Lattice™
Semiconductor
Corporation

GAL®6002 Designs Using Synario/ABEL and CUPL

Introduction

Lattice Semiconductor's GAL6002 is the most complex 24-pin PLD available. The GAL6002 is an FPLA (Field Programmable Logic Array) offering a programmable AND array and a programmable OR array. It has a total of 38 registers and two synchronous clock pins. Each of the logic outputs can be configured for multiple clock control.

This application note covers the Synario/ABEL and CUPL syntax required to fully utilize the GAL6002 and its many features. An example source file for both the ABEL and CUPL compilers is included.

Architecture Description

The GAL6002 has an FPLA architecture that contains both a programmable AND array and a programmable OR array. Inputs from the I/O pins as well as from internal registered feedback signals are brought into the AND array. The AND array then feeds the OR array, which in turn feeds the logic macrocells.

All input signals from the outside world that come into the AND array may be registered, latched, or directly connected to the AND array. Access from the outside world into the AND array is is obtained through the Input Logic Macrocell (ILMC) or by the Input Output Logic Macrocell (IOLMC). Latch or register control is applied through a pin called I/CLK. The I/CLK pin may also be brought into the AND array and used as normal input. Inputs into the AND array from the Output Logic Macrocells (OLMCs) as well as from the Buried Logic Macrocells (BLMCs) are fed directly to the AND array.

The architecture of the GAL6002 allows the OLMC to be used as a BLMC if the inverting output buffer is placed into the high impedance state. The non-driven I/Os by way of the IOLMC then have access to the AND array where they may be used as inputs.

Logic function formation is done in either the OLMC or the BLMC. There are ten OLMCs and eight BLMCs. Only the ten OLMCs have access to pins and the AND array, while the BLMCs are fed back to the AND array. The OLMC has a polarity selection capability at the D input of the Flip Flop, where as the BLMC does not.

The outputs of the OLMC drive the pins through an inverting buffer. The output enables of the inverting buffers are controlled by individual product terms.

BLMC / OLMC Configurations

Each of the OLMCs as well as the BLMCs have three possible configurations, listed below for the ABEL syntax.

1. Combinational
2. DE-type register, synchronously clocked, with a clock enable
3. D-type register, synchronously clocked

The OLMC and BLMC, if configured for registered operation, share a product term that is used for resetting the D-Flip-Flop. Each OLMC and BLMC in the registered mode of operation may be clocked from the dedicated clock pin called OCLK. If OCLK is used, then the clock may be gated off with a Clock Enable Sum Of Terms which may be unique to each OLMC or BLMC. Otherwise, a unique clock may be created from a Sum Of Terms. If a Clock Enable or a Clock Sum Of Terms is used, then either a positive or negative edge may be selected as the active edge.

Synario/ABEL Syntax

Node Definitions

The following node definitions are specific to Synario/ABEL for the use of pins as inputs to be either latched or registered.

```
“Pn#  Fnt.  Nd.   Pn#   Fnt.   Nd.
“|1   Clk         |24   VCC
“|2   InR   70    |23   InR I/O 89
“|3   InR   71    |22   InR I/O 88
“|4   InR   72    |21   InR I/O 87
“|5   InR   73    |20   InR I/O 86
“|6   InR   74    |19   InR I/O 85
“|7   InR   75    |18   InR I/O 84
“|8   InR   76    |17   InR I/O 83
“|9   InR   77    |16   InR I/O 82
“|10  InR   78    |15   InR I/O 81
“|11  InR   79    |14   InR I/O 80
“|12  GND         |13   CLK
```

Buried Register Node Definitions:

```
nodes 33, 32, 31, 30, 29, 28, 27, 26;
```

Input Pin Declared as Combinatorial:

No declaration statements are required as this is the default condition of the inputs.

Input Pin Declared as Registered:

```
IR0 node 75; “Present in the declaration
              section of source file
Pin7    Pin 7;
EQUATIONS
IR0.D = Pin7;“Present in the declaration
              section of source file
IR0.C = Pin1;
```

Input Pin Declared as Latched:

```
IL0 node 70; “This line should be
              present prior to the
              equations statement

EQUATIONS
IL0.D = Pin2; “Defining inputs to the
               latches
IL0.LE = Pin1; “Defining Latch control
                to the latched inputs
```

I/O Pin Declared as Combinatorial Input:

No declaration statements are required as this is the default condition of the inputs.

I/O Pin Used as Input and Declared as Registered:

```
IR0_I/O node 80; “Present in the
                  declaration section of
                  source file
Pin14    Pin 14;

EQUATIONS
IR0_I/O .D = Pin14; “Present in the
                     declaration section
                     of source file
IR0_I/O .C = Pin1;
```

I/O Pin Used as Input and Declared as Latched:

```
IL0_I/O  node 81;   “This line should be
                     present prior to
                     the equations
                     statement

EQUATIONS
IL0_I/O .D = Pin15; “Defining inputs to
                     the latches
IL0_I/O .LE = Pin1; “Defining Latch
                     control to the
                     latched inputs
```

OLMC as a Combinatorial:

```
“I/O pins definitions
Pin23   Pin 23;
Pin23 istype ‘com’;

EQUATIONS
Pin23 = IL0.Q & IR0.Q;
         “Straight combinatorial feed
          through of signals that have
          been previously latched and
          registered
```

OLMC as D-Type Flip-Flop:

```
Pin21   Pin 21;
Pin21 istype ‘reg_d’; “Clocked by
                       product term.

EQUATIONS
Pin21 := IL2.Q & IR2.Q;
          “Pin21 is registering values
           that have been latched
Pin21.clk = Pin10; “and registered at
                    the input pins.
                    Default is clocking
                    through OCLK pin.
```

OLMC as D-Type Flip-Flop with a Gated Clock:

```
Pin22   Pin 22;
Pin22 istype ‘reg_g’; “clocking by OCLK
                       .CE control

EQUATIONS
Pin22 := IL1.Q & IR1.Q;       “Pin22 is
       registering values
       that have been latched
       and registered at the input pins

Pin22.ce = IL3.Q;  “Clock enable control
                    is by a latched
                    value at the input
                    Pins Clocking of
                    Pin22 is through the
                    clock pin OCLK
```

BLMC as a Combinatorial:

```
BN7 node 33;
BN7 istype ‘com’; “This definition may
                   be omitted as it is
                   achieved by default
EQUATIONS
BN7= Pin10& IL0.Q & IR0.Q;
```

BLMC as D-Type Flip-Flop:

```
BN7 node 33;
BN7 istype ‘reg_d’;

EQUATIONS
BN7:= Pin10;
BN7.clk = IR0.Q; “Clock input may come
                  from OCLK or from the
                  array Default clocking
                  is through OCLK pin.
```

BLMC as D-Type Flip-Flop with a Gated Clock:

```
BN6 node 32;
BN6 istype ‘reg_g’; “clocking by OCLK
                     .CE control

EQUATIONS
BN6 := IL1.Q & IR1.Q;   “Equation is
                         clocked by the
                         pin OCLK
BN6.ce = IL3.Q;         “Clock enable is
                         by use of a
                         latched input
```

CUPL Syntax and Logic Construction

Buried Register Node Definitions:

Nodes 25 through to 32 are assigned to BLMCs 0 through 7 respectively.

OLMC Used as Buried Logic Macrocells:

Use of the OLMC as buried registers requires the use of node numbers and the NODE declaration. The node numbers 33 through 42 are assigned to the OLMCs 14 through 23 respectively. If the OLMC are not to be buried then the PIN declaration statement is to be used.

Device Declaration:

```
Device g6002;
```

Inputs declared as Combinatorial:

Inputs that do not have extensions associated with them are configured as combinatorial inputs that go straight through to the AND array.

Example of Usage —

Pin declarations:
```
Pin 2 = IN2;
Pin 14 = OUT;
```

Use in equations:
```
OUT = IN2;
```

Use of I/O Pins as Combinatorial Inputs —

Pin declarations:
```
Pin 15 = OUT2;
```

Use in equations:
```
OUT = OUT2;
```

Inputs Declared as Registered:

The extension .DQ declares pins that are used as inputs to be configured as registered.

Example of Usage—
Use of dedicated input pins as registered inputs —

Pin declarations:
```
Pin 14 = OUT;
Pin 2 = IN2;
```

Use in equations:
```
OUT = IN2.DQ;
```

Use of I/O pins as registered inputs to combinatorial outputs —
Pin declarations:

```
Pin 14 = OUT;
Pin 15 = OUT2;
```

Use in equations:

```
OUT = OUT2.DQ;
```

Inputs Declared as Latched:

The extension .IOL declares pins that are used as inputs to be configured as latched.

Example of Usage —
Use of dedicated input pins as latched inputs —

Pin declarations:

```
Pin 14 = OUT;
Pin 2 = IN2;
```

Use in equations:

```
OUT = IN2.IOL;
```

Use of I/O pins as latched inputs —

Pin declarations:

```
Pin 14 = OUT;
Pin 15 = OUT2;
```

Use in equations:

```
OUT = OUT2.IOL;
```

BLMC and OLMC as a Combinatorial:

For use as a combinatorial output no extension is to be used as this is the default state of the outputs.

Example of OLMC Usage —

Pin declarations:

```
Pin 2 = IN2;
Pin 15 = OUT2;
```

Use in equations:

```
OUT2 = OUT2;
OUT2.OE = IN2;
```

If an output enable equation is not present the compiler causes the output always to be driven.

Example of BLMC Usage —

Pin declaration and BLMC declaration:

```
Pin 2 = IN2;
NODE   BLMCOUT1;
```

Use in equations:

```
BLMCOUT1 = IN2;
```

Note that there is no output enable capability in the BLMC as there is no direct access to the pins. Outputs from the BLMC are brought directly back into the AND array.

BLMC and OLMC as D-Type Flip-Flop:

Clock for this configuration is obtained from a product term rather than the dedicated clock pin.

Example of OLMC Usage —

Pin declarations:

```
Pin 2 = IN2;
Pin 15 = OUT2;
```

Use in equations:

```
OUT2.CK = IN2;
OUT2.OE = IN2;
OUT2.AR = IN2;
OUT2.D = OUT2 & IN2;
```

Example of BLMC Usage —

Pin declaration and BLMC declaration:

```
Pin 2 = IN2;
NODE   BLMCOUT1;
```

Use in equations:

```
BLMCOUT1.CK = IN2;
BLMCOUT1.AR = IN2;
BLMCOUT1.D = OUT2 & IN2;
```

BLMC and OLMC as D-Type Flip-Flop with a Gated Zclock:

Clock for this configuration is obtained from the dedicated clock pin rather than a product term. The clock can be gated off by use of a Clock Enable product term.

Example of OLMC Usage —

Pin declarations and BLMC declaration:

```
Pin 2 = IN2;
Pin 15 = OUT2;
```

Use in equations:

```
OUT2.CE = IN2;
OUT2.OE = IN2;
OUT2.AR = IN2;
OUT2.D = OUT2 & IN2;
```

Example of BLMC Usage —

Pin declarations:

```
Pin 2 = IN2;
NODE   BLMCOUT1;
```

Use in equations:

```
BLMCOUT1.CE = IN2;
BLMCOUT1.AR = IN2;
BLMCOUT1.D = OUT2 & IN2;
```

LATTICE SEMICONDUCTOR CORPORATION
5555 Northeast Moore Court
Hillsboro, Oregon 97124 U.S.A.
Tel.: (503) 681-0118
FAX: (503) 681-3037
http://www.latticesemi.com

November 1996