VHDL

de.wikibooks.org

26. August 2014

Inhaltsverzeichnis

0.1 Einführung

VHDL (VHSIC (Very High Speed Integrated Circuits) Hardware Description Language) ist in Europa die verbreitetste Hardware-Beschreibungssprache. Daneben gibt es VERILOG. Ursprünglich wurde sie entwickelt, um Testumgebungen für die Simulation integrierter Schaltungen zu entwickeln. Daher ist VHDL eine relativ komplexe Programmiersprache, deren Konstrukte nicht zwangsläufig synthetisierbar sind. Das führt bereits zum üblichen Ablauf der Hardwareentwicklung: Nach der Festlegung der Funktionalität wird diese mittels VHDL beschrieben. Der entstandene "Source-Code" wird compiliert und anschließend simuliert. Nachdem die einwandfreie Funktion sichergestellt ist wird der VHDL-Code direkt mit Hilfe eines Syntheseprogramms in eine Gatternetzliste umgesetzt. Dazu benötigt das Synthesewerkzeug eine von der Zielhardware abhängige Elementbibliothek, die der Chiphersteller zur Verfügung stellt.

0.1.1 Zum Wesen von VHDL

Im Innersten besteht jede digitale Hardwareschaltung aus kombinatorischer Logik und speichernden Elementen.

Unter **Kombinatorik** versteht man *NICHT/UND/ODER/Exclusiv-ODER*-Gatter und deren Kombinationen. Also jede Art von Verknüpfungen von einem oder mehreren Eingängen zu einem oder mehreren Ausgängen. Eine Änderung eines Eingangs bewirkt eine unmittelbare Wirkung auf den Ausgang. Eine wirkliche Hardwareschaltung benötigt dazu jedoch geringe Laufzeiten. Diese Laufzeiten kombinatorischer Logik werden aber in einer reinen *RTL*- Beschreibung nicht im VHDL-Code modelliert, obwohl dies an sich möglich wäre.

Speichernde Elemente sind *Flipflops* oder *Latches.* Latches sollte man als Zustandsspeicher in synchronen Schaltungen vermeiden. Flipflops sind Grundelemente, die einen Dateneingang und einen Takteingang haben. Der Ausgang übernimmt üblicherweise den Zustand des Eingangs mit der steigenden Taktflanke. Es gibt auch Flipflops, die mit der fallenden Flanke schalten. Zusätzlich können Flipflops auch einen asynchronen Reset-Eingang haben. Dieser setzt im Normalfall zu Beginn, nach Anlegen der Versorgungsspannung, das Flipflop in den Grundzustand (Ausgang hat den Pegel „0“).

Um diese zwei Arten der realen Hardware nachzubilden, ist die grundsätzliche Denkweise für ein VHDL-Programm deutlich anders, als es für den seriellen Ablauf beispielsweise eines C-Programms nötig ist. VHDL ist im Grunde eine Aneinanderreihung von **Prozessen**, die quasi simultan abgearbeitet werden. Der VHDL-Simulator wird zwar den Code in irgendeiner Weise in eine serielle Software umwandeln. Die Wirkungsweise der Prozesse ist jedoch so, als würden sie wirklich völlig gleichzeitig bearbeitet.

Zu welchem Zeitpunkt ein Prozess abgearbeitet wird, bestimmt die **Sensitivity-List**. Dies ist im Prozess-Header eine Liste von vereinbarten Signalen. Ändert eines dieser Signale den Wert, so wird der Prozess angestoßen. Führt dieses zu einer Änderung eines Ausgangs und ist dieser wiederum in der Sensitivity-List eines anderen Prozesses, so wird auch dieser mit dem neuen Wert des Signals angestoßen.

Man unterscheidet zwei Typen von Prozessen: Kombinatorische und synchron getaktete Prozesse, analog zu der eingangs besprochenen realen Hardware. Kombinatorische Prozesse haben in der Sensitivity-List alle Eingangssignale und beschreiben im Inneren deren Verknüpfung. Synchron getaktete Prozesse haben in der Sensitivity-List nur „reset“ und „clock“. Im Inneren wird beschrieben, welches Signal oder auch welche Verknüpfung von Signalen zur Taktflanke am Ausgang übernommen werden soll.

Beispiel für einen kombinatorischen Prozess:

```
procname: process(a,b,c)
begin
  x <= (a and b) or c;
end process;
```

-- Ausgang "x" ist eine Verknüpfung von "a","b","c"

Beispiel für einen synchron getakteten Prozess (und asynchronem, low-aktivem Reset):

```
procname: process(nres,clk)
begin
  if (nres='0') then
     q <= '0';
  elsif (clk'event and clk='1') then
     q <= x;
  end if;
end process;
```

-- FlipFlop mit Ausgang "q" schaltet mit steigender "clk"-Flanke und übernimmt den Wert von "x" -- x stammt aus kombinatorischem Prozess!!

Zur Beschreibung kombinatorischer Vorgänge gibt es die aus vielen Sprachen bekannte Konstrukte:

```
n: process(a,b)
begin
  x <= a and b;    -- Und-Verknüpfung von "a" und "b"
end;
```

--dazu gleichwertig:

```
n: process(a,b)
begin
  if (a='1') and (b='1') then
     x <= '1';
  else
     x <= '0';
  end if;
end;
```

--dazu gleichwertig:

```
n: process(a,b)
variable ab : std_logic_vector(1 downto 0);
begin
  ab(0) := a;
  ab(1) := b;
  case ab is
    when "00" => x <= '0';
    when "01" => x <= '0';
    when "10" => x <= '0';
    when "11" => x <= '1';
    when others => null;
  end case;
end;
```

Im Weiteren baut VHDL einen begrenzten Rahmen von logischen Elementen als ein Bauelement zusammen, das wiederum mit der diskreten Schaltungstechnik vergleichbar ist. So ist es ähnlich wie in der altbekannten TTL-Schaltungstechnik, dass ein solches Bauteil Eingänge, Ausgänge und ein Innenleben hat. VHDL definiert dieses Element oder diesen Block mit seinen "Pins" in der "**Entity**". Die "**Architecture**" beschreibt dann mit den oben gezeigten Prozessen das Innenleben.

Formal:

```
entity 74LS00 is
  port (i0,i1 : in bit;
        i2,i3 : in bit;      -- Eingänge
        o0    : out bit;     -- Ausgang
        o1    : out bit);
end 74LS00;
```

```
architecture behv of 74LS00 is

begin

signal z : bit;      -- Vereinbarung internen Signale

comb_1: process(i0,i1,i2,i3)
  begin
    o0 <= not(i0 and i1);    -- nand nr.1
    z  <= not(i2 and i3);    -- nand nr.2
  end comb_1;

comb_2: process(z)
  begin
    o1 <= z;
  end comb_2;

end behv;
```

Man sieht: In der "entity" wird eine Portliste vereinbart, die alle nach außen führenden Signale beinhalten muss. Interne Signale werden wie oben gezeigt vereinbart. Üblicherweise führen diese nach der Synthese zu realen "Drähten" in der Schaltung, können aber auch, wie in diesem Fall "z", wegoptimiert werden. Besonderheit: Signale, die den Block verlassen, können *nicht* in der "architecture" verschaltet werden. Das heißt, alle in der entity als "out" deklarierten Signale können nirgends in der "sensitivity list" eines Prozesses oder als Zuweisungswert erscheinen. Sie sollen ebenfalls nur in einem einzigen Prozess des VHDL-Files zugewiesen werden, so wie ein "Draht" ebenfalls zu einem Zeitpunkt immer nur einen Pegel führen kann.

Ein File mit einer Entity und einer Architecture ist bereits eine kompilierbare, vollständige VHDL-Komponente. Ähnlich wie auf einer Platine können auch mehrere VHDL-Bauteile miteinander quasi verdrahtet werden. In diesem Fall wird in einem anderen oder auch gleichen VHDL-File diese Komponente als "**component**" eingebunden.

Beispiel einer "Component"-Deklaration:

```
component 74LS00
  port (i0,i1,i2,i3 : in bit;
        o0,o1       : out bit);
end component;
```

Beispiel für eine "Component"-Instanziierung:

```
architecture behv of test is

  -- Signal-Deklarierungen:

  signal a,b,c,d : bit;

  -- Componenten-Deklarierungen:

  component adder
    port(a,b       : in bit;
         sum,carry : out bit);
  end component;


begin
```

```
  teil_a: adder port map (a,b,c,d);  -- Komponente "adder" hat den Namen
 "teil_a"
                                     -- a,b,c,d ist angeschlossen

end behv;
```

0.2 Basis-Konstrukte

Nach dieser kurzen Einführung über die grundsätzlichen Gedanken von VHDL sollen im Folgenden in alphabetischer Reihenfolge die sprachlichen Basis-Konstrukte beschrieben werden:

0.2.1 aggregates

Ein Aggregat ist ein Klammerausdruck, der mehrere Einzelelemente zu einem Vektor zusammenfasst, wobei die Elemente durch Kommata getrennt werden.

(wert_1,wert_2,...)

(element_1 => wert_1,element_2 => wert_2,...)

Beispiele:

```
signal databus : bit_vector(3 downto 0);
signal d1,d2,d3,d4 : bit;
...
databus <= (d1,d2,d3,d4);
```

identisch mit:

```
databus(3) <= d1;
databus(2) <= d2;
databus(1) <= d3;
databus(0) <= d4;



--
type zustand is (idle,run,warte,aktion);  -- enumerated type
signal state : zustand;
--

type packet is record
  flag   : std_logic;
  nummer : integer range 0 to 7;
  daten  : std_logic_vector(3 downto 0);
end record;
signal x : packet;
...
x <= ('1',3,"0011");
--

type vierbit is array(3 downto 0) of std_ulogic;
type speicher is array(0 to 7) of vierbit;

variable xmem : speicher := (others=>'0');   -- mit '0' vorbelegen
--
```

```
variable dbus : std_logic_vector(15 downto 0) := (others=>'Z');
```

0.2.2 alias

Ein Alias bezeichet einen Teil eines Signals.

```
alias ''alias_name'' : ''alias_type''(range) is ''signal_name''(range);
```

Beispiel:

```
signal daten_in : bit_vector(11 downto 0);

alias opcode : bit_vector(3 downto 0) is daten_in(3 downto 0);

alias daten  : bit_vector(7 downto 0) is daten_in(11 downto 4);
```

0.2.3 architecture

Eine Designeinheit in VHDL, die das Verhalten oder die Struktur einer Entity beschreibt.

```
architecture ''architecture_name'' of ''entity_name'' is
  declarations
begin
  concurrent statements
end architecture;
```

0.2.4 arrays

```
type ''type_name'' is array (range) of ''element_type'';
```

Beispiele:

```
type speicher is array (0 to 1023) of integer range 0 to 255;
signal sram : speicher;

type mem8 is array (natural range <>) of std_logic_vector(7 downto 0);
signal sram8_1024 : mem8(1023 downto 0);
signal sram8_8  : mem8(7 downto 0);
```

Zugriff/Initialisierung:

```
-- beide gleichwertig, Elemente 1 und 2 werden mit "5" beglückt
sram8_8 <= ("0", "0", "0", "0", "0", "5", "5", "0" );
```

```
sram8_8 <= ( 2 | 1 => "5", others => "0");

-- oder einzelnes Element mit 0xB füllen
sram8_8(2) <= x"B"
```

0.2.5 assert

```
assert ''bedingung'' report ''string'' severity ''severity_level'';
```

Überwache, dass *bedingung* erfüllt ist, wenn NICHT report *string* mit severity *severity_ level*

```
severity_level ist "error" (default), "note", "warning" oder "failure"
```

Beispiel:

```
assert (a > c) report "a muss grösser c sein" severity note;

assert (true)  report "diese Meldung wird nie ausgegeben" severity note;
assert (false) report "diese Meldung wird immer ausgegeben" severity note;
```

0.2.6 attributes

T repräsentiert ein beliebiger Typ, A repräsentiert ein Array, S repräsentiert ein beliebiges Signal und E repräsentiert eine Entity.

Liste von vordefinierten Attributen	
Attribut	Beschreibung
T'BASE	Basistyp von T
T'LEFT	Wert am weitesten links in T. (Grösster wenn downto)
T'RIGHT	Wert am weitesten rechts in T. (Kleinster wenn downto)
T'HIGH	Grösster Wert in T.
T'LOW	Kleinster Wert in T.
T'ASCENDING	Boolean, TRUE wenn range definiert mit "to".
T'IMAGE(X)	String der den Wert X repräsentiert.
T'VALUE(X)	Wert vom Typ T, konvertiert vom String X.
T'POS(X)	Integer Position von X im diskreten Typ T.
T'VAL(X)	Wert vom diskreten Typ an integer Position X.
T'SUCC(X)	Wert vom diskreten Typ der auf X folgt
T'PRED(X)	Wert vom diskreten Typ der vor X liegt
T'LEFTOF(X)	Wert vom diskreten Typ links von X
T'RIGHTOF(X	) Wert vom diskreten Typ rechts von X
A'LEFT	Eintrag ganz links in A
A'LEFT(N)	Eintrag ganz links in Dimension N von A
A'RIGHT	Eintrag ganz rechts in A
A'RIGHT(N)	Eintrag ganz links in Dimension N von A

Liste von vordefinierten Attributen	
Attribut	Beschreibung
A'HIGH	Höchster Eintrag in A
A'HIGH(N)	Höchster Eintrag in Dimension N von A
A'LOW	Tiefster Eintrag in A
A'LOW(N)	Tiefster Eintrag in Dimension N von A
A'RANGE	Range von A'LEFT to A'RIGHT oder A'LEFT downto A'RIGHT
A'RANGE(N)	Range von Dimension N in A
A'REVERSE_RANGE	Range in A to und downto umgekehrt
A'REVERSE_RANGE(N)	REVERSE_RANGE von Dimension N in A
A'LENGTH	Integer Wert der Anzahl Elemte in A
A'LENGTH(N)	Anzahl Werte in Dimension N von A
A'ASCENDING	Boolean, TRUE wenn range definiert mit "to"
A'ASCENDING(N)	Boolean, TRUE wenn Dimension N in A definiert mit "to"
S'DELAYED(t)	Signalwert zur Zeit NOW -t
S'STABLE	TRUE, wenn kein Event in S
S'STABLE(t)	TRUE, wenn kein Event in S für Zeit t
S'QUIET	TRUE, wenn kein Event in diesem Simulations Zyklus
S'QUIET(t)	TRUE, wenn kein Event in S für Zeit t
S'TRANSACTION	Bit Signal, invertiert immer wenn Signal S ändert
S'EVENT	TRUE, wenn Signal S Event in diesem Simulationszyklus hatte
S'ACTIVE	TRUE, wenn Signal S aktiv in diesem Simulationszyklus
S'LAST_EVENT	Zeit seit letzem Event auf Signal S
S'LAST_ACTIVE	Zeit seit Singal S zuletzt aktiv
S'LAST_VALUE	Vorhergehender Wert von Signal S
S'DRIVING	
S'DRIVING_VALUE	
E'SIMPLE_NAME	String mit Name der Entitiy E
E'INSTANCE_NAME	String mit Designs Hirarchie inkl. Entity E
E'PATH_NAME	String zu Design Wurzel von E

Beispiele:

```
signal'LEFT           7           bei std_logic_vector(7 downto 0);
                      0           bei std_logic_vector(0 to 7);
signal'RIGHT          0           bei std_logic_vector(7 downto 0);
                      7           bei std_logic_vector(0 to 7);
signal'HIGH           7           bei std_logic_vector(7 downto 0);
                      7           bei std_logic_vector(0 to 7);
signal'LOW            0           bei std_logic_vector(7 downto 0);
                      0           bei std_logic_vector(0 to 7);
signal'RANGE          7 downto 0  bei std_logic_vector(7 downto 0);
                      0 to 7      bei std_logic_vector(0 to 7);
signal'REVERSE_RANGE  0 to 7      bei std_logic_vector(7 downto 0);
                      7 downto 0  bei std_logic_vector(0 to 7);
signal'LENGTH         8           bei std_logic_vector(7 downto 0);
```

```
                          8          bei std_logic_vector(0 to 7);
  signal'EVENT      if (clk'event and clk='1') then
```

0.2.7 block statements

```
''block_name'': block
  declarations
begin
  concurrent statements
end block;
```

0.2.8 case

```
case ''expression'' is
  when ''fall_1'' => ''sequential statement''
  when ''fall_2'' => ''sequential statement''
  when others     => ''sequential statement''
end case;
```

Beispiel1:

```
case wert is
  when 0      => w <= '1';
  when 1      => w <= '0';
  when 2 | 3  => w <= a;
  when 4 to 7 => w <= b;
  when others => w <= 'X';
end case;
```

Beispiel2:

```
wert <= '0';
case din is
  when "00"   => wert <= '1';
  when others => null;          -- "when others" soll immer vorhanden sein
end case;                       -- durch default-Zuweisung is "null"-statement
 möglich!
```

Beispiel3:

```
    type state_type is ( IDLE, DO_SOMETHING );
    ...
    case state is
      when IDLE         => tx_line <= '0';
      when DO_SOMETHING => tx_line <= '1';
      when others =>
        assert false report "case defaulted!" severity failure;
     end case;
```

0.2.9 component declaration

Deklaration zur Festlegung des Namens und der Schnittstelle einer Komponente, die einer Entitydeklaration und Architecture zugeordnet sein muss.

```
component ''component_name''
  generic (''generic_liste'');
  port    (''port_liste'');
end component;
```

0.2.10 component instantiation

```
label: ''component_name''
generic map ( ''generic1'' => '' generic1_entity'' )
port map (
          ''component_port1'' => ''entity_port1'',
          ''component_port2'' => ''entity_port2'',
          ...
          ''component_portx'' => ''entity_portx''
          );
```

0.2.11 constant

```
constant ''constant_name'' : type := value;
```

Beispiel:

```
constant festwert : std_logic_vector(7 downto 0) := "10101100";
constant zeitwert : time := 50 ns;

type rdatum is array (0 to 3) of bit_vector(7 downto 0);
constant rom : rdatum :=
  ("00000001",
   "00000010",
   "00000011",
   "00000100");
```

0.2.12 entity

Eine Struktureinheit in einem VHDL- Entwurfssystem. Beschreibt die Schnittstellen eines VHDL-Funktionsblocks nach außen. Mit Hilfe von Port-Anweisungen erfolgt die Deklaration der Anschlüsse innerhalb der Entity. Zu jeder Entity gehört eine Architecture.

```
entity ''entity_name'' is
  generic (generic_list);
  port    (port_list);
end ''entity_name'';
```

0.2.13 exit-Anweisung

mit der exit-Anweisung wird die "innerste" Schleife verlassen und mit der Anweisung, die direkt auf die Schleife folgt, fortgefahren.

Beispiel: Bestimmen der Anzahl der führenden Nullen

```
for i in signal'range loop
 exit when signal(i)='1';
 null_v := null_v + 1;
end loop;
```

0.2.14 file declaration

Beispiel:

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.std_logic_textio.all;
library std;
  use std.textio.all;
```

```
architecture sim of dut is
  file my_file : text open write_mode is "my_file.dat";
begin
  proc: process(clk)
    variable outline : line;
    variable counter : integer := 0;
  begin
    if rising_edge(clk) then
      write(outline, string'("Takt: "));
      write(outline, counter);
      writeline(my_file, outline);
      counter := counter + 1;
    end if;
  end process proc;
end sim;
```

0.2.15 for loop

```
''eventuell_label'': for ''parameter'' in ''range'' loop
  sequential statements
end loop ''eventuell_label'';
```

- Parameter muss nicht deklariert werden.
- Parameter darf im loop nicht verändert werden

Muss loop synthetisierbar sein:

- range ist statisch
- keine wait statments im loop

Beispiel:

```
for i in 0 to 7 loop
  w(i) <= a(i) and b;   -- 8bit bus "a" mit Einzelsignal "b" verunden
end loop;
```

Beispiel 2:

```
for i in 1 to 10 loop
    if (REPEAT = '1') then
        i := i-1;    -- Error
    end if;
end loop;
```

0.2.16 functions

Eine der beiden Möglichkeiten in VHDL, Code mittels eines einfachen Aufrufmechanismus wiederverwertbar zu machen. Functions werden gewöhnlich mit ihrem Namen und einer in Klammern stehenden Liste der Eingangsparameter aufgerufen und können nur ein Ausgangsargument liefern- vgl. auch Procedure.

```
function ''funktion_name'' (parameter_list) return ''type'' is
  declarations
begin
  sequential statements
end function_name;
```

Beispiel:

```
function parity_generator (din : std_ulogic_vector)
                           return std_ulogic is
  variable t : std_ulogic := '0';   -- variable mit default Zuweisung
begin
  for i in din'range loop           -- ganze Busbreite
     t := t xor din(i);
  end loop;
  return t;
end parity_generator;
```

Aufruf der Funktion als "concurrent" oder "sequential statement":

```
  sig_pary <= parity_generator(data_bus);
```

Achtung: keine "signal assignments" oder "wait"

0.2.17 generate

```
''label'': for ''parameters'' in ''range'' generate
  concurrent statements
end generate ''label'';
```

oder

```
''label'': if ''condition'' generate
  concurrent statements
end generate ''label'';
```

Beispiel:

```
architecture gen of test is

  component volladdierer
    port (x,y,ci : in bit;
          s,co   : out bit);
  end component;

  component halbaddierer
    port (x,y  : in bit;
          s,co : out bit;
  end component;

  signal carry : bit_vector(0 to 7);

begin

  gen_addierer: for i in 0 to 7 generate

     niedrigstes_bit: if i=0 generate
       w0: entity halbaddierer port map
           (x(i),y(i),s(i),carry(i));
     end generate niedrigstes_bit;

     hoeheres_bit: if i>0 generate
       wi: entity volladdierer port map
           (x(i),y(i),carry(i-1),s(i),carry(i));
     end generate hoeheres_bit;

  end generate gen_addierer;

  co <= carry(7);

end gen;
```

Beispiel2:

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;
...
    dft_mem_in      : in  std_logic_vector(const1 downto 0);
    dft_mem_out     : out std_logic_vector(const2 downto 0);
    pwr_mem_ctrl_in : in  std_logic_vector(3 downto 0)
...
  gen0: for i in dft_mem_out'range generate
    signal tmp : std_logic_vector(dft_mem_out'range);
  begin
    tmp <= std_logic_vector(resize(unsigned(dft_mem_in), dft_mem_out'length));
    dft_mem_out(i) <= tmp(i) xor (pwr_mem_ctrl_in(0) xor pwr_mem_ctrl_in(1) xor
 pwr_mem_ctrl_in(2) xor pwr_mem_ctrl_in(3));
  end generate gen0;
```

0.2.18 generic

```
entity ''entity_name'' is
  generic (generic_list);
  port    (port_list);
end ''entity_name'';
```

Beispiel: (wichtig für skalierbare Blöcke!!)

```
  entity test is
    generic (n : integer := 15); -- hierbei ist 15 der Default-Wert, falls kein
 Generic
                                 -- bei der Initialisierung angegeben wird
    port    (a : in std_ulogic_vector(n-1 downto 0));
  end test;
```

0.2.19 if

```
if     condition_a then
  sequential statements
elsif condition_b then
  sequential statements
else
  sequential statements
end if;
```

Beispiel:

```
if nreset='0' then
   count <= 0;
elsif clk'event and clk='1' then
  if count=9 then
    count <= 0;
  else
    count <= count+1;
  end if;
end if;
```

0.2.20 library

0.2.21 names

0.2.22 next-Anweisung

Die next-Anweisung beendet den aktuellen Schleifendurchlauf vorzeitig; das bedeutet, dass die Anweisungen bis zur end-loop-Anweisung übersprungen werden und mit dem nächsten Schleifendurchlauf fortgefahren wird.

Beispiel: Bestimmen der Anzahl der Nullen in einem Vektor

```
for i in signal'range loop
 next when signal(i)='1';
 null_v := null_v + 1;
end loop;
```

0.2.23 notations

```
hex_var := 16#8001#;

binary_s <= b"000_111_010";
octal_s  <= o"207";
hex_s    <= x"01_FB";
```

0.2.24 null statement

Falls durch die Syntax ein Statement erforderlich ist, kann das "Null"-Statement verwendet werden, um anzuzeigen, dass nichts zu tun ist. Vgl. hierzu beispielsweise 'when others => null;' einer case-Anweisung.

```
proc: process(clk)
begin
  if rising_edge(clk) then
    case select is
      when "00" => reg <= '1';
      when "11" => reg <= '0';
      when others => null;
    end case;
  end if;
end process proc;
```

0.2.25 operators

Logische Operatoren für Typen: bit,boolean,bit_vector,std_logic,std_logic_vector

```
and      -- und
or       -- oder
nand     -- nicht und
nor      -- nicht oder
xor      -- exclusive oder
xnor     -- exclusives nicht oder
```

Vergleichs Operatoren Ergebnis: boolean

```
=        -- Gleichheit
/=       -- Ungleichheit

<        -- kleiner
>        -- groesser
<=       -- kleiner gleich (Achtung bei Type int: Speichern)
```

```
>=      -- grösser gleich
```

Arithmetische Operatoren für Typen: integer,real

```
a  <= a + 7;       -- Addition
r1 <= r2 - 3.1415  -- Subtraktion (real)
m  <= x * y        -- Multiplikation
d  <= m / 2        -- Division
```

VHDL93:

```
sll   -- shift left  logical
srl   -- shift right logical
sla   -- shift left arith.
sra   -- shift right arith.
rol   -- rotate left
ror   -- rotate right
```

0.2.26 package

```
package ''package_name'' is
  declarations
end package;
```

Beispiele:

```
package demo is
  constant nullwert : bit_vector := "00000000";
  function foo ( v : std_ulogic ) return std_ulogic;
  component adder          -- Dessen Implementierung ist vielleicht in
 irgendeiner Library vorcompiliert,
    port(x,y,ci : in bit;  -- da ein Component niemals in der package body
 definiert werden kann.
         s,co   : out bit);
  end component;
end demo;

package body demo is
  function foo ( v : std_ulogic ) return std_ulogic is
  begin
    return v;
  end function;
end package body;
```

Package-Aufruf lautet dann:

```
use work.demo.all;

 entity xx is
 port
   ( wert : out bit_vector(7 downto 0));
 end xx;

 architecture behv of xx is
 begin
   wert <= nullwert;
```

```
end behv;
```

0.2.27 procedures

-- Parameter können constant, variable oder Signale sein. Auf Signale kann gelesen (in) oder geschrieben (out) werden.

```
procedure ''procedure_name'' (paramter_list) is
  declarations
begin
  sequential statements
end ''procedure_name'';
```

Beispiel:

```
procedure parity_generator
  (signal din : in  std_ulogic_vector;
   signal par : out std_ulogic) is
variable t : std_ulogic := '0';
begin
  for i in 0 to din'range loop
    t := t xor din(i);
  end loop;
  par <= t;
end parity_generator;
```

Proceduren in Packages:

```
package my_package is


  procedure parity_generator
    (signal din : in  std_ulogic_vector;
     signal par : out std_ulogic);


end my_package;


package body my_package is

  procedure parity_generator
   (signal din : in  std_ulogic_vector;
    signal par : out std_ulogic)  is

    variable t : std_ulogic := '0';
  begin
      for i in 0 to din'range loop
      t := t xor din(i);
      end loop;
      par <= t;
  end parity_generator;

end my_package;
```

Aufruf:

```
...
parity_generator(databus,par_bit);
...
```

0.2.28 process

```
''optionales_label'': process (optionale sensitivity liste)
  declarations
begin
  sequential statements;
end process optionales_label;
```

0.2.29 records

```
  type my_record_t is record
    element1 : std_logic;
    element2 : std_logic;
    element3 : std_logic_vector(1 downto 0);
    element4 : std_logic_vector(4 downto 0);
  end record;

  type my_array_t is array (3 downto 0) of my_record_t;



  signal my_signal_s : my_array_t;

  my_signal_s <= (others => ('0','0',(others => '0'),(others => '0')));
```

0.2.30 type definiert in VHDL

type Time is range --implementation defined-- ;

```
  units
     fs;            -- femtosecond
     ps  = 1000 fs; -- picosecond
     ns  = 1000 ps; -- nanosecond
     us  = 1000 ns; -- microsecond
     ms  = 1000 us; -- millisecond
     sec = 1000 ms; -- second
     min = 60  sec; -- minute
     hr  = 60  min; -- hour
  end units;
```

0.2.31 type struktur

```
  types-+-scalar----+-discrete-------+-integer-------+-integer
        |           |                |               +-natural
        |           |                |               +-positive
```

```
     |          |                |
     |          |                +-enumeration---+-boolean
     |          |                                +-bit
     |          |                                +-character
     |          |                                +-file_open_kind
     |          |                                +-file_open_status
     |          |                                +-severity_level
     |          |
     |          +-floating point-+-----------------real
     |          |
     |          +-physical-------+-----------------delay_length
     |                           +-----------------time
     |
     +-composite-+-array----------+-constrained-
     |          |                |
     |          |                +-unconstrained-+-bit_vector
     |          |                                +-string
     |          |
     |          +-record-
     |
     +-access-
    |
    +-file-
```

0.2.32 Typumwandlung (type conversion)

```
library ieee;
 use ieee.std_logic_1164.all;
 use ieee.numeric_std.all;
```

```
std_logic_vector  →  unsigned          | unsigned(arg)
std_logic_vector  →  signed            | signed(arg)
std_logic_vector  →  integer           |
std_logic_vector->signed/unsigned->integer
natural           →  unsigned          | to_unsigned(arg,size)
integer           →  signed            | to_signed(arg,size)
integer           →  std_logic_vector  | integer->signed->std_logic_vector
unsigned          →  std_logic_vector  | std_logic_vector(arg)
unsigned          →  integer           | to_integer(arg)
signed            →  std_logic_vector  | std_logic_vector(arg)
signed            →  integer           | to_integer(arg)
```

```
unsigned          →  unsigned          | resize(arg,size)
signed            →  signed            | resize(arg,size)
```

```
std_ulogic        →  bit               | to_bit(arg)
std_ulogic_vector →  bit_vector        | to_bitvector(arg)
std_ulogic_vector →  std_logic_vector  | to_stdlogicvector(arg)
bit               →  std_ulogic        | to_stdulogic(arg)
bit_vector        →  std_logic_vector  | to_stdlogicvector(arg)
bit_vector        →  std_ulogic_vector | to_stdulogicvector(arg)
std_logic_vector  →  std_ulogic_vector | to_stdulogicvector(arg)
std_logic_vector  →  bit_vector        | to_bitvector(arg)
```

Beispiele:

```
signal value_i   : integer range 0 to 15;
```

```
signal value_u   : unsigned(3 downto 0);
signal value_slv : std_logic_vector(3 downto 0);
…
value_i   <= 14;
value_u   <= to_unsigned(value_i, value_u'length); -- Typumwandlung
value_slv <= std_logic_vector(value_u);            -- cast (Typen sind eng
verwandt)
```

0.2.33 type declaration

```
type <memory_type> is array(0 to 9) of std_logic_vector(7 downto 0);
type <fsm_type> is (idle, run, ready);
type <hour_range_type> is range 1 to 12;
subtype <vector_type> is std_logic_vector(5 downto 0);
```

0.2.34 use

Verwendung von Bibliotheken, oder Teilen daraus.

Beispiel:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

0.2.35 variable declaration

```
variable ''Variablenname'' : ''Typ'' := ''Initialisierungswert''
```

Beispiel:

```
variable MeinBool : boolean := false;
```

0.2.36 variable assignment

```
''Variablenname'' := ''Wert'';
```

0.2.37 wait

- wait until *condition*
- wait on *signal list*
- wait for *time* -- nicht synthetisierbar

- wait;

Beispiel:

```
wait until din="0010";
```

oder:

```
stimulus: process
begin

  loop
    clk <= '0';
    wait for 50 ns;
    clk <= '1';
    wait for 50 ns;
  end loop;

end process;
```

0.2.38 when

signal_ name <= *expression* when *condition* else
expression when *condition* else
expression;

Beispiel (Tristate Port):

```
data <= data_out when data_enable = '1' else 'Z';
```

0.2.39 while

`while` *condition* loop
sequential statements
end loop;

Beispiel (Register rechts schieben):

```
if clk'event and clk='1' then
  i:=0;
  while i<7 loop
    buswert(i) <= buswert(i+1);
    i:=i+1;
  end loop;
  buswert(7) <= din;
end if;
```

0.2.40 with select

```
with select_signal select
 dst_signal <= src_signal when select_value1,
               src_signal when select_value2,
               src_signal when others;
```

Selektive Signalzuweisung außerhalb eines Prozesses, als Alternative zu case:

```
signal sel_s                 : std_logic_vector(1 downto 0);
signal monitor_s,one_s,two_s : std_logic_vector(3 downto 0);
```

```
with sel_s select
  monitor_s <= one_s when "00",
               two_s when "11",
               "0000" when others;
```

0.3 Operator precedence

VHDL operators in order of precedence (Highest first)

Note: The concatenate operator & has a lower order of precedence than some arithmetic operators +/-

```
**
abs
not
*
/
mod
rem
+
-
+
-
&
sll
srl
sla
sra
rol
ror
=
/=
<
<=
>
>=
and
or
nand
nor
xor
xnor
```

0.4 Beispielprogramme

0.4.1 Zustandsmaschinen

Die im Folgenden beschriebene Codierung einer Zustandsmaschine erhebt nicht den Anspruch die eleganteste Lösung darzustellen, sie soll vielmehr einen Eindruck der Sprache und der grundsätzlichen Abläufe vermitteln.

```
architecture demo of zustandsmaschine is

  constant vectorbreite : integer := 3;

  type zustandsvector is std_logic_vector(vectorbreite downto 0);

  signal zustand : zustandsvector;

  -- "one hot" Codierung !!
  constant warte_zustand   : zustandsvector := "0001";
  constant a_zustand       : zustandsvector := "0010";
  constant b_zustand       : zustandsvector := "0100";
  constant c_zustand       : zustandsvector := "1000";

  signal gehe_zu_warte_zustand : std_logic;
  signal gehe_zu_a_zustand     : std_logic;
  signal gehe_zu_b_zustand     : std_logic;
  signal gehe_zu_c_zustand     : std_logic;

  signal timer : std_logic_vector(3 downto 0);

begin
--------------------------------------------
zustaende : process(nres,clk)
begin
   if    (nres='0') then
       zustand <= warte_zustand;
   elsif (clk'event and clk='1') then
     if    (gehe_zu_warte_zustand='1') then
       zustand <= warte_zustand;
     elsif (gehe_zu_a_zustand='1') then
       zustand <= a_zustand;
     elsif (gehe_zu_b_zustand='1') then
       zustand <= b_zustand;
     elsif (gehe_zu_c_zustand='1') then
       zustand <= c_zustand;
     else                      -- der "else" Pfad ist optional!
       zustand <= zustand;    -- ohne diese Zeilen identische Funktion!!
     end if;
end process zustaende;
--------------------------------------------
zustandsweiterschaltung : process(eingang1,eingang2,zustand,timer)
begin
  gehe_zu_warte_zustand <= '0';   -- default assignments
  gehe_zu_a_zustand     <= '0';
  gehe_zu_b_zustand     <= '0';
  gehe_zu_c_zustand     <= '0';

  case zustand is
    when warte_zustand =>
                          if    (eingang1='1') then    -- Priorisierung von
                            gehe_zu_a_zustand <= '1';  -- "eingang1" und
 "eingang2"
                          elsif (eingang2='1') then
                            gehe_zu_c_zustand <= '1';
                          end if;
```

```
    when a_zustand     =>
                          if (timer="0000") then         -- Maschine bleibt im
 "a_zustand"
                            gehe_zu_b_zustand <= '1';    -- bis Timer abgelaufen
                          end if;
    when b_zustand     =>
                          gehe_zu_warte_zustand <= '1';
    when c_zustand     =>
                          gehe_zu_a_zustand <= '1';
    when others        =>                                 -- diese Zeile sichert
 Vollständigkeit
                          gehe_zu_warte_zustand <= '1';   -- der Auscodierung!!!
  end case;
end process zustandsweiterschaltung;
--------------------------------------------
wartezeit : process (nres,clk)
begin
  if    (nres='0') then
      timer <= "1010";
  elsif (clk'event and clk='1') then
      if (zustand=a_zustand) then     -- Zahler steht auf "1010"
         timer <= timer-1;            -- nur im "a_zustand" läuft er rückwärts
      else
         timer <= "1010";
      end if;

  end if;
end process wartezeit;
--------------------------------------------
--------------------------------------------
getakteter_ausgang : process (nres,clk)
begin
  if    (nres='0') then
      ausgang1 <= '0';
  elsif (clk'event and clk='1') then
      if (zustand=b_zustand) then
         ausgang1 <= '1';          -- eine "clk"-Periode langer Pulse
      else                         -- Achtung: erscheint einen Takt nach
 "b_zustand"!!
         ausgang1 <= '0';
      end if;
  end if;
end process getakteter_ausgang;
--------------------------------------------
--------------------------------------------
asynchroner_ausgang : process (gehe_zu_b_zustand)
begin

  ausgang2 <= gehe_zu_b_zustand;    -- eine "clk"-Periode langer Pulse (könnte
 spiken!!)

end process asynchroner_ausgang;
--------------------------------------------
--------------------------------------------
```

0.5 Fehlervermeidung

0.5.1 Unbeabsichtiges Erzeugen eines "Latch"

In der Synthese wird unbeabsichtigt ein "Latch" implementiert. Die Ursache ist meist, dass in einem kombinatorischen Prozess die Zuweisungen auf ein Signal nicht vollständig auscodiert wurden:

Beispiel:

```
if    (a='1') and (b='0') then
    x <= '1';
elsif (a='0') and (b='1') then
    x <= '0';
end if;
```

Die Fälle a=1 und b=1 ebenso wie a=0 und b=0 wurden nicht definiert. Die Folge ist, dass die Synthese versucht in diesen Fällen den aktuellen Zustand beizubehalten. Dies erfolgt durch die Implementierung eines "Latch".

0.5.2 Vergessen von Signalen in der "sensitivity list" eines kombinatorischen Prozesses

Werden Signale in der Liste nicht absichtlich weggelassen, um ein bestimmtes Verhalten der Schaltung zu erzeugen, sondern aus Nachlässigkeit nicht hinzugefügt, wird als Folge das Verhalten der Simulation von dem der realen Gatterschaltung abweichen. Es gibt vhdl-Editoren, die die "sensitivity list" prüfen. Auch in der Synthese erscheinen meistens Warnmeldung.

en:Programmable Logic/VHDL[1] fr:Conception et VHDL[2]

0.5.3 Verwenden von Reset-Signalen in einem Design

Asynchrone Reset-Signale gehören bei VHDL-Designs unbedingt synchronisiert:

```
library    ieee;
use    ieee.std_logic_1164.all;

entity myEnt is
  port(
    rst_an : in std_logic;
    clk: in std_logic;
    rst: in std_logic;
    sigIn: in std_logic_vector(3 downto 0);
    sigOut: out std_logic_vector(3 downto 0));
end entity myEnt;

architecture myArch of myEnt is
```

1 http://en.wikibooks.org/wiki/Programmable%20Logic%2FVHDL
2 http://fr.wikibooks.org/wiki/Conception%20et%20VHDL

```
    signal sync_rst_r : std_logic_vector(1 downto 0);
    signal mySig: std_logic_vector(sigIn'range);

  begin

    process(clk)
    begin
      if (rising_edge(clk)) then
        sync_rst_r <= sync_rst_r(0) & rst_an;
      end if;
    end process;

    process(clk, sync_rst_r)
    begin
      if sync_rst_r(1) = '0' then
          mySig <= (others => '0');
      elsif (rising_edge(clk)) then
        if (rst = '1') then
          mySig <= (others => '0');
        else
          mySig <= sigIn;
        end if;
      end if;
    end process;

  sigOut <= mySig;

  end architecture;
```

Für detailierte Information zum Thema Reset im FPGAs siehe das Whitepaper von Xilinx "Get Smart About Reset: Think Local, Not Global" (englischsprachig).

0.5.4 Generieren von "Clock"-Signalen in einem Design

Die "Clock"-Signale werden mit Hilfe einer speziellen Verdrahtung auf dem FPGA verteilt. Ein "Clock"-Signal darf nie durch Logik erzeugt werden. Im Folgenden wird ein schlechtes Beispiel gezeigt, in welchem ein neues Clock Signal erzeugt wird:

```
library ieee;
use ieee.std_logic_1164.all;

entity clock_divider_ent is
port (
    clk        : in    std_logic;
    clkDiv        : out    std_logic);

end clock_divider_ent;

architecture synth of clock_divider_ent is

    signal counter:    integer range 1023 downto 0;
    signal clkDivInt: std_logic := '0';

begin

  process(clk)
  begin
      if (rising_edge(clk)) then
          if (counter = 0) then
              counter <= 1023;
              clkDivInt <= not clkDivInt;
          else
```

```
                counter <= counter - 1;
            end if;
        end if;
    end process;

  end synth;
```

In diesem Beispiel wird eine bessere Implementierung für den oben gezeigten Block dargestellt. Das Signal clkDivInt wird hier zu einem kurzen Puls: jedes Mal wenn der Zähler den Wert null erreicht, bleibt dieser Puls hoch nur während eines einzigen Taktzyklus vom clk.

```
library ieee;
use ieee.std_logic_1164.all;

architecture synth of clock_divider_ent is

    signal counter:    integer range 2047 downto 0;
    signal clkDivInt: std_logic := '0';

begin

  process(clk)
  begin
      if (rising_edge(clk)) then
          counter <= counter - 1;
          clkDivInt <= '0';
          if (counter = 0) then
              counter   <= 2047;
              clkDivInt <= '1';
          end if;
      end if;
  end process;

end synth;
```

```
...
process(clk)
begin
  if(rising_edge(clk)) then
    if (clkDiv = '1') then
      ...
    end if;
  end if;
end process;
...
```

0.5.5 Vergleich von Zahlen

Nach Möglichkeit sollte immer auf "=" und nicht auf "<", ">", "<=" oder ">=" verglichen werden, da diese aufwendiger in Hardware zu implementieren sind.

0.5.6 std_logic_arith vs numeric_std

std_logic_arith wurde nicht standarisiert und ist nicht überall gleich implementiert. Stattdessen, nur **numeric_std** verwenden:

```
library ieee;
use ieee.std_logic_arith.all;  -- Vermeiden, nicht standardisiert.
use ieee.numeric_std.all;      -- Die beiden zusammen gibt nur Ärger.
```

Kategorie:Buch[3]

3 http://de.wikibooks.org/wiki/Kategorie%3ABuch

1 Autoren

Edits	User
2	Brunowe[1]
1	Dirk Huenniger[2]
1	Klaus Eifert[3]
3	MichaelFrey[4]
1	Qwertz84[5]
1	Stefan Majewsky[6]
1	ThePacker[7]

1 http://de.wikibooks.org/wiki/Benutzer:Brunowe
2 http://de.wikibooks.org/wiki/Benutzer:Dirk_Huenniger
3 http://de.wikibooks.org/wiki/Benutzer:Klaus_Eifert
4 http://de.wikibooks.org/wiki/Benutzer:MichaelFrey
5 http://de.wikibooks.org/wiki/Benutzer:Qwertz84
6 http://de.wikibooks.org/wiki/Benutzer:Stefan_Majewsky
7 http://de.wikibooks.org/wiki/Benutzer:ThePacker

Abbildungsverzeichnis

- GFDL: Gnu Free Documentation License. http://www.gnu.org/licenses/fdl.html
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. http://creativecommons.org/licenses/by-sa/3.0/
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. http://creativecommons.org/licenses/by-sa/2.5/
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. http://creativecommons.org/licenses/by-sa/2.0/
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. http://creativecommons.org/licenses/by-sa/1.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/deed.en
- cc-by-2.5: Creative Commons Attribution 2.5 License. http://creativecommons.org/licenses/by/2.5/deed.en
- cc-by-3.0: Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/deed.en
- GPL: GNU General Public License. http://www.gnu.org/licenses/gpl-2.0.txt
- LGPL: GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
- PD: This image is in the public domain.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.
- LFK: Lizenz Freie Kunst. http://artlibre.org/licence/lal/de
- CFR: Copyright free use.

- EPL: Eclipse Public License. http://www.eclipse.org/org/documents/epl-v10.php

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[8]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

8 Kapitel 2 auf Seite 35

2 Licenses

2.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”. * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

2.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "freeïn the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Versionöf the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A SSecondary Sectionïs a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The Ïnvariant Sectionsäre certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Textsäre certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent"copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparentïs called Öpaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page"means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page"means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher"means any person or entity that distributes copies of the Document to the public.

A section Ëntitled XYZ"means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as Äcknowledgements", "Dedications", Ëndorsements", or "History".) To "Preserve the Titleöf such a section when you modify the Document means that it remains a section Ëntitled XYZäccording to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "Historyïn the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History§ection. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled Äcknowledgementsör "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled Ëndorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled Ëndorsementsör to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled Ëndorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "Historyïn the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled Äcknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled Ëndorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an äggregateïf the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled Äcknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ör any later versionäpplies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site"(or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration"(or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA"means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

Ïncorporate"means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is ëligible for relicensingïf it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts."line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

2.3 GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.