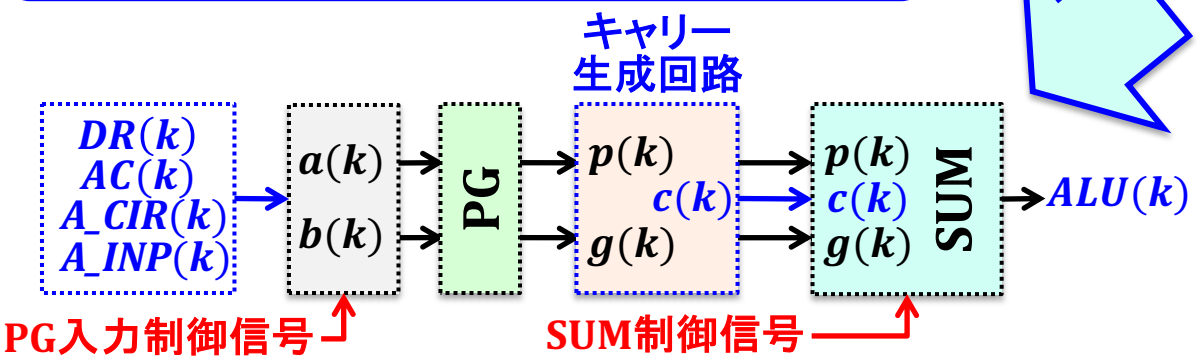
全加算器（FA）のPG項分解：Carry Lookahead型で必須、Ripple-Carry型でもキャリー伝搬遅延を低減のメリット

ALUビットスライス回路構成改良

ALU設計改良ポイント：
- 全加算器（FA）をPG項・SUM項に分解
  → キャリー伝搬遅延の低減
- 加算器の機能共有化
- ACクリア・ロード機能をALUで実現

加算器の機能共有化方法:
- 全加算器の片方の入力を0にすることで、他方の入力を出力（転送）する機能を追加（LDA・CIR・INP・CLA・AC保持）
- 16ビット加算器でADD・CIL・INCを実現
- SUM回路でAND・CMA機能を追加





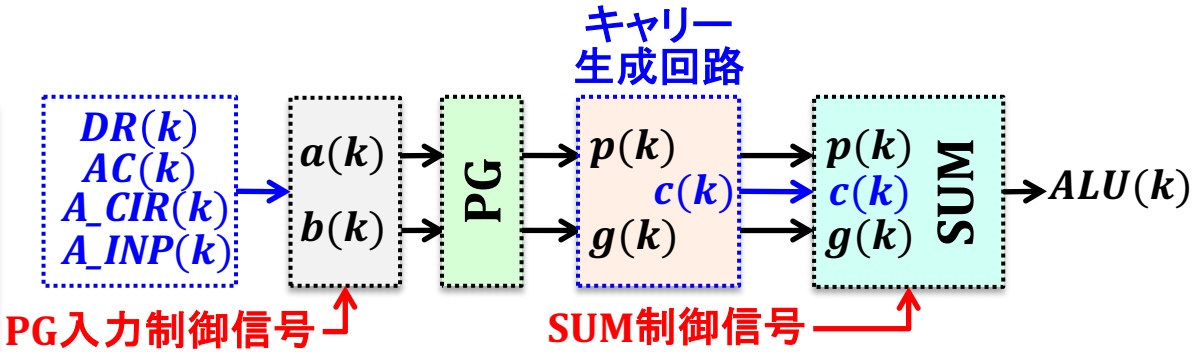
ALU改良前：
- 7出力のMUX選択
- AC保持・クリア・インクリメント機能なし

加算器の機能共有化

$A_CIR = \{E, AC(15:1)\}$

$A_INP = \{AC(15:8), INPR\}$

$C_{in} = CI(0)$（16ビット加算器の最下位ビットキャリー入力）



$b(k) = 0, C_{in} = 0 \Rightarrow c(k) = 0$ を利用

$$p(k) = a(k) \oplus b(k)$$
$$g(k) = a(k) \cdot b(k)$$

$$s(k) = p(k) \oplus c(k)$$

$$ALU(k) = \begin{cases} s(k) & \text{(加算・転送)} \\ g(k) & \text{(AND)} \\ p(k) \oplus 1 & \text{(CMA)} \end{cases}$$

$CIL : ALU = AC + AC + E$

$INC : ALU = AC + 0 + 1$

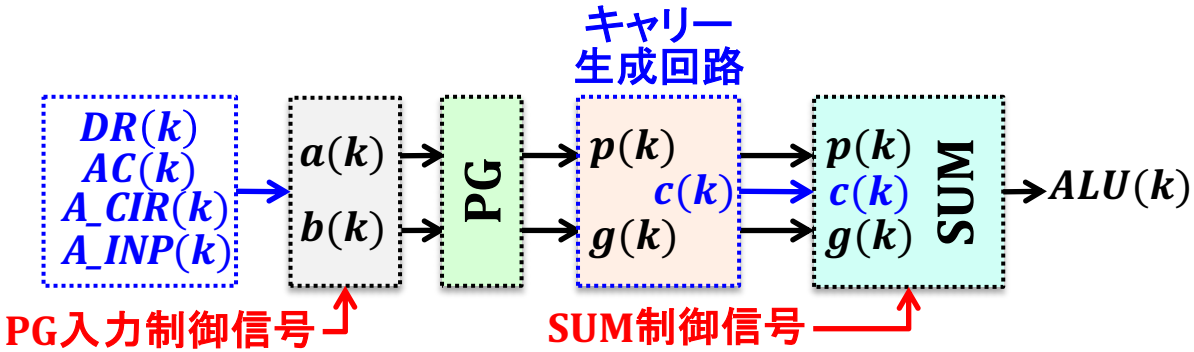
$CMA : ALU = \overline{AC} = AC \oplus 1$

命令	$a(k)$	$b(k)$	C_{in}	出力：$ALU(k)$
LDA	$DR(k)$	0	0	$s(k)$（a入力転送）
CIR	$A_CIR(k)$	0	0	$s(k)$（a入力転送）
INP	$A_INP(k)$	0	0	$s(k)$（a入力転送）
CLA	0	0	0	$s(k)$（0転送）
AC保持	$AC(k)$	0	0	$s(k)$（a入力転送）
ADD	$DR(k)$	$AC(k)$	0	$s(k)$（加算）
CIL	$AC(k)$	$AC(k)$	E	$s(k)$（加算）
INC	$AC(k)$	0	1	$s(k)$（加算）
AND	$DR(k)$	$AC(k)$	*	$g(k)$（AND）
CMA	$AC(k)$	0	*	$p(k) \oplus 1$（CMA）

ALU入力制御信号

ACを変更する命令動作

$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CMA = r \cdot OP2(9)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$



C_{in}：16ビット加算器の最下位ビットキャリー入力

$A_IN(3:0)$	$a(k)$
0001	$DR(k)$
0010	$A_CIR(k)$
0100	$A_INP(k)$
1000	$AC(k)$
0000	00..00

B_IN	$b(k)$
1	$AC(k)$
0	00..00

$C_IN(1:0)$	C_{in}
01	E
10	1
00	0

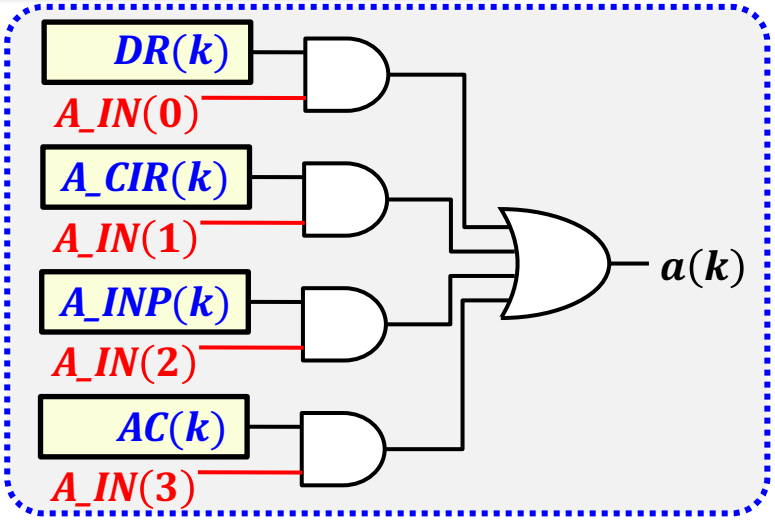
命令	$a(k)$	$b(k)$	C_{in}	出力：$ALU(k)$
LDA	$DR(k)$	0	0	$s(k)$（a入力転送）
CIR	$A_CIR(k)$	0	0	$s(k)$（a入力転送）
INP	$A_INP(k)$	0	0	$s(k)$（a入力転送）
CLA	0	0	0	$s(k)$（0転送）
AC保持	$AC(k)$	0	0	$s(k)$（a入力転送）
ADD	$DR(k)$	$AC(k)$	0	$s(k)$（加算）
CIL	$AC(k)$	$AC(k)$	E	$s(k)$（加算）
INC	$AC(k)$	0	1	$s(k)$（加算）
AND	$DR(k)$	$AC(k)$	*	$g(k)$（AND）
CMA	$AC(k)$	0	*	$p(k) \oplus 1$（CMA）

ALU入力制御論理設計

ACを変更する命令動作

$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CMA = r \cdot OP2(9)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$

$A_IN(3:0)$	$a(k)$	制御信号論理式
0001	$DR(k)$	$A_IN(0) = OP_LDA + OP_ADD + OP_AND$
0010	$A_CIR(k)$	$A_IN(1) = OP_CIR$
0100	$A_INP(k)$	$A_IN(2) = OP_INP$
1000	$AC(k)$	$A_IN(3) = OP_LDA + OP_ADD + OP_AND + OP_CIR + OP_INP + OP_CLA$
0000	00..00	$\overline{A_IN(0) + A_IN(1) + A_IN(2) + A_IN(3)} = OP_CLA$



命令	$a(k)$	$b(k)$	C_{in}	出力：$ALU(k)$
LDA	$DR(k)$	0	0	$s(k)$（a入力転送）
CIR	$A_CIR(k)$	0	0	$s(k)$（a入力転送）
INP	$A_INP(k)$	0	0	$s(k)$（a入力転送）
CLA	0	0	0	$s(k)$（0転送）
AC保持	$AC(k)$	0	0	$s(k)$（a入力転送）
ADD	$DR(k)$	$AC(k)$	0	$s(k)$（加算）
CIL	$AC(k)$	$AC(k)$	E	$s(k)$（加算）
INC	$AC(k)$	0	1	$s(k)$（加算）
AND	$DR(k)$	$AC(k)$	*	$g(k)$（AND）
CMA	$AC(k)$	0	*	$p(k) \oplus 1$（CMA）

ALU入力制御論理設計

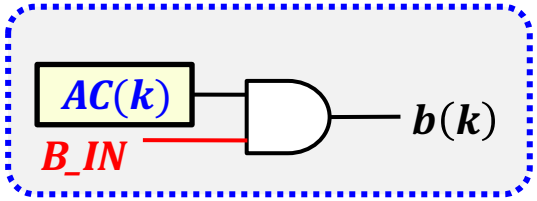
ACを変更する命令動作

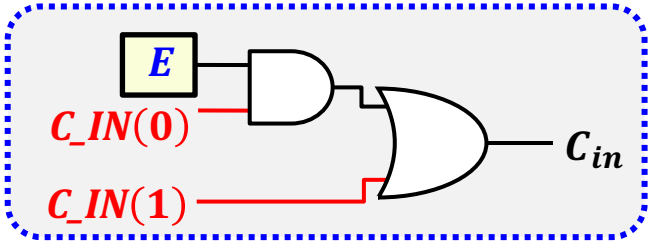
$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CMA = r \cdot OP2(9)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$

B_IN	$b(k)$	制御信号論理式
1	$AC(k)$	$B_IN = OP_ADD + OP_CIL + OP_AND$
0	00..00	$\overline{OP_ADD + OP_CIL + OP_AND}$

$C_IN(1:0)$	C_{in}	制御信号論理式
01	E	$C_IN(0) = OP_CIL$
10	1	$C_IN(1) = OP_INC$
00	0	$\overline{OP_CIL + OP_INC}$

C_{in} : 16ビット加算器の最下位ビットキャリー入力





命令	$a(k)$	$b(k)$	C_{in}	出力 : $ALU(k)$
LDA	$DR(k)$	0	0	$s(k)$（a入力転送）
CIR	$A_CIR(k)$	0	0	$s(k)$（a入力転送）
INP	$A_INP(k)$	0	0	$s(k)$（a入力転送）
CLA	0	0	0	$s(k)$（0転送）
AC保持	$AC(k)$	0	0	$s(k)$（a入力転送）
ADD	$DR(k)$	$AC(k)$	0	$s(k)$（加算）
CIL	$AC(k)$	$AC(k)$	E	$s(k)$（加算）
INC	$AC(k)$	0	1	$s(k)$（加算）
AND	$DR(k)$	$AC(k)$	*	$g(k)$（AND）
CMA	$AC(k)$	0	*	$p(k) \oplus 1$（CMA）

ALU出力制御論理設計

ACを変更する命令動作

$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CMA = r \cdot OP2(9)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$

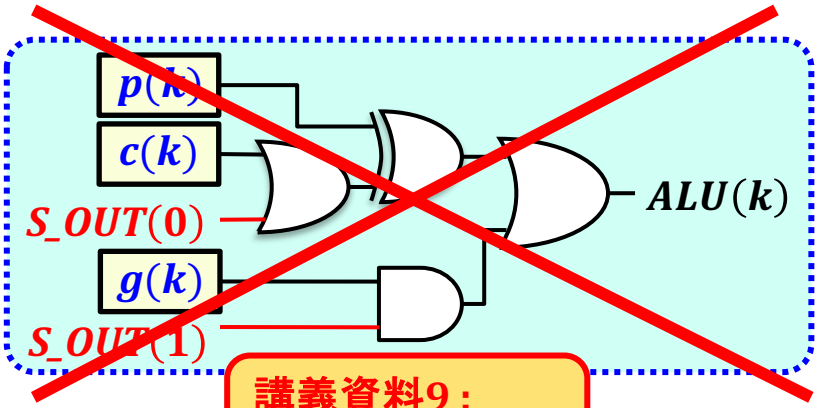
$S_OUT(1:0)$	$ALU(k)$	制御信号論理式
01	$p(k) \oplus 1$	$S_OUT(0) = OP_CMA$
10	$g(k)$	$S_OUT(1) = OP_AND$
00	$p(k) \oplus c(k)$	（加算・転送）

$$ALU(k) = \begin{cases} s(k) & \text{(加算・転送)} \\ g(k) & \text{(AND)} \\ p(k) \oplus 1 & \text{(CMA)} \end{cases}$$

$$p(k) = a(k) \oplus b(k)$$
$$g(k) = a(k) \cdot b(k)$$

$$s(k) = p(k) \oplus c(k)$$

$$CMA : ALU = \overline{AC} = AC \oplus 1$$



講義資料9：
訂正（次ページ）

命令	$a(k)$	$b(k)$	C_{in}	出力：$ALU(k)$
LDA	$DR(k)$	0	0	$s(k)$（a入力転送）
CIR	$A_CIR(k)$	0	0	$s(k)$（a入力転送）
INP	$A_INP(k)$	0	0	$s(k)$（a入力転送）
CLA	0	0	0	$s(k)$（0転送）
AC保持	$AC(k)$	0	0	$s(k)$（a入力転送）
ADD	$DR(k)$	$AC(k)$	0	$s(k)$（加算）
CIL	$AC(k)$	$AC(k)$	E	$s(k)$（加算）
INC	$AC(k)$	0	1	$s(k)$（加算）
AND	$DR(k)$	$AC(k)$	*	$g(k)$（AND）
CMA	$AC(k)$	0	*	$p(k) \oplus 1$（CMA）

ALU出力制御論理設計

ACを変更する命令動作

$OP_AND = D(0) \cdot T(5)$
$OP_ADD = D(1) \cdot T(5)$
$OP_LDA = D(2) \cdot T(5)$
$OP_CLA = r \cdot OP2(11)$
$OP_CMA = r \cdot OP2(9)$
$OP_CIR = r \cdot OP2(7)$
$OP_CIL = r \cdot OP2(6)$
$OP_INC = r \cdot OP2(5)$
$OP_INP = p \cdot OP2(11)$

$S_OUT(1:0)$	$ALU(k)$	制御信号論理式
01	$p(k) \oplus 1$	$S_OUT(0) = OP_CMA$
10	$g(k)$	$S_OUT(1) = OP_AND$
00	$p(k) \oplus c(k)$	（加算・転送）

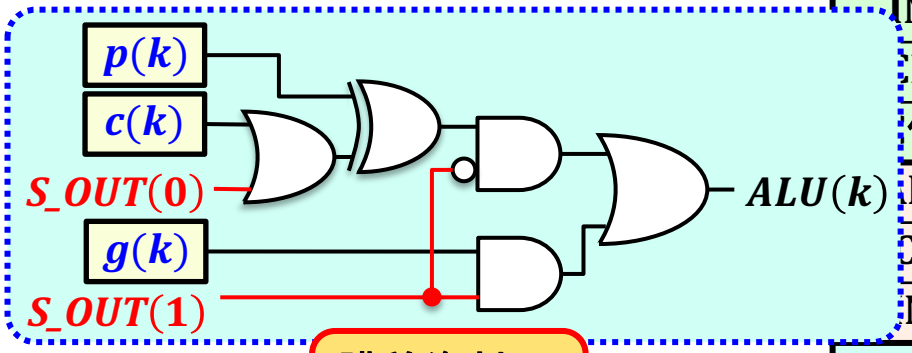
$$ALU(k) = \begin{cases} s(k) & \text{(加算・転送)} \\ g(k) & \text{(AND)} \\ p(k) \oplus 1 & \text{(CMA)} \end{cases}$$

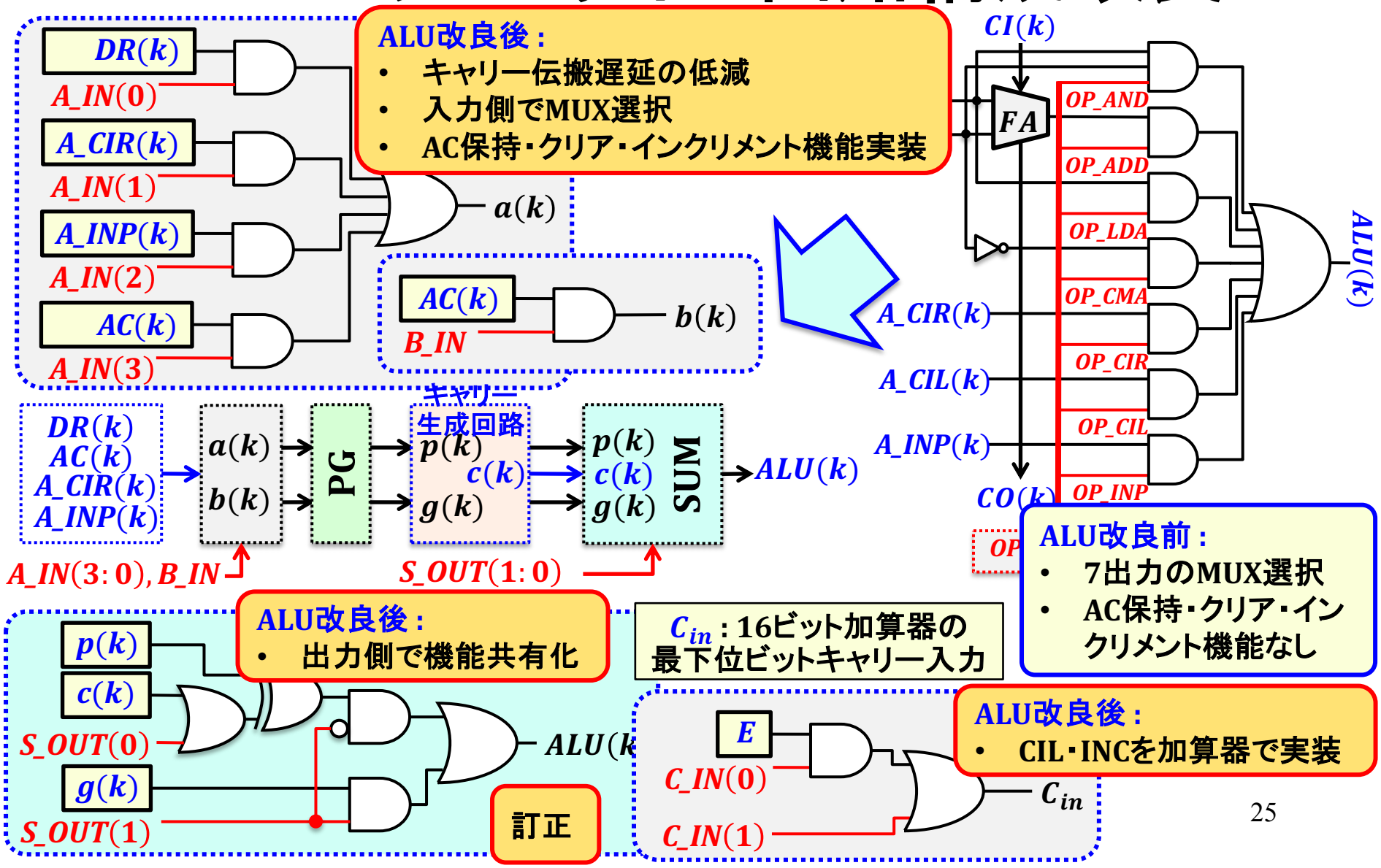
$$p(k) = a(k) \oplus b(k)$$
$$g(k) = a(k) \cdot b(k)$$

$$s(k) = p(k) \oplus c(k)$$

$$CMA : ALU = \overline{AC} = AC \oplus 1$$

命令	$a(k)$	$b(k)$	C_{in}	出力 : $ALU(k)$
LDA	$DR(k)$	0	0	$s(k)$（a入力転送）
CIR	$A_CIR(k)$	0	0	$s(k)$（a入力転送）
INP	$A_INP(k)$	0	0	$s(k)$（a入力転送）
LA	0	0	0	$s(k)$（0転送）
保持	$AC(k)$	0	0	$s(k)$（a入力転送）
DD	$DR(k)$	$AC(k)$	0	$s(k)$（加算）
IL	$AC(k)$	$AC(k)$	E	$s(k)$（加算）
NC	$AC(k)$	0	1	$s(k)$（加算）
AND	$DR(k)$	$AC(k)$	*	$g(k)$（AND）
CMA	$AC(k)$	0	*	$p(k) \oplus 1$（CMA）



講義資料9：
訂正

ALUビットスライス回路構成改良