VHDL Carry Chain Adder (6C)

-
-

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

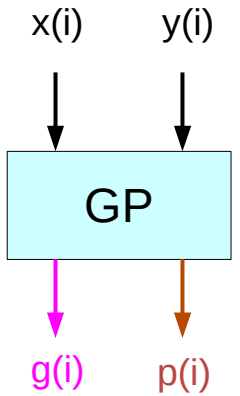
GP Cell

x(i), y(i) : ($\log_2$B)-bit number

Generate	$G_i = a_i \cdot b_i$
Propagate	$P_i = a_i \oplus b_i$

Generate	g(i) = 1	If x(i) + y(i) > (B – 1)
	0	otherwise

Propagate	p(i) = 1	If x(i) + y(i) = (B – 1)
	0	otherwise



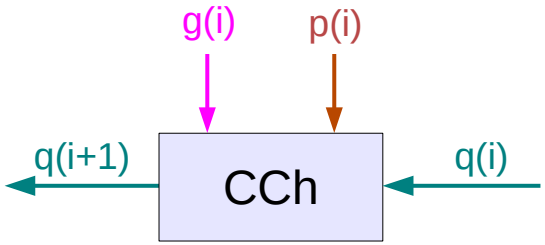
Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Carry Chain Cell (1)

q(i+1), q(i) : 1-bit number

$$c_{out} = G_i + P_i c_i$$

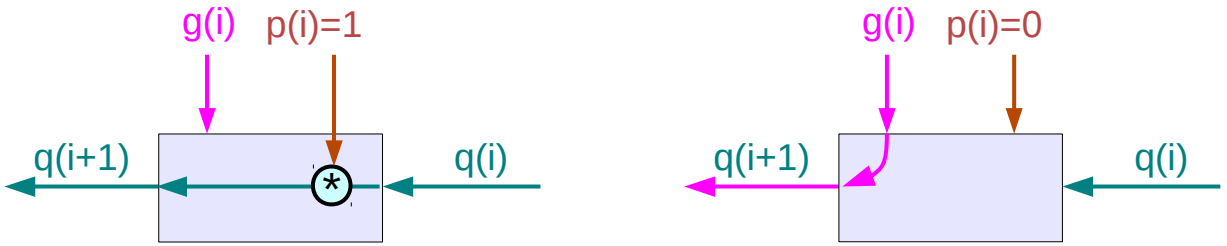
q(i+1)	= q(i)	when p(i) = 1	Propagate
	= g(i)	otherwise	Generate



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Carry Chain Cell (2)

q(i+1)	= q(i)	when p(i) = 1	Propagate
	= g(i)	otherwise	Generate

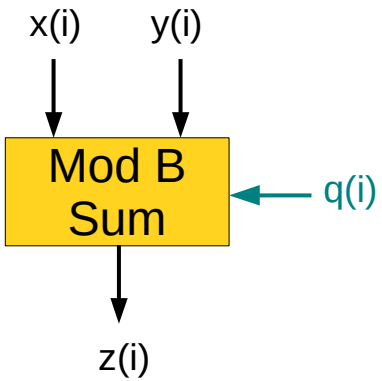


Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

$$c_{out} = G_i + P_i c_i$$

Mod B Sum Cell

$$z(i) = (x(i) + y(i) + q(i)) \bmod B$$



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

4-ary Carry Chain Addition Example

Generate	g(i) = 1	If x(i) + y(i) > 3
	0	otherwise

Propagate	p(i) = 1	If x(i) + y(i) = 3
	0	otherwise

q(i+1)	= q(i)	when	p(i) = 1	Propagate
	= g(i)	otherwise		Generate

z(i) = (x(i) + y(i) +q(i)) mod 4

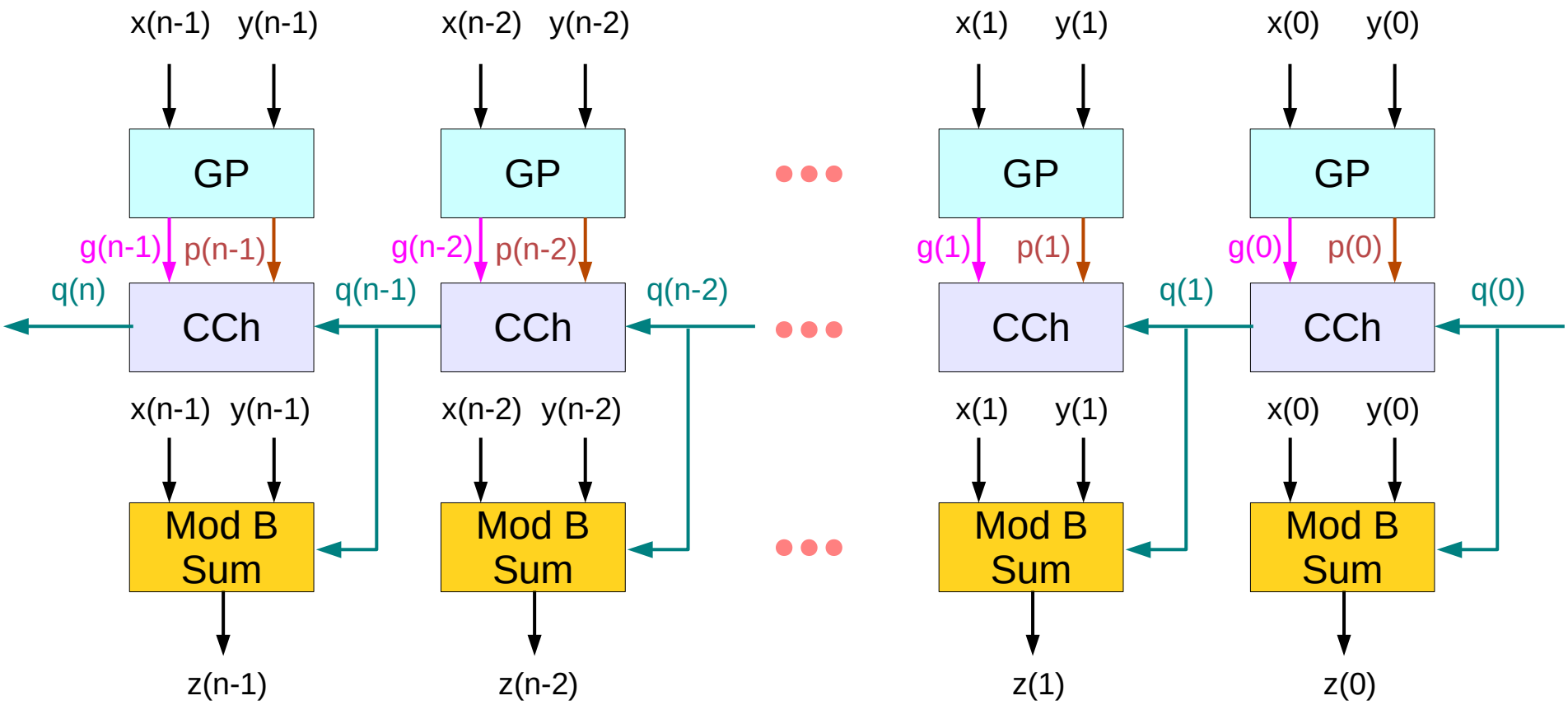
	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

p(i)	0	1	2	3
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

g(i)	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Carry Chain Adder



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Carry Chain Addition

```
-- computation of the generation and propagation conditions
for i in 0..n-1 loop
    g(i) := g(x(i), y(i));
    p(i) := p(x(i), y(i));
end loop

-- carry computation
q(0) := c_in;
for i in 0..n-1 loop
  if p(i)=1 then q(i+1):=q(i); else q(i+1):=g(i); end if;
end loop

-- sum computation
for i in 0..n-1 loop
    z(i) := (x(i)+y(i)+q(i)) mod B
end loop;
z(n) := q(n);
```

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Computation Decomposition

the first iteraton includes **2.n B-ary** operations
computation of g(i) and p(i) that could be executed in parallel

```
g(i) := g(x(i), y(i));
p(i) := p(x(i), y(i));
```

The second iteration is made up of **n** iteration steps
that must be used executed sequentially
as q(i+1) is a function of q(i)
consists of **binary** operation only

```
if p(i)=1 then q(i+1):=q(i); else q(i+1):=g(i); end if;
```

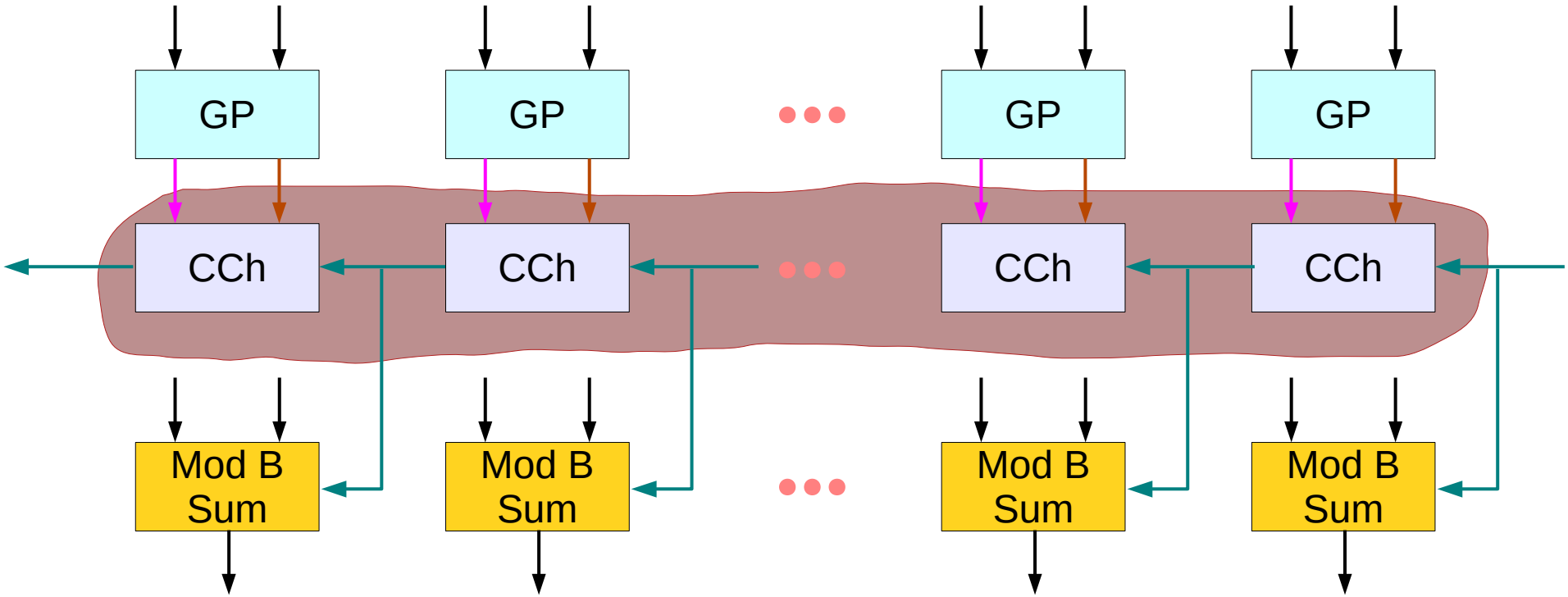
the last iteration includes **n B-ary** operations
computation of z(i) that could be executed in parallel

```
z(i) := (x(i)+y(i)+q(i)) mod B
```

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Sequential and concurrent computations

Splits the operations into **concurrent B-ary** ones (1st and 3rd iterations)
And **sequential binary** ones (2nd iterations)



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

B-ary n-digit Carry Chain Adder VHDL Code

```
q(0) <= c_in;

iterative_step for i in 0 to n-1 generate
    p(i) <= '1' when x(i)+y(i) = B-1 else '0';
    g(i) <= '1' when x(i)+y(i) > B-1 else '0';
    with p(i) select q(i+1) <= q(i) when '1', g(i) when others;
    z(i) <= (x(i)+y(i)+ conv_integer(q(i))) mod B;
end generate;

c_out <= q(n);
```

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

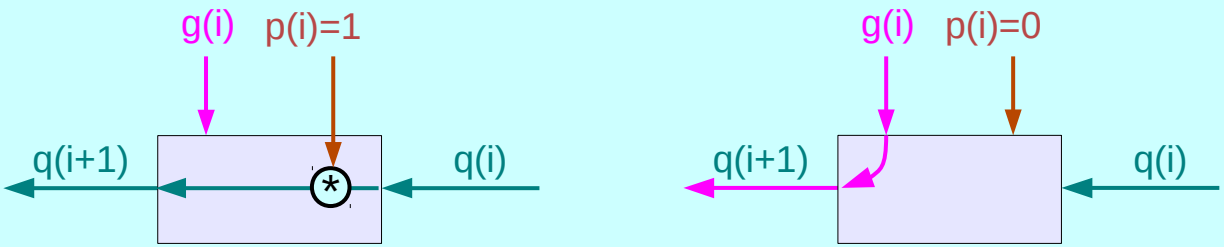
Computation Substitution

The **sequential binary** operations are the same whatever base **B** is

However, the expected computation time can be reduced
by the substitution of the relatively complex instruction

```
if x(i)+y(i)+q(i)>B-1 then q(i+1):=1 else q(i+1):=0 end if;
```

```
if p(i)=1 then q(i+1):= q(i) else q(i+1):=g(i); end if;
```



Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Relaxing g(i)

```
if p(i)=1 then q(i+1):= q(i) else q(i+1):=g(i); end if;
```

the corresponding Boolean equation

q(i+1) = p(i).q(i) V not(p(i)).g(i)

the generate function g(a,b) can be relaxed as bellows

g(a,b) = 1	if a + b > B-1 ….. when p(i) = 0 …. not(p(i))=1
g(a,b) = 0	if a + b < B-1 ….. when p(i) = 0 …. not(p(i))=1
g(a,b) = 1/0 dont care	if a + b = B-1 ….. when p(i) = 1 …. not(p(i))=0

the original generate and propagate function

g(a,b) = 1	if a + b > B-1,
g(a,b) = 0	otherwise
p(a,b) = 1	if a + b = B-1,
p(a,b) = 0	otherwise

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Relaxed g(i) Examples

	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

p(i)	0	1	2	3
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

g(i)	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1

Original g(i) **: 4-ary**

	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

p(i)	0	1	2	3
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

g(i)	0	1	2	3
0	0	0	0	X
1	0	0	X	1
2	0	X	1	1
3	X	1	1	1

Relaxed g(i) **: 4-ary**

	0	1
0	0	1
1	1	2

p(i)	0	1
0	0	1
1	1	0

g(i)	0	1
0	0	X
1	X	1

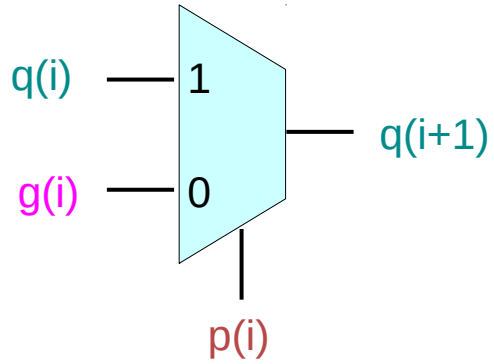
Relaxed g(i) **: binary**

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

Multiplexer Carry Chain

```
if p(i)=1 then q(i+1):= q(i) else q(i+1):=g(i); end if;
```

q(i+1) = p(i).q(i) V not(p(i)).g(i)



	0	1
0	0	1
1	1	2

p(i)	0	1
0	0	1
1	1	0

g(i)	0	1
0	0	X
1	X	1

Synthesis of Arithmetic Circuits: FPGA, ASIC and Ebedded Systems, J-P Deschamps et al

References

[1] http://en.wikipedia.org/
[2] J-P Deschamps,et. al., “Sunthesis of Arithmetic Circuits”, 2006