Commonly Used Boolean Algebra Properties

Property	Name	Description
A(B+C) = AB+AC A+(BC) = (A+B)(A+C)	Distributive (AND) Distributive (OR)	(AND) Same as multiplication in regular algebra (OR) Not at all like regular algebra
AB=BA A+B=B+A	Commutative	Variable order does not matter. Good practice is to sort variables alphabetically
(AB)C=A(BC) (A+B)+C = A+(B+C)	Associative	Same as regular algebra
AA' = 0 A+A' = 1	Complement (AND) Complement (OR)	(AND) Clearly one of A, A' must be 0. 1·0 = 0·1=0 (OR) Clearly one of A,A' must be 1. 1+0 = 0+1 =**1**
A·1=A A+0=A	Identity (AND) Identity (OR)	(AND) Result of A·1 is always A's value **0**·1=**0** **1**·1=**1** (OR) Result of A+0 is always A's value **0**+0=**0** **1**+0=**1**
A·0=0 A+1=1	Null elements	Result doesn't depend on the value of A
A·A=A A+A=A	Idempotent	Duplicate values can be removed
(A')' = A	Involution	(0')'=(1)'=0 (1')'=(0)'=1
A(A+B) = A A+(AB) = A	Covering	In either case, the value of B is irrelevent
(AB)+(AB') = A (A+B)(A+B')= A	Combining	This is an application of the complement and identity properties listed above
AB+B'C= AC (A+B)(B'+C)=(A+C)	Consensus	The conjunction of all unique terms
(AB)'=A'+B'	DeMorgan's Law (for AND)	Each literal complemented ANDs become ORs
(A+B)' = A'B'	DeMorgan's Law (for OR)	Each literal complemented, ORs become ANDs
A+A'B=A+B	Redundancy	A+A'B=(A+A')(A+B)

VHDL Cheat-Sheet

Concurrent Statements		**Sequential Statements**
Concurrent Signal Assignment	⇔	Signal Assignment
target <= *expression*;		*target* <= *expression*;
A <= B **AND** C; DAT <= (D **AND** E) **OR** (F **AND** G);		A <= B **AND** C; DAT <= (D **AND** E) OR (F **AND** G);
Conditional Signal Assignment	⇔	*if* statements

target <= *expressn* **when** *condition* **else**
expressn **when** *condition* **else**
expressn;
		if (*condition*) **then**
		{ sequence of statements }
		elsif (*condition*) **then**
		{ sequence of statements }
		else --(the else is optional)
		{ sequence of statements }
		end if;

F3 <= '1' **when** (L='0' **AND** M='0') **else** '1' **when** (L='1' **AND** M='1') **else** '0';		**if** (SEL = "111") **then** F_CTRL <= D(7); **elsif** (SEL = "110") **then** F_CTRL <= D(6); **elsif** (SEL = "101") **then** F_CTRL <= D(1); **elsif** (SEL = "000") **then** F_CTRL <= D(0); **else** F_CTRL <= '0'; **end if;**
Selective Signal Assignment	⇔	*case* statements
with *chooser_expression* **select** target <= *expression* **when** *choices*, *expression* **when** *choices;*		**case** *expression* **is** **when** *choices* => {sequential statements} **when** *choices* => {sequential statements} **when others** => {sequential statements} **end case**;
with SEL **select** MX_OUT <= D3 **when** "11", D2 **when** "10", D1 **when** "01", D0 **when** "00", '0' **when others**;		**case** ABC **is** **when** "100" **=>** F_OUT **<=** '1'; **when** "011" **=>** F_OUT **<=** '1'; **when** "111" **=>** F_OUT **<=** '1'; **when others =>** F_OUT **<=** '0'; **end case;**
Process		
label: **process**(*sensitivity_list*) **begin** {*sequential_statements*} **end process** *label*;		

proc1: **process**(A,B,C)
begin
if (A = '1' **and** B = '0') **then**
F_OUT <= '1';
elsif (B = '1' **and** C = '1') **then**
F_OUT <= '1';
else
F_OUT <= '0';
end if;
end process proc1;		