XAPP093 November 10, 1997 (Version 1.1)

Application Note by Peter Alfke

Introduction

All Xilinx SRAM-based FPGAs can be in-system configured and re-configured an unlimited number of times.

This application note describes the procedures for reconfiguring the more traditional Xilinx FPGAs of the XC3000, XC4000, and XC5200 families.

All configuration information is stored in latches that are loaded serially, conceptually like a shift register. There are several different bit-serial or byte-parallel configuration data interfaces, selected by logic levels on three mode inputs, but – with the exception of the XC5200 Express mode – they all result in the bit-serial loading of the configuration latches. The byte-parallel interfaces in Master Parallel and Peripheral modes act just as an 8-bit parallel-to-serial converter. Between devices in a daisy-chain, the configuration information is transmitted bit-serially with a common Configuration Clock (CCLK). In Master and Peripheral modes, CCLK is generated by the lead FPGA device, in Slave Serial mode, CCLK comes from an external source.

Reconfiguration of an operational device, or a daisy-chain of devices, goes through the following sequence of events:

- Reconfiguration is initiated by pulling a specific device pin Low.
- First, all outputs are 3-stated, except HDC = High, $\overline{\text{LDC}}$ and DONE = Low
- Then, all internal registers, flip-flops and latches, as well as the configuration storage latches are cleared. During this time, the $\overline{\text{INIT}}$ output is being pulled Low.
- Then, the Mode inputs and $\overline{\text{RESET}}$ or $\overline{\text{PROGRAM}}$ inputs are sampled to determine the selected configuration mode and whether to start the new configuration process, or to wait.
- Then configuration data is accepted and loaded into the internal latches and distributed through the daisy-chain.
- When all configuration information has been entered, the user outputs are activated, DONE goes High and the internal reset is released, all in the order specified in the configuration bitstream. All devices in a daisy-chain perform each of these operations in synchronism.

Important Considerations

- Reconfiguration is “all or nothing”. There is no way to restrict reconfiguration to a part of the chip.
- Reconfiguration takes a specific time, determined only by device type, size and clock speed, independent of the particular configuration pattern. Configuration takes from tens to hundreds of milliseconds. During that time, all user-outputs of the device, or the whole daisy-chain of devices, are 3-stated with weak internal pull-ups, except for HDC and $\overline{\text{LDC}}$, which are active High or Low respectively.
- All user-data stored in registers, flip-flops or latches is erased. There is no way to retain data inside the device from one configuration to the next.

These limitations are absolute. If they are not acceptable, the user must resort to creative solutions, like piggy-backing multiple devices.

The designer of reconfigurable applications should be familiar with the normal configuration process of each device, as described in the individual product descriptions. There is also pertinent information about daisy-chain operation, especially about mixed daisy chains, in other application notes.

Interconnecting the $\overline{\text{INIT}}$ pins of all devices in a daisy-chain is mandatory for reconfiguration, since this is the only way to guarantee that the master device does wait for the rest of the daisy-chain to be cleared, before starting the reconfiguration. Only the first configuration after power-up makes the master device spend four times as many clock periods as any slave during the initial clear operation, so that the master cannot possibly get ahead of the slaves. Reconfiguration, however, does not slow down the master this way, so the interconnection of all $\overline{\text{INIT}}$ pins must serve that same purpose.

In Master Serial mode, it is highly recommended that the active Low level of $\overline{\text{INIT}}$ be used to reset the XC1700-family Serial PROM.

Reconfiguration Time

Reconfiguration time is usually more critical than the original power-on configuration time, which is often masked by the general power-on delays.

Here are some suggestions to reduce reconfiguration time.

- A daisy-chain is obviously not conducive to fast configuration, it should be broken up into shorter blocks, perhaps single devices. Multiple devices can be configured in parallel, but can still use a common CCLK, and can also be made to start up together. If the devices differ in size or family, they should all be given the same length count as the largest device in the group.
- Configuration Mode
  Parallel and Peripheral modes are not any faster than Master Serial mode, since all modes (with the exception of XC5200 Express mode) internally operate on serial data. The internally generated CCLK frequency is guard-banded to never approach the upper limit of what the device can tolerate. Therefore, the fastest possible configuration mode for XC3000 and XC4000-series devices is Slave Serial, with an external well-controlled source for CCLK. Its frequency can be up to 10 MHz for all 5-V devices, and there are ways to increase the average clock rate well beyond that, but they require dynamic clock frequency changes and an intimate understanding of the configuration frame structure.
  At 10 MHz, configuration time per device ranges from 1.5 ms for the XC3020A to 42 ms for the XC4025E and 192 ms for the XC4085XL.
- Possible Contention Problems:
  Certain user outputs become active during the configuration process:
  Address outputs during Master Parallel mode, Chip Select and Ready/Busy during Peripheral modes.
  The designer must make sure that these active outputs do not cause contention with other logic that might use the same pins as device inputs.

Initiating Reconfiguration in Different Xilinx Device Families

XC3000 Series

There are three alternatives:

1. Pull $\overline{\text{RESET}}$ Low while DONE is permanently grounded externally.

This is the simplest scheme, but it precludes the use of $\overline{\text{RESET}}$ to clear the flip-flops and latches in the operating user-design. $\overline{\text{RESET}}$ must be pulled Low for more than six microseconds to overcome its internal low-pass filtering. Configuration starts when $\overline{\text{RESET}}$ has gone High again.

2. Pull DONE Low with an open-drain (“open-collector”) output. This assumes that DONE was High, i.e. that the previous configuration was successful. Reconfiguration starts as soon as the internal memory has been cleared. DONE can be released anytime.

3. Pull DONE Low with an open-drain (“open-collector”) output and pull $\overline{\text{RESET}}$ Low. Keep $\overline{\text{RESET}}$ Low for at least six microseconds while DONE is Low. DONE can be released anytime after that, or not released at all. See alternative 1.

XC4000 Series and XC5200 Family

Pull the $\overline{\text{PROGRAM}}$ input Low for at least 0.3 microseconds to initiate clearing the configuration memory, then pull $\overline{\text{PROGRAM}}$ up to start the new configuration process.

While PROGRAM is held Low, a Low level on $\overline{\text{INIT}}$ indicates that the device is continuously clearing the configuration memory. When $\overline{\text{PROGRAM}}$ has been pulled up, $\overline{\text{INIT}}$ stays Low during one more clear operation, then goes High.

All device families, except the original XC4000, have a continuously active pull-up resistor on the $\overline{\text{PROGRAM}}$ pin.

FPGAs Can Control Their Own Reconfiguration

Pulling PROGRAM, RESET or DONE low can trigger a reconfiguration, as described above. When a user output is connected to drive the reconfiguration pin, the FPGA can trigger its own reconfiguration. Although the triggering output will go 3-state once reconfiguration is initiated, this trigger operation is reliable.

Such auto-reconfiguration offers interesting opportunities for small systems using a single FPGA in Master Parallel configuration mode. A manually operated switch selects the most significant address bits of the PROM, and the FPGA compares the switch settings against a stored value. Upon detecting a difference, it can trigger reconfiguration that is loaded from the newly selected PROM address range. Or an external CMOS register can be loaded with the intended reconfiguration address range and then control the upper bits of the PROM address