Early output logic and Anti-Tokens

Charlie Brej (cb@cs.man.ac.uk)

Introduction

The synchronous design model has proved extremely successful for designing logic for many decades. One of the main advantages of global synchronisation is that the designer is freed from worries about inter-unit communication; it can be assumed that any units in a device can communicate on any clock edge. Unfortunately synchronisation across a system, even a single chip, is increasingly difficult. One reason for this is the very high clock speeds employed in high performance designs. The effect of this starts to become problematic as the reachability of silicon area within one clock cycle becomes smaller.

Asynchronous logic

To address this some designers have looked at asynchronous communications, both between functional units and within blocks themselves. Such techniques allow elasticity in the system, alleviating or eliminating timing closure problems. In asynchronous designs, instead of all data progressing from one stage to another on the strike of the clock, the data progresses independently, maniging their own timing. The data packets flowing through the system can be thought of as tokens.
Units have a number of input and output channels where they communicate tokens with other units. Each unit waits until it has accepted all tokens on all its input channels, processes the data, and generates tokens on its outputs.

Early output logic

In many situations some of the inputs are redundant to the generation of the correct output. These ‘early output’ cases are common in both low level circuits, such as OR gates where one of the inputs is one and high level circuits, such as multiplexers where only one of the data inputs is neccesary to generate a result. In a multiplexer the necessity of a particular input is reliant on other input data (in this case the select input). The forced wait for all inputs to arrive before a result is generated is unnecessary and slows down the operation of the system.
Generating the result once sufficient data to complete the operation has arrived removes some of the unnecesary synchronisations and allows a more ‘free’ operation. However, although the result can be generated early, the unit has to wait for the late input to arrive in order to acknowledge it.

Anti-tokens

The late, unnecessary inputs stall the unit which can reduce the system performance. The unit instead of waiting for the late input can send a signal back up the pipeline towards the late token in order to cancel it and then continue. This signal travels in the opposite direction to normal tokens and when a collision happens between a it and token both are eliminated. Because of its behaviour it is called an ‘anti-token’.
Anti-tokens can progress backwards through the pipeline and when entering a unit which generates the late token, will remove all inputs present and generate anti-tokens on further input channels. This removes all tokens which would have combined to create the late token which was to be destroyed.
The action of removing the result closer to its origin both increases the system performance by releasing stages to operate on the next input set and reduces power consumption by stopping speculative operation from being conducted.