Chapter 2
Modulo Addition and Subtraction

In this Chapter, the basic operations of modulo addition and subtraction are considered. Both the cases of general moduli and specific moduli of the form 2^n-1 and 2^n+1 are considered in detail. The case with moduli of the form 2^n+1 can benefit from the use of diminished-1 arithmetic. Multi-operand modulo addition also is discussed.

2.1 Adders for General Moduli

The modulo addition of two operands A and B can be implemented using the architectures of Figure 2.1a and b [1, 2]. Essentially, first $A+B$ is computed and then m is subtracted from the result to find whether the result is larger than m or not. (Note that TC stands for two's complement.) Then using a 2:1 multiplexer, either $(A+B)$ or $(A+B-m)$ is selected. Thus, the computation time is that of one n-bit addition, one $(n+1)$-bit addition and delay of a multiplexer. On the other hand, in the architecture of Figure 2.2b, both $(A+B)$ and $(A+B-m)$ are computed in parallel and one of the outputs is selected using a 2:1 multiplexer depending on the sign of $(A+B-m)$. Note that a carry-save adder (CSA) stage is needed for computing $(A+B-m)$ which is followed by a carry propagate adder (CPA). Thus, the area is more than that of Figure 2.2a, but the addition time is less. The area A and computation time Δ for both the techniques can be found for n-bit operands assuming that a CPA is used as

P.V. Ananda Mohan, *Residue Number Systems*, DOI 10.1007/978-3-319-41385-3_2

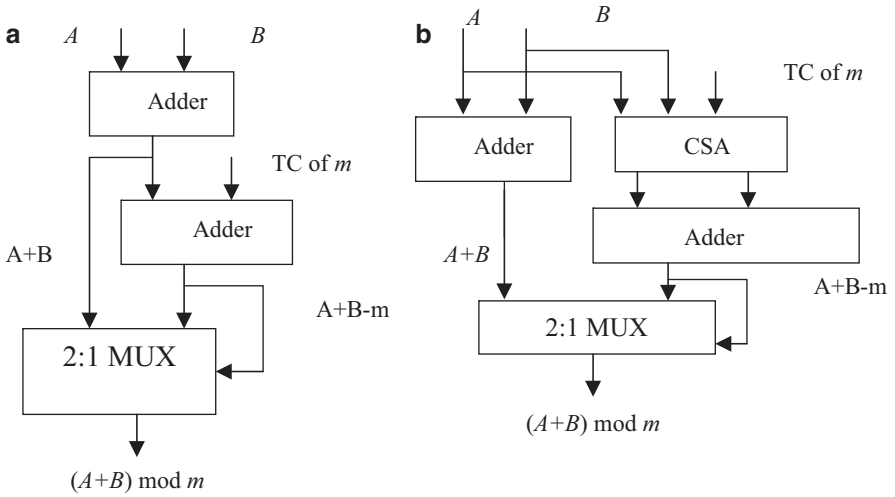

Figure 2.1 Modulo adder architectures: (**a**) sequential (**b**) parallel

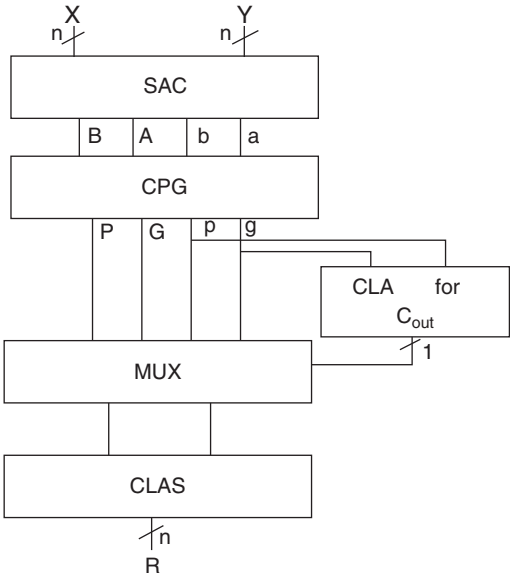

Figure 2.2 Modular adder due to Hiasat (adapted from [6] ©IEEE2002)

$$\begin{aligned} A_{cascade} &= (2n+1)A_{FA} + nA_{2:1MUX} + nA_{INV}, \quad \Delta_{cascade} = (2n+1)\Delta_{FA} + \Delta_{2:1MUX} + \Delta_{INV} \\ A_{Parallel} &= (3n+2)A_{FA} + nA_{2:1MUX} + nA_{INV}, \quad \Delta_{parallel} = (n+2)\Delta_{FA} + \Delta_{2:1MUX} + \Delta_{INV} \end{aligned} \tag{2.1}$$

where Δ_{FA}, $\Delta_{2:1MUX}$, and Δ_{INV} are the delays and A_{FA}, $A_{2:1MUX}$ and A_{INV} are the areas of a full-adder, 2:1 Multiplexer and an inverter, respectively. On the other

hand, by using VLSI adders with regular layout e.g. Brent−Kung adder [3], the area and delay requirements will be as follows:

$$\begin{aligned} &A_{cascade} = 2n(\log_2 n + 1)A_{FA} + nA_{2:1MUX} + nA_{INV}, \quad \Delta_{cascade} = 2(\log_2 n + 1)\Delta_{FA} + \Delta_{INV}, \\ &A_{Parallel} = (n + 1 + \log_2 n + \log_2(n + 1) + 2)A_{FA} + nA_{2:1MUX} + nA_{INV}, \\ &\Delta_{parallel} = ((\log_2 n + 1) + 2)\Delta_{FA} + \Delta_{2:1MUX} + \Delta_{INV} \end{aligned} \tag{2.2}$$

Subtraction is similar to the addition operation wherein $(A-B)$ and $(A-B+m)$ are computed sequentially or in parallel following architectures similar to Figure 2.1a and b.

Multi-operand modulo addition has been considered by several authors. Alia and Martinelli [4] have suggested the mod m addition of several operands using a CSA tree trying to keep the partial results at the output of each CSA stage within the range $(0, 2^n)$ by adding a proper value. The three-input addition in a CSA yields n-bit sum and carry vectors S and C. S is always in the range $\{0, 2^n\}$. The computation of $(2C+S)_m$ is carried out as $(2C+S)_m = L+H+2T_C+T_S = L+H+T+km$ where $k>0$ is an integer. Note that $L = 2(C-T_C)$ and $H = S-T_S$ were $T_S = s_{n-1}2^{n-1}$ and $T_C = c_{n-1}2^{n-1} + c_{n-2}2^{n-2}$. Thus, using s_{n-1}, c_{n-1}, c_{n-2} bits, T can be obtained using a 7:1 MUX and added to L, H. Note that L is obtained from C by one bit left shift and H is obtained as $(n-1)$-bit LSB word of S.

All the operands can be added using a CSA tree and the final result $U_F = 2C_F + S_F$ is reduced using a modular reduction unit which finds U_F, U_F-m, $U_F-2\,m$ and $U_F-3\,m$ using two CLAs and based on the sign bits of the last three words, one of the answers is selected.

Elleithi and Bayoumi [5] have presented a $\theta(1)$ algorithm for multi-operand modulo addition which needs a constant time of five steps. In this technique, the two operands A and B are written in redundant form as A_1, A_2 and B_1, B_2, respectively. The first three are added in a CSA stage which will yield sum and carry vectors. These two vectors temp1 and temp2 and B_2 are added in another CSA which will yield sum and carry vectors temp3 and temp4. In the third step, to temp3 and temp4 vectors, a correction term (2^n-m) or $2(2^n-m)$ is added in another CSA stage depending on either one or both carry bits of temp1 and temp2 are 1 to result in the sum and carry vectors temp5 and temp6. Depending on the carry bit, in the next step (2^n-m) is added to yield final result in carry save form as temp7 and temp8. There will be no overflow thereafter.

Hiasat [6] has described a modulo adder architecture based on a CSA and multiplexing the carry generate and propagate signals before being driven to the carry computation unit. In this design, the output carry is predicted that could result from computation of $A+B+Z$ where $Z = 2^n-m$. If the predicted carry is 1, an adder proceeds in computing the sum $A+B+Z$. Otherwise, it computes the sum $A+B$. Note that the calculation of Sum and Carry bits in case of bit z_i being 1 or 0 is quite simple as can be seen for both these cases:

$$s_i = a_i \oplus b_i, \quad c_{i+1} = a_i b_i \quad \text{and} \quad \hat{s}_i = \overline{a_i \oplus b_i}, \quad \hat{c}_{i+1} = a_i + b_i$$

Thus, half-adder like cells which give both the outputs are used. Note that s_i, c_{i+1}, $\hat{s}_i$, $\hat{c}_{i+1}$ serve as inputs to carry propagate and generate unit which has outputs P_i, G_i, p_i, g_i corresponding to both the cases. Based on the computation of c_{out} using a CLA, a multiplexer is used to select one of these pairs to compute all the carries and the final sum. The block diagram of this adder is shown in Figure 2.2 where SAC is sum and carry unit, CPG is carry propagate generate unit, and CLA is carry look ahead unit for computing C_{out}. Then using a MUX, either P, G or p, g are selected to be added using CLA summation unit (CLAS). The CLAS unit computes all the carries and performs the summation $P_i \oplus c_i$ to produce the output R. This design leads to lower area and delay than the designs in Refs. [1, 5].

Adders for moduli (2^n-1) and (2^n+1) have received considerable attention in literature which will be considered next.

2.2 Modulo (2^n-1) Adders

Efstathiou, Nikolos and Kalamatinos [7] have described a mod (2^n-1) adder. In this design, the carry that results from addition assuming carry input is zero is taken into account in reformulating the equations to compute the sum. Consider a mod 7 adder with inputs A and B. With the usual definition of generate and propagate signals, it can be easily seen that for a conventional adder we have

$$c_0 = G_0 + P_0 c_{-1} \tag{2.3a}$$

$$c_1 = G_1 + P_1 c_0 \tag{2.3b}$$

$$c_2 = G_2 + P_2 G_1 + P_2 P_1 g_0 \tag{2.3c}$$

Substituting c_{-1} in (2.3a) with c_2 due to the end-around carry operation of a mod (2^n-1) adder, we have

$$c_0 = G_0 + P_0 G_2 + P_0 P_2 G_1 + G_0 P_2 P_1 G_0 = G_0 + P_0 G_2 + P_0 P_2 G_1 \tag{2.4}$$

$$c_1 = G_1 + P_1 G_0 + P_1 P_0 G_2 \tag{2.5a}$$

$$c_2 = G_2 + P_2 G_1 + P_2 P_1 G_o \tag{2.5b}$$

An implementation of mod 7 adder with double representation of zero (i.e. output = 7 or zero) is shown in Figure 2.3a where $s_i = P_i \oplus c_{i-1}$. A simple modification can be carried out as shown in Figure 2.3b to realize a single zero. Note that the output can be 2^n-1, if both the inputs are complements of each other. Hence, this condition can be used by computing $P = P_0 P_1 P_2 \ldots P_{n-1}$ and modifying the equations as

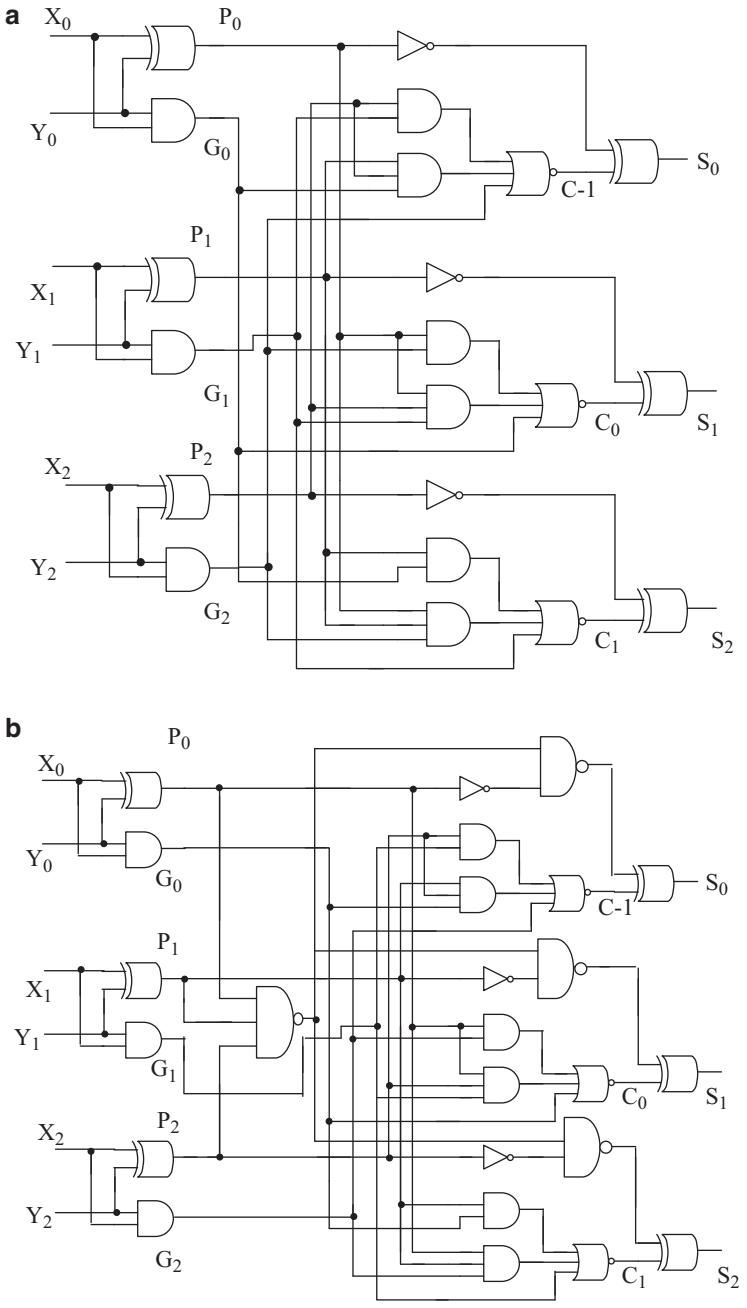

Figure 2.3 (**a**) Mod 7 adder with double representation of zero (**b**) with single representation of zero (adapted from [7] ©IEEE1994)

$$s_i = \overline{(P_i + P)} \oplus c_{i-1} \quad \text{for } 0 \leq i \leq n-1. \tag{2.6}$$

The architectures of Figure 2.3, although they are elegant, they lack regularity. Instead of using single level CLA, when the operands are large, multiple levels can also be used.

Another approach is to consider the carry propagation in binary addition as a prefix problem. Various types of parallel-prefix adders e.g. (a) Ladner−Fischer [8], (b) Kogge-Stone [9], (c) Brent−Kung [3] and (d) Knowles [10] are available in literature. Among these, type (a) requires less area but has unlimited fan out compared to type (b). But designs based on (b) are faster.

Zimmerman [11] has suggested using an additional level for adding end-around-carry for realizing a mod (2^n-1) adder (see Figure 2.4a) which needs extra hardware and more over, this carry has a large fan out thus making it slower. Kalampoukas et al. [12] have considered modulo (2^n-1) adders using parallel-prefix adders. The idea of carry recirculation at each prefix level as shown in Figure 2.4b has been employed. Here, no extra level of adders will be required, thus having minimum logic depth. In addition, the fan out requirement of the carry output is also removed. These architectures are very fast while consuming large area.

The area and delay requirements of adders can be estimated using the unit-gate model [13]. In this model, all gates are considered as a unit, whereas only exclusive-OR gate counts for two elementary gates. The model, however, ignores fan-in and fan-out. Hence, validation needs to be carried out by using static simulations. The area and delay requirements of mod (2^n-1) adder described in [12] are $3n\log n + 4n$ and $2\log n + 3$ assuming this model.

Efstathiou et al. [14] have also considered design using select-prefix blocks with the difference that the adder is divided into several small length adder blocks by proper interconnection of propagate and generate signals of the blocks. A select-prefix architecture for mod (2^n-1) adder is presented in Figure 2.5. Note that d, f and g indicate the word lengths of the three sections. It can be seen that

$$\begin{aligned}
c_{in,0} &= BG_2 + BP_2BG_1 + BP_2BP_1BG_0 \\
c_{in,1} &= c_{out,0} = BG_0 + BP_0BG_2 + BP_0BP_2BG_1 \\
c_{in,2} &= c_{out,1} = BG_1 + BP_1BG_0 + BP_1BP_0BG_2
\end{aligned}$$

where BG_i and BP_i are block generate and propagate signals outputs of each block.

Tyagi [13] has given an algorithm for selecting the lengths of the various adder blocks suitably with the aim of minimization of adder delay. Note that designs based on parallel-prefix adders are fastest but are more complex. On the other hand, CLA-based adder architecture is area effective. Select prefix-architectures achieve delay closer to parallel prefix adders and have complexity close to the best adders.

Patel et al. [15] have suggested fast parallel-prefix architectures for modulo (2^n-1) addition with a single representation of zero. In these, the sum is computed with a carry in of "1". Later, a conditional decrement operation is

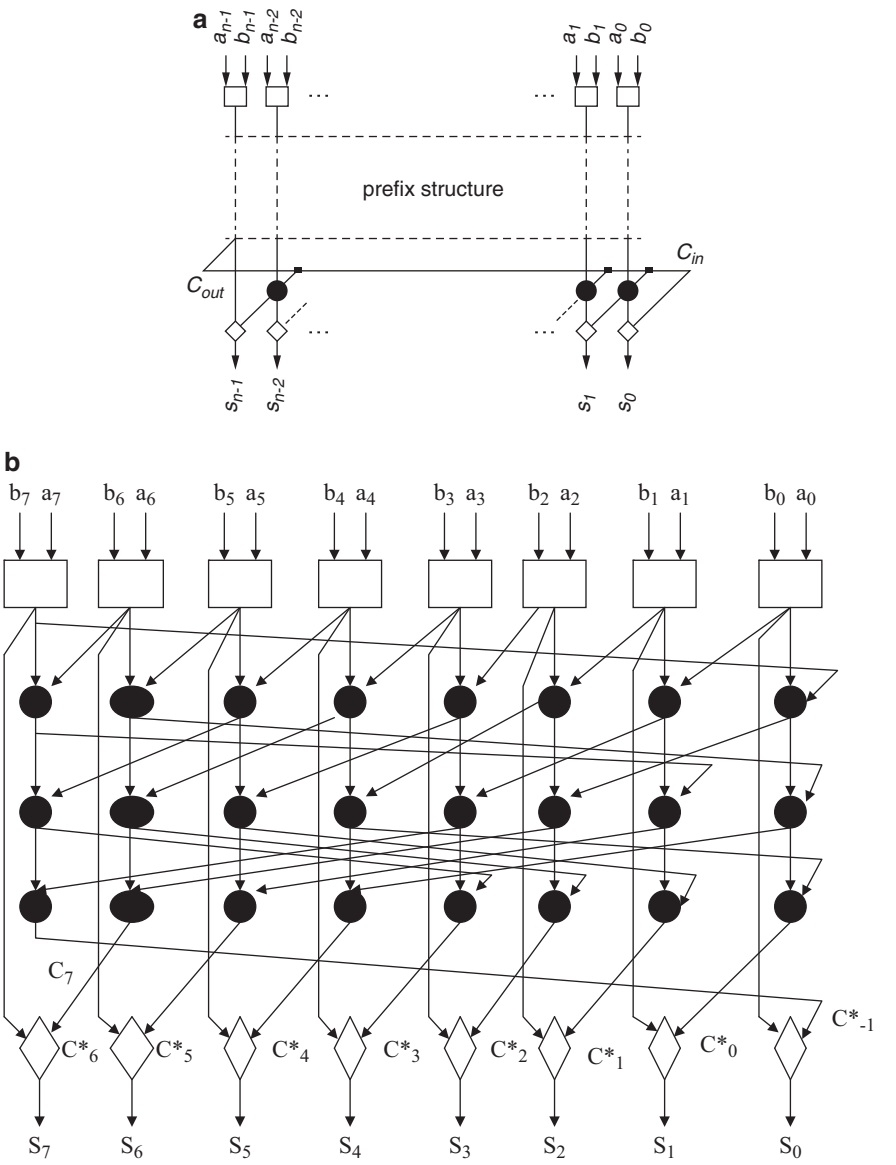

Figure 2.4 Modulo (2^n-1) adder architectures due to (**a**) Zimmermann and (**b**) modulo (2^8-1) adder due to Kalampoukas et al. ((a) adapted from [11] ©IEEE1999 and (b) adapted from [12] ©IEEE2000)

performed. However, by cyclically feeding back the carry generate and carry propagate signals at each prefix level in the adder, the authors show that significant improvement in latency is possible over existing designs.

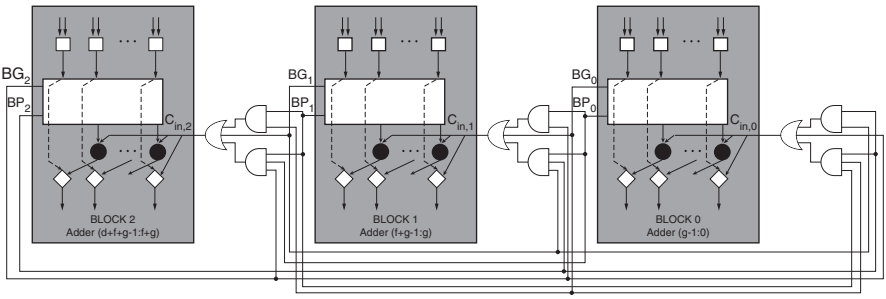

Figure 2.5 Modulo $2^{d+f+g}-1$ adder design using three blocks (adapted from [14] ©IEEE2003)

2.3 Modulo $(2^n + 1)$ Adders

Diminished-1 arithmetic is important for handling moduli of the form $2^n + 1$. This is because of the reason that this modulus channel needs one bit more word length than other channels using moduli 2^n and 2^n-1. A solution given by Liebowitz [16] is to represent the numbers still by n bits only. The diminished-1 number corresponding to normal number A in the range 1 to 2^n is represented as $d(A) = A-1$. If $A = 0$, a separate channel with one bit which is 1 is used. Another way of representing A in diminished-1 arithmetic is (A_z, A_d) where $A_z = 1$, $A_d = 0$ when $A = 2^n$, $A_z = 0$, $A_d = A-1$ otherwise. Due to this representation, some rules need to be built to perform operations in this arithmetic which are summarized below. Following the above notation, we can derive the following properties [17]:
(a) $A + B = C$ corresponds to

$$d(A + B) \;=\; (d(A) + d(B) + 1) \bmod (2^n + 1) \tag{2.7}$$

(b) Similarly, we have

$$d(A - B) = \Big(d(A) + \overline{d(B)} + 1\Big) \bmod (2^n + 1) \tag{2.8}$$

(c) It follows further that

$$d\Big(\sum\nolimits_{k=1}^{n} A_k\Big) = (d(A_1) + d(A_2) + d(A_3) + \quad \ldots \quad + d(A_k) + n - 1) \;\bmod\; (2^n + 1) \tag{2.9}$$

Next,

$$d\big(2^k A\big) \;= d(A + A + A + \ldots + A) \;=\; \big(2^k d(A) + 2^k - 1\big) \bmod (2^n + 1).$$

or

$$2^k d(A) \;=\; \big(d\big(2^k A\big) - 2^k + 1\big) \bmod (2^n + 1) \tag{2.10}$$

In order to simplify the notation, we denote a diminished-1 number using an asterisk e.g. $d(A) = A^* = A-1$.

Several mod (2^n+1) adders have been proposed in literature. In the case of diminished-1 numbers, mod (2^n+1) addition can be formulated as [11]

$$\begin{aligned} S-1 = S^* &= (A^*+B^*+1) \mod (2^n+1) \\ &= (A^*+B^*) \mod (2^n) \quad \text{if} \quad (A^*+B^*) \\ &\geq 2^n \quad \text{and} \quad (A^*+B^*+1) \quad \text{otherwise} \end{aligned} \tag{2.11}$$

where A^* and B^* are diminished-1 numbers and $S = A+B$. The addition of 1 can be carried out by inverting the carry bit C_{out} and adding in a parallel-prefix adder with $C_{in} = \overline{C_{out}}$ (see Figure 2.6):

$$(A^*+B^*+1) \mod (2^n+1) = \left(A^*+B^*+\overline{C_{out}}\right) \mod (2^n) \tag{2.12}$$

In the case of normal numbers as well [11], we have

$$S+1 = (A+B+1) \mod (2^n+1) = \left(A+B+\overline{C_{out}}\right) \mod (2^n) \tag{2.13}$$

where $S = A+B$ with the property that $(S+1)$ is computed. In the design of multipliers, this technique will be useful.

Note that diminished-1 adders have a problem of correctly interpreting the zero output since it may represent a valid zero (addition with a result of 1) or a real zero output (addition with a result zero) [14]. Consider the two examples of modulo

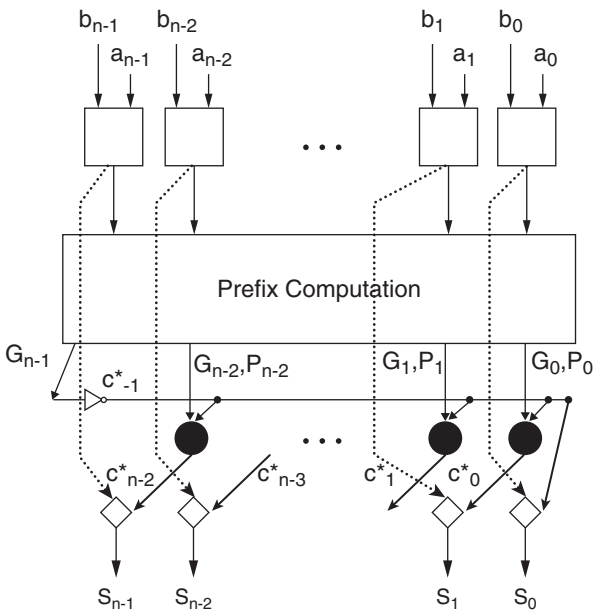

Figure 2.6 Modulo (2^n+1) adder architecture for diminished-1 arithmetic (adapted from [18] ©IEEE2002)

9 addition (a) $A=6$ and $B=4$ and (b) $C=5$ and $B=4$ using diminished-1 representation:

A^*	101	C^*	100
B^*	011	B^*	011
C_{out}	1 000	C_{out}	0 111
$\overline{C_{out}}$	0	$\overline{C_{out}}$	1
	000 Correct result		000 result indicating zero

Note that real zero occurs when the inputs are complimentary. Hence, this condition needs to be detected using logical AND of the exclusive-OR of a_i and b_i. The EXOR gates will be already present in the front-end CSA stage.

Vergos, Efstathiou and Nikolos have presented two mod (2^n+1) adder architectures [18] for diminished-1 numbers. The first one leads to CLA implementation and was derived by associating the re-entering carry equation with those producing the carries of the modulo addition similar to that for mod (2^n-1) described earlier [12]. In this architecture, both one and two level CLAs have been considered. The second architecture uses parallel-prefix adders and also was derived by re-circulation of the carries in each level of parallel-prefix structure. This architecture avoids the problem of fan-out and the additional level needed in Zimmerman's technique shown in Figure 2.6.

Efstathiou, Vergos and Nikolos [14] extended the above ideas by using select-prefix blocks which are faster than the previous ones for designing mod $(2^n \pm 1)$ adders for diminished-1 operands. Here, the lengths of the blocks can be selected appropriately as well as the number of the blocks. The derivation is similar to that for mod (2^n-1) adders with the difference that the equations contain block carry propagate, and block generate signals instead of bit level propagate and generate signals. In these, an additional level is used to add the carry after the prefix computation. A structure using two stages is presented in Figure 2.7. Note that in this case

$$c_{in,0} = (BG_1 + BP_1BG_0)'$$
$$c_{in,1} = c_{out,0} = BG_0 + BP_0BG'_1$$

These designs need lesser area than designs using parallel-prefix adders while they are slower than CLA-based designs.

Efstathiou, Vergos and Nikolos [19] have described fast parallel-prefix modulo (2^n+1) adders for two $(n+1)$-bit numbers which use two stages. The first stage computes $|X+Y+2^n-1|_{2^{n+1}}$ which has $(n+2)$ bits. If MSB of the result is zero, then 2^n+1 is added mod 2^{n+1} and the n LSBs yield the result. For computing $M = X+Y+2^n-1$, a CSA is used followed by a $(n+1)$-bit adder. The authors use parallel-prefix with fast carry increment (PPFCI) architecture and also a totally

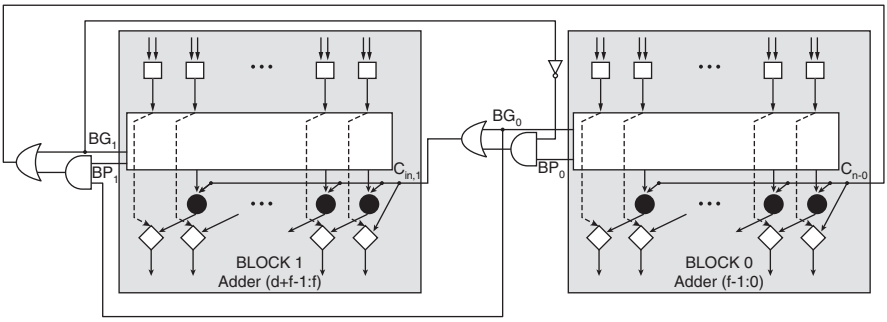

Figure 2.7 Diminished-1 modulo $(2^{d+f}+1)$ adder using two blocks (adapted from [14] ©IEEE2004)

parallel-prefix architecture. In the former, an additional stage for re-entering carry is used, whereas in the latter case, carry recirculation is done at every prefix level.

The architecture of Hiasat [6] can be extended to the case of modulus (2^n+1) in which case we have $Z=2^n-1$ and the formulae used are as follows:

$$R=|X+Y+Z|_{2^n} \quad \text{if } X+Y+Z\geq 2^{n+1} \quad \text{and} \quad R=|X+Y+Z|_{2^n}+1 \quad \text{otherwise.}$$

Note that, in this case, the added bit z_i is always 1 in all bit positions.

Vergos and Efstathiou [20] proposed an adder that caters for both weighted and diminished-1 operands. They point out that a diminished-1 adder can be used to realize a weighted adder by having a front-end inverted EAC CSA stage. Herein, $A+B$ is computed where A and B are $(n+1)$-bit numbers using a diminished-1 adder. In this design, the computation carried out is

$$|A+B|_{2^n+1}=\left||A_n+B_n+D+1|_{2^n+1}+1\right|_{2^n+1}=|Y+U+1|_{2^n+1} \qquad (2.14)$$

where Y and U are the sum and carry vector outputs of a CSA stage computing A_n+B_n+D:

$$\begin{array}{ll} carry & Y=y_{n-2}y_{n-3}.....y_o\overline{y_{n-1}} \\ sum & U=u_{n-1}u_{n-2}.....u_1u_o \end{array}$$

where $D=2^n-4+2\overline{c_{n+1}}+\overline{s_n}$. Note that A_n, B_n are the words formed by the n-bit LSBs of A and B, respectively, and s_n, c_{n+1} are the sum and carry of addition of 1-bit words a_n and b_n. It may be seen that D is the n-bit vector $11111...1\overline{c_{n+1}s_n}$.

An example will be illustrative. Consider $n=4$ and the addition of $A=16$ and $B=11$. Evidently $a_n=1$, $b_n=0$, $A_n=0$ and $B_n=11$ and $D=01110$ yielding $(16+11)_{17}=((0+11+14+1)_{17}+1)_{17}=10$. Note that the periodic property of residues mod (2^n+1) is used. The sum of the n th bits is complimented and added to get D and a correction term is added to take into account the mod (2^n+1) operation.

The mod $(2^n + 1)$ adder for weighted representation needs a diminished-1 adder and an inverted end-around-carry stage. The full adders of this CSA stage perform $(A_n + B_n + D)$ mod $(2^n + 1)$ addition. Some of the FAs have one input "1" and can thus be simplified. The outputs of this stage Y and U are fed to a diminished-1 adder to obtain $(Y + U + 1)$ mod 2^n. The architecture is presented in Figure 2.8. It can be seen that every diminished-1 adder can be used to perform weighted binary addition using an inverted EAC CSA stage in the front-end.

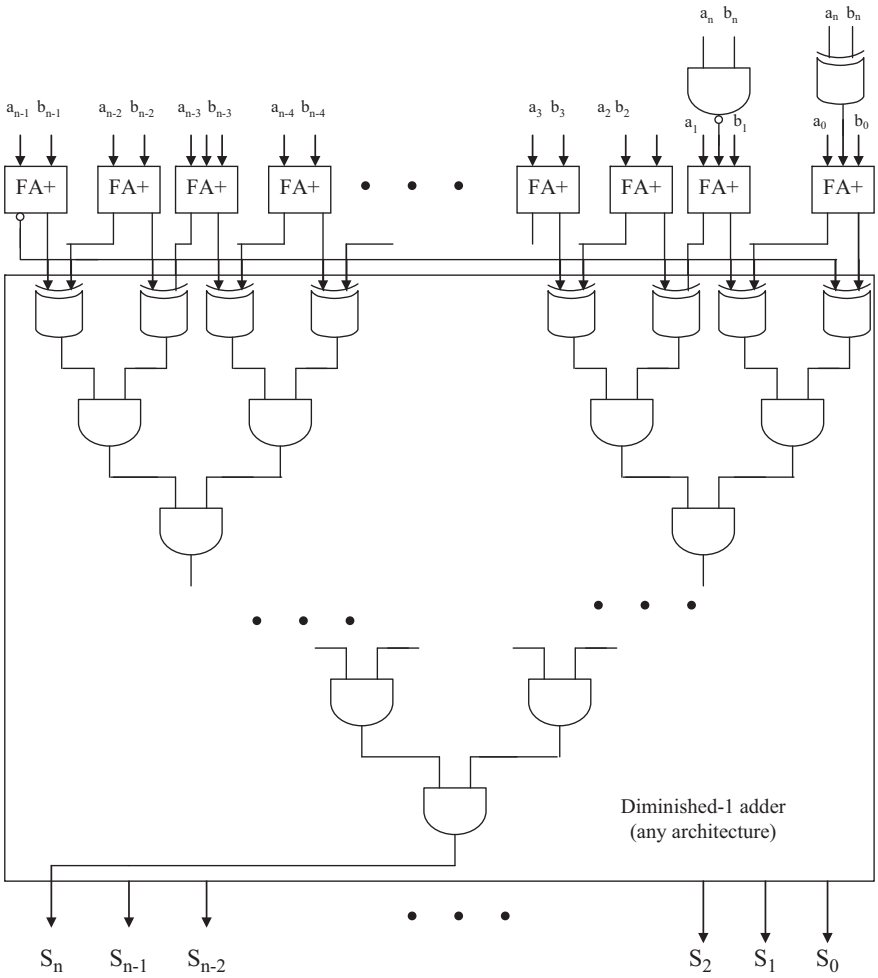

Figure 2.8 Modulo $(2^n + 1)$ adder for weighted operands built using a diminished-1 adder (adapted from [20] ©IEEE2008)

In another technique due to Vergos and Bakalis [21], first A^* and B^* are computed such that $A^*+B^*=A+B-1$ using a translator. Then, a diminished-1 adder can sum A^* and B^* such that

$$\left||A+B|_{2^n+1}\right|_{2^n} = |A^*+B^*|_{2^n} + \overline{c_{out}} \tag{2.15}$$

where c_{out} is the carry of the n-bit adder computing A^*+B^*. However, Vergos and Bakalis do not present the details of obtaining A^* and B^* using the translator. Note that in this method, the inputs are $\leq (2^n-1)$.

Lin and Sheu [22] have suggested the use of two parallel adders to find A^*+B^* and A^*+B^*+1 so that the carry of the former adder can be used to select the correct result using a multiplexer. Note that Lin and Sheu [22] have also suggested partitioning the n-bit circular carry selection (CCS) modular adder to m number of r-bit blocks similar to the select-prefix block type of design considered earlier. These need circular carry selection addition blocks and circular carry generators. Juang et al. [23] have given a corrected version of this type of mod (2^n+1) adder shown in Figure 2.9a and b. Note that this design uses a dual sum carry look ahead adder (DS-CLA). These designs are most efficient among all the mod (2^n+1) adders regarding area, time and power.

Juang et al. [24] have suggested considering $(n+1)$ bits for inputs A and B. The weighted modulo (2^n+1) sum of A and B can be expressed as

$$\begin{aligned}\left||A+B|_{2^n+1}\right|_{2^n} &= |A+B-(2^n+1)|_{2^n} \text{ if } (A+B)>2^n \\ &= |A+B-(2^n+1)|_{2^n}+1 \quad \text{otherwise}\end{aligned} \tag{2.16}$$

Thus, weighted modulo (2^n+1) addition can be obtained by subtracting the sum of A and B by (2^n+1) and using a diminished-1 adder to get the final modulo sum by making the inverted EAC as carry-in.

Denoting Y' and U' as the carry and sum vectors of the summation $A+B-(2^n+1)$, where A and B are $(n+1)$-bit words, we have

$$|A+B-(2^n+1)|_{2^n} = \left|\sum_{i=0}^{n-2}\left(2^i\left(2y_i'+u_i'\right)+2^{n-1}\left(2a_n+2b_n+a_{n-1}+b_{n-1}+1\right)\right)\right|_{2^n} \tag{2.17}$$

where

$$y_i' = a_i \vee b_i, \quad u_i' = \overline{a_i \oplus b_i}.$$

As an illustration, consider $A=16$, $B=15$ and $n=4$. We have

$$|A+B-(2^n+1)|_{2^n} = |16+15-17|_{16} = 14$$

and for $A=6$, $B=7$,

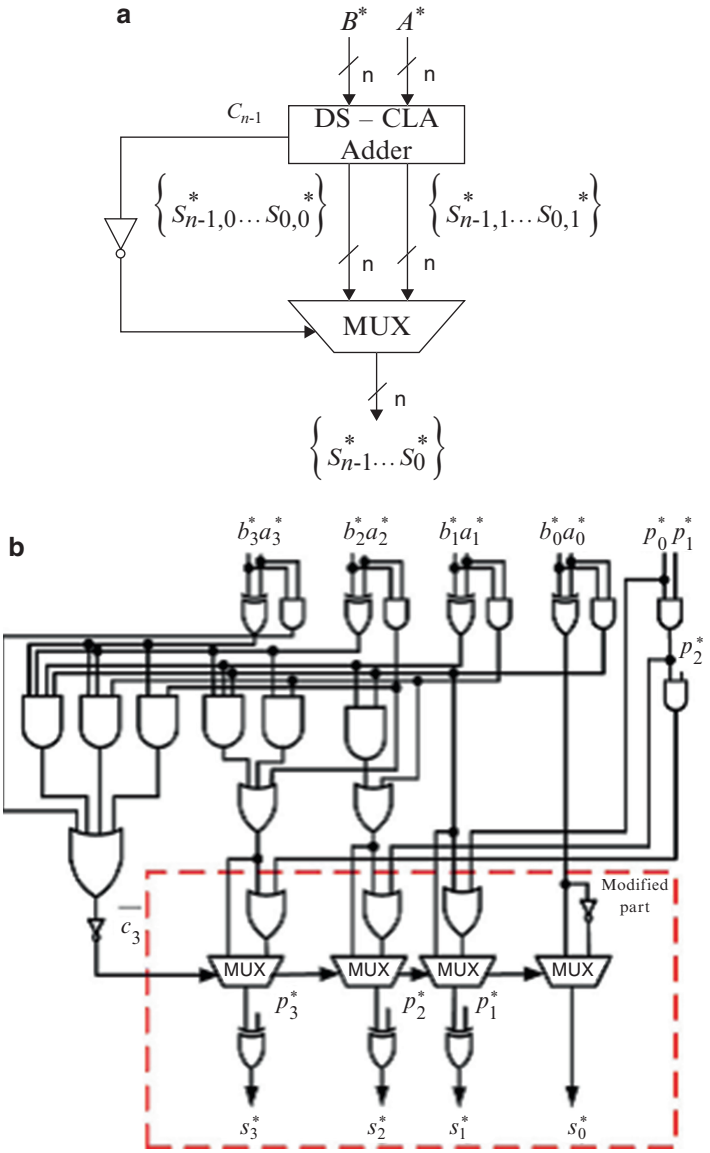

Figure 2.9 (**a**) Block diagram of CCS diminished-1 modulo $(2^n + 1)$ adder and (**b**) Logic circuit of CCS diminished-1 modulo $(2^4 + 1)$ adder ((a) adapted from [22] ©IEEE2008, (b) adapted from [23] ©IEEE2009)

$$|A+B-(2^n+1)|_{2^n} = |6+7-17|_{16}+1 = 13.$$

The multiplier of 2^{n-1} in (2.17) can be at most 5 since $0 \le A, B \le 2^n$. Since only bits n and $n-1$ are available, the authors consider the $(n+1)$-th bit to merge with C_{out}:

$$\left||A+B|_{2^n+1}\right|_{2^n} = |A+B-(2^n+1)|_{2^n} = |Y'+U'|_{2^n} + \overline{c_{out} \vee FIX} \qquad (2.18)$$

where $y'_{n-1} = a_n \vee b_n \vee a_{n-1} \vee b_{n-1}$, $u'_{n-1} = \overline{a_{n-1} \oplus b_{n-1}}$ and $FIX = a_n b_n \vee a_{n-1} b_n \vee a_n b_{n-1}$. Note that y'_{n-1} and u'_{n-1} are the values of the carry bit and sum bit produced by the addition $2a_n + 2b_n + a_{n-1} + b_{n-1} + 1$. The block diagram is presented in Figure 2.10a together with the translator in b. Note that FAF block generates y'_{n-1}, u'_{n-1} and FA blocks generate y'_i, u'_i for $i = 0,1,\ldots, n-2$

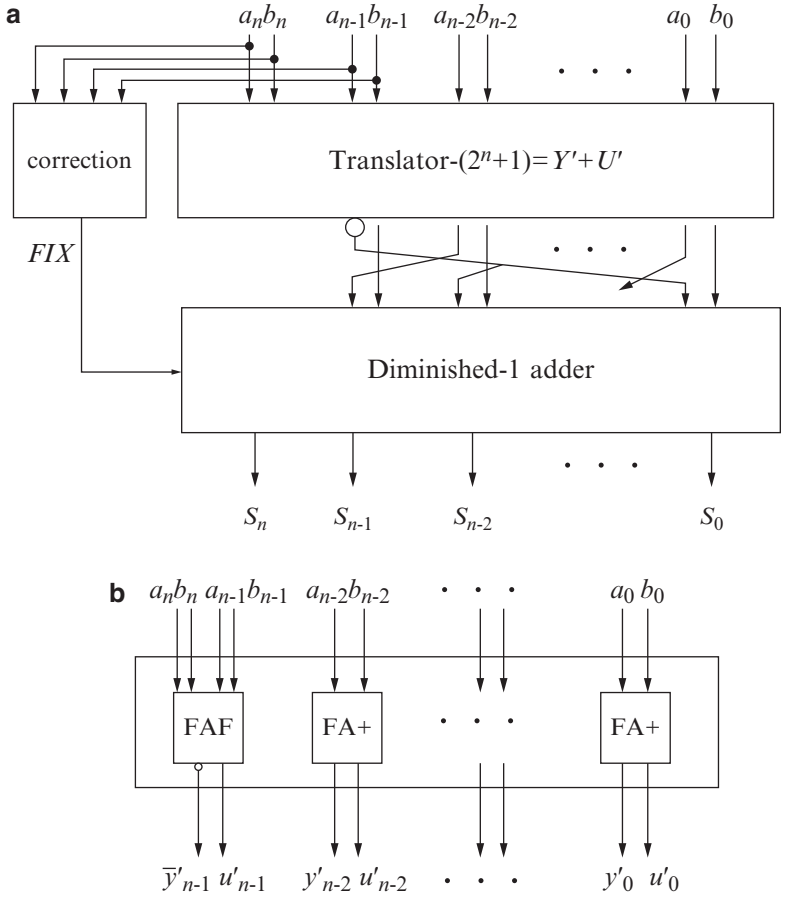

Figure 2.10 (**a**) Architecture of weighted modulo (2^n+1) adder with the correction scheme and (**b**) translator A + B–(2^n+1) (adapted from [24] ©IEEE2010)

where $y'_i = a_i \vee b_i$ and $u'_i = \overline{a_i \oplus b_i}$. Note also that *FIX* is wired OR with the carry c_{out} to yield the inverted EAC as the carry in. The *FIX* bit is needed since value greater than 3 cannot be accommodated in y_{n-1} and u_{n-1}.

The authors have used Sklansky [25] and Brent−Kung [3] parallel-prefix adders for the diminished-1 adder.

References

1. M.A. Bayoumi, G.A. Jullien, W.C. Miller, A VLSI implementation of residue adders. IEEE Trans. Circuits Syst. **34**, 284–288 (1987)
2. M. Dugdale, VLSI implementation of residue adders based on binary adders. IEEE Trans. Circuits Syst. **39**, 325–329 (1992)
3. R.P. Brent, H.T. Kung, A regular layout for parallel adders. IEEE Trans. Comput. **31**, 260–264 (1982)
4. G. Alia, E. Martinelli, Designing multi-operand modular adders. Electron. Lett. **32**, 22–23 (1996)
5. K.M. Elleithy, M.A. Bayoumi, A θ(1) algorithm for modulo addition. IEEE Trans. Circuits Syst. **37**, 628–631 (1990)
6. A.A. Hiasat, High-speed and reduced area modular adder structures for RNS. IEEE Trans. Comput. **51**, 84–89 (2002)
7. C. Efstathiou, D. Nikolos, J. Kalanmatianos, Area-time efficient modulo 2^n-1 adder design. IEEE Trans. Circuits Syst. **41**, 463–467 (1994)
8. R.E. Ladner, M.J. Fischer, Parallel-prefix computation. *JACM* **27**, 831–838 (1980)
9. P.M. Kogge, H.S. Stone, A parallel algorithm for efficient solution of a general class of recurrence equations. IEEE Trans. Comput. **22**, 783–791 (1973)
10. S. Knowles, A family of adders, in *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, Vail, 11 June 2001–13 June 2001. pp. 277–281
11. R. Zimmermann, Efficient VLSI implementation of Modulo $(2^n \pm 1)$ addition and multiplication, *Proceedings of the IEEE Symposium on Computer Arithmetic*, Adelaide, 14 April 1999–16 April 1999. pp. 158–167
12. L. Kalampoukas, D. Nikolos, C. Efstathiou, H.T. Vergos, J. Kalamatianos, High speed parallel prefix modulo (2^n-1) adders. IEEE Trans. Comput. **49**, 673–680 (2000)
13. A. Tyagi, A reduced area scheme for carry-select adders. IEEE Trans. Comput. **42**, 1163–1170 (1993)
14. C. Efstathiou, H.T. Vergos, D. Nikolos, Modulo $2^n \pm 1$ adder design using select-prefix blocks. IEEE Trans. Comput. **52**, 1399–1406 (2003)
15. R.A. Patel, S. Boussakta, Fast parallel-prefix architectures for modulo 2^n-1 addition with a single representation of zero. IEEE Trans. Comput. **56**, 1484–1492 (2007)
16. L.M. Liebowitz, A simplified binary arithmetic for the fermat number transform. IEEE Trans. ASSP **24**, 356–359 (1976)
17. Z. Wang, G.A. Jullien, W.C. Miller, An efficient tree architecture for modulo (2^n+1) multiplication. *J. VLSI Sig. Proc. Syst.* **14**(3), 241–248 (1996)
18. H.T. Vergos, C. Efstathiou, D. Nikolos, Diminished-1 modulo 2^n+1 adder design. IEEE Trans. Comput. **51**, 1389–1399 (2002)
19. S. Efstathiou, H.T. Vergos, D. Nikolos, Fast parallel prefix modulo (2^n+1) adders. IEEE Trans. Comput. **53**, 1211–1216 (2004)
20. H.T. Vergos, C. Efstathiou, A unifying approach for weighted and diminished-1 modulo (2^n+1) addition. IEEE Trans. Circuits Syst. II Exp. Briefs **55**, 1041–1045 (2008)

21. H.T. Vergos, D. Bakalis, On the use of diminished-1 adders for weighted modulo $(2^n + 1)$ arithmetic components, *Proceedings of the 11th Euro Micro Conference on Digital System Design Architectures, Methods Tools*, Parma, 3–5 Sept. 2008. pp. 752–759
22. S.H. Lin, M.H. Sheu, VLSI design of diminished-one modulo $(2^n + 1)$ adders using circular carry selection. IEEE Trans. Circuits Syst. **55**, 897–901 (2008)
23. T.B. Juang, M.Y. Tsai, C.C. Chin, Corrections to VLSI design of diminished-one modulo $(2^n + 1)$ adders using circular carry selection. IEEE Trans. Circuits Syst. **56**, 260–261 (2009)
24. T.-B. Juang, C.-C. Chiu, M.-Y. Tsai, Improved area-efficient weighted modulo $2^n + 1$ adder design with simple correction schemes. IEEE Trans. Circuits Syst. II Exp. Briefs **57**, 198–202 (2010)
25. J. Sklansky, Conditional sum addition logic. IEEE Trans. Comput. **EC-9**, 226–231 (1960)

http://www.springer.com/978-3-319-41383-9

Residue Number Systems
Theory and Applications
Ananda Mohan, P.V.
2016, X, 351 p. 131 illus., 17 illus. in color., Hardcover
ISBN: 978-3-319-41383-9
A product of Birkhäuser Basel