

A Fast Crank Angle Resolved Zero-Dimensional NO_x Model Implemented on a Field-Programmable Gate Array

Kenan Muric

Scania CV AB / Lund University

Per Tunestal

Lund University

Ola Stenlaas

Scania CV AB

ABSTRACT

In the automotive industry, the piezo-based in-cylinder pressure sensor is getting commercialized and used in production vehicles. For example, the pressure sensor offers the opportunity to design algorithms for estimation of engine emissions, such as soot and NO_x, during a combustion cycle. In this paper a zero-dimensional NO_x model for a diesel engine is implemented that will be used in real time. The model is based on the thermal NO_x formation and the Zeldovich mechanism using two non-geometrical zones: burned and unburned zone. The influence of EGR on combustion temperature was modeled using a well-known thermodynamic identity where specific heat at constant pressure is included. Specific heat will vary with temperature and the gas composition. The model was implemented in LabVIEW using tools specific for an FPGA (Field-Programmable Gate Array). In order to simplify and implement the model, least squares criterion based polynomial approximations are used that enables the utilization of fast algorithms as well as sub-routines (sub-VIs). The sub-routines can be used to save space on the Field Programmable Gate Array (FPGA) and thus minimizing the risk of potential issues regarding overmapping of the hardware. In this case the interpolating functions are polynomials that only consume addition and multiplication operations. This is suited for the objective in mind due to the fact that the model tailored for an FPGA cannot, in a sufficient manner, handle highly complex calculations nor divisions. The time results obtained during the execution of the model indicates that it is possible to update the NO_x, at a given temporal state, well below the time corresponding to a crank angle degree. The FPGA NO_x model was tested against measurement data collected from a Scania engine. The time needed to execute an iteration of the model was approximately 3 μs.

CITATION: Muric, K., Tunestal, P. and Stenlaas, O., "A Fast Crank Angle Resolved Zero-Dimensional NO_x Model Implemented on a Field-Programmable Gate Array," *SAE Int. J. Engines* 6(1):2013, doi:10.4271/2013-01-0344.

INTRODUCTION

The next step in high level engine control development will be to control the combustion behavior while it is occurring. Realizing this is not trivial due to the time constraints imposed by an engine running at a speed of for example 1500 rpm. However, previous work in the area of in-cylinder combustion control has demonstrated that it is possible. Zander [1] presented a crank angle resolved controller that reduced the cycle-to-cycle variations by controlling the fuel injection duration for each combustion event. The model used by Zander, in the control strategy, was a heat release calculation based on pressure measurements

from a piezo-based pressure sensor. The feedback consisted of a proportional controller that was shown to be sufficient in minimizing the cycle-to-cycle variation of pressure and accumulated heat release. Zheng [2] has also conducted studies on the validity region of heat release analysis using an FPGA for cycle-to-cycle control. The variables of interest for feedback control in Zheng's case were CA50, max pressure and rate of pressure rise.

Crank angle resolved fast NO_x models have been a research topic in recent years. The first models of CA-resolved NO_x formation introduced were single-zone and two zone models [3][4]. Schröer [3] calculated the rate of heat

release using a one zone model, and to calculate the combustion temperature he used an average temperature calculated based on the ideal gas law relationship. NO_x could then be calculated using empirically derived relations. Schröder did not use any chemistry to describe the relation between NO_x , oxygen, nitrogen and temperature. Heider [4] used another approach where two zones are applied with an additional model describing the heat transfers between the two zones. Essentially, his approach is to define two separate zones, burned and unburned zone, to calculate the NO_x by applying the first law of thermodynamics, the extended Zeldovich mechanism and the ideal gas law. Previous researchers from Lund [5][6][7] have all continued on this modeling approach and contributed with improvements in the field of zero-dimensional NO_x models. Egnell [5] showed that certain calculation steps in the zero dimensional model can be tabulated to reduce the computational load. Fröberg [6] implemented a fast crank angle resolved model possible to calculate NO_x on a personal computer in the range of a minute. Wilhelmsson [7] made a simplified NO_x model suitable for embedded systems. However, the model presented by Wilhelmsson does not try to account the effects of EGR and temperature losses. The model also utilizes large tables to implement the Zeldovich mechanism, which is a drawback if one wants to implement it on an FPGA for engine control purposes. In the model presented in this paper, EGR and temperature losses are accounted for and the original Zeldovich mechanism is mathematically reformulated in order to simplify the calculation and make suitable for an FPGA.

MODEL OF NO_x FORMATION

The model presented here is based on previous work by others in the field of NO_x formation modeling [5][6][7]. The model is zero-dimensional and assumes two distinct zones during the combustion process, an unburned and a burned zone. The unburned zone consists of fuel, EGR and air while the burned zone contains the products formed after the combustion. Temperatures are calculated assuming an isentropic compression and expansion in the unburned zone. The temperature increase in the burned zone due to combustion is calculated using the thermodynamic identity:

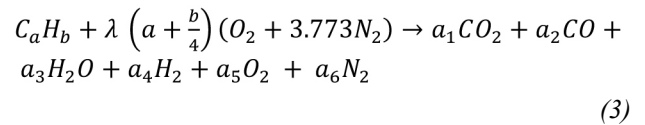
$$dQ = mc_p dT \quad (1)$$

The calculation contains a specific heat, mass and a quantity of heat. The heat is calculated using the heat release equation based on the first law of thermodynamics where $\frac{dQ_{HT}}{d\phi}$ represents the heat transfers:

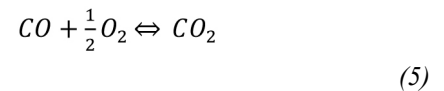
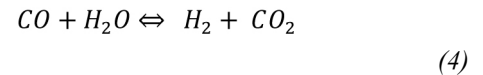
$$\frac{dQ_{comb}}{d\phi} = \frac{\gamma}{\gamma-1} p \frac{dV}{d\phi} + \frac{1}{\gamma-1} V \frac{dp}{d\phi} + \frac{dQ_{HT}}{d\phi} \quad (2)$$

The measurement input in the heat release equation is crank angle-resolved (CA-resolved) cylinder pressure

measurement. The specific heat at constant pressure can be calculated using NASA tabulations of specific heats [8]. The impact of the EGR on the combustion temperature is integrated in the model through the dependency of the specific heat on the levels of EGR. It is worthwhile to notice that the specific heat will depend on two variables. At a given global lambda the variables will be temperature and the fraction of EGR in the cylinder gas content. Also, temperatures will decrease due to radiative heat transfer, temperature drop due to dissociation of carbon dioxide and water, and gas expansion. These factors are accounted for in the model used in the final implementation. Once the temperature in the burned zone is known it is possible to calculate the NO_x levels using the thermal mechanism based on the original Zeldovich mechanism. The combustion chemistry for an arbitrary fuel, $C_a H_b$, is given as:



A set of equations can be derived using the reaction above and the water-gas shift reaction stated as:



This will give a system of equations [5]. a_1, a_2, a_3, a_4, a_5 and a_6 corresponds to the equilibrium fractions of CO_2, CO, H_2O, O_2 , and N_2 respectively. A local lambda is defined and set as $\lambda_{local} = 1$.

$$a_1 + a_2 = a \quad (6)$$

$$2(a_3 + a_4) = b \quad (7)$$

$$2a_1 + a_2 + a_3 + 2a_5 = 2\lambda_{local} \left(a + \frac{b}{4} \right) \quad (8)$$

$$a_6 = 3.773\lambda_{local} \left(a + \frac{b}{4} \right) \quad (9)$$

$$a_1 a_4 = K_{p1} a_2 a_3 \quad (10)$$

$$K_{p2}^2 a_2^2 a_5 = \frac{a_1^2 \sum_{i=1}^6 a_i}{P} \quad (11)$$

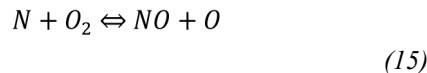
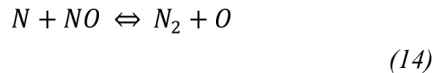
As stated, λ_{local} is a local lambda value that is set to equal 1. The reasoning behind this is based on the conceptual model of diesel combustion described by John Dec [9] where

combustion at the diffusive layer occurs at stoichiometric conditions. The equations can be solved using a Gauss-Newton solver and an initial guess based on the combustion chemistry, omitting any possible water-gas shift mechanism. The equilibrium constants (K_{p1} and K_{p2}) needed in the equations describing the water-gas shift can be calculated by applying following equations (G^0 = Gibbs free energy, H^0 = Enthalpy, S^0 = Entropy and v_i = Stoichiometric coefficient):

$$G^0(T) = H^0 - S^0T \quad (12)$$

$$\ln K_p = -\sum_i \frac{v_i G_i^0}{RT} \quad (13)$$

One should notice that the equations above are dependent on temperature and normalized pressure (normalized with 1 atm). Finally, the Zeldovich mechanism can be applied to calculate the rate of NO_x production in the burned zone. The two reactions used are the original reactions presented by Zeldovich [10]:



The formation rate (concentration change per time unit) can then be written as:

$$\frac{d[\text{NO}]}{dt} = \frac{2R_1 \left(1 - \frac{[\text{NO}]^2}{[\text{NO}]_e^2} \right)}{\frac{[\text{NO}]}{[\text{NO}]_e} R_1 + 1} \quad (16)$$

Time integration of the formation rate will then give the NO_x concentration at current time. In Figure 1A a CA-resolved NO_x -result is shown while in Figure 1B a plot of NO_x estimation versus NO_x measurements is presented.

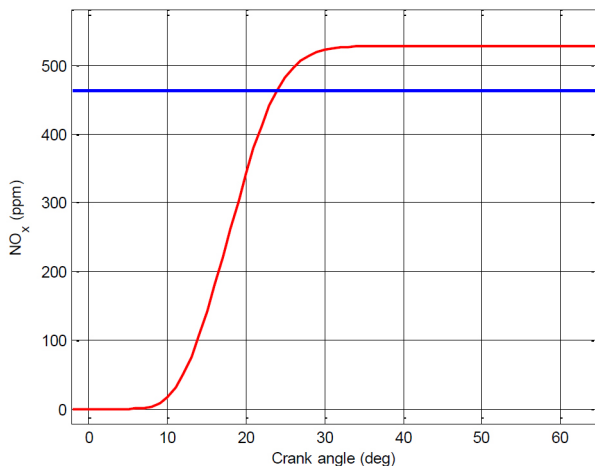


Figure 1A. Example of a crank angle resolved NO_x formation simulation conducted on measurements from a Scania 480 hp engine.

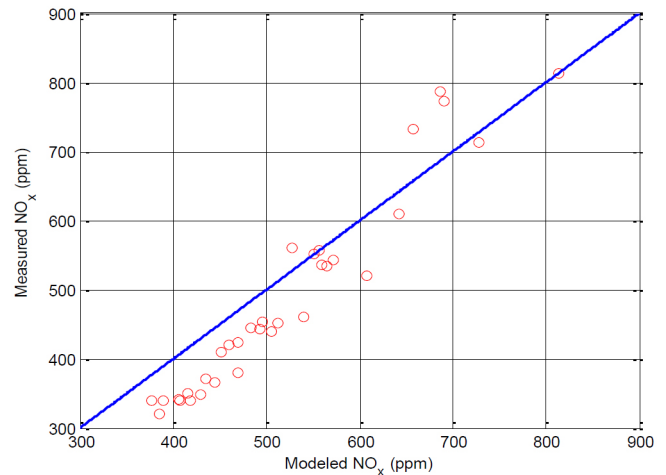


Figure 1B. Results obtained on a set of data on the same engine. The NO_x values collected for the evaluation of the model are averaged measurements (i.e. measurements are collected during the amount of combustion cycles corresponding to a time frame of three minutes and the average value is used) from the exhaust pipe.

HARDWARE AND SOFTWARE

The FPGA was invented by Ross Freeman during the 80's [5]. The main advantage of a Field-Programmable Gate Array compared to an ordinary processor is the possibility to run segments of code in a truly parallel manner. The ordinary processor executes in a sequential manner where tasks are prioritized and run depending on the priority level. This is not the case when an FPGA is used for real-time executions.

The principal components of the FPGA are the logic blocks, connect blocks and switch boxes (see Figure 2). The logic block contains of multiple input lookup tables that enable the implementation of mathematical operations such as addition, Boolean logic, subtraction, multiplication etcetera. The logical blocks are designed and embedded in a routing structure. Input and output signals are routed to the logic block through connection blocks. The switch box seen in Figure 2 provides the possibility to reroute the connection between different logic blocks. The switch box has essentially two possible routing resources available, horizontal and vertical, which makes the rerouting of signals possible [11]. The FPGA is a reconfigurable device where a high-level software can be used for configuration of the hardware. The FPGA is not quite as fast as an ASIC, but the straightforward approach to reconfigure the device makes it a suitable option.

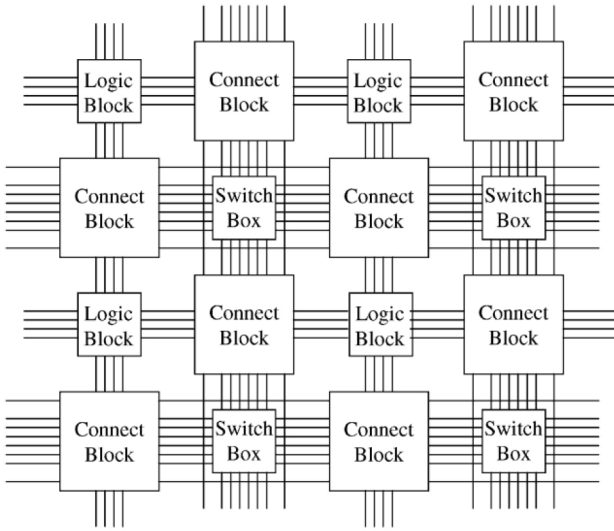


Figure 2. The generic layout of the FPGA routing architecture of a so called island structure [11].

In this project two properties in the software are essential to utilize the benefits of the FPGA. It should be straightforward to implement parallel code as well as integrating the code with the FPGA hardware. In this project LabVIEW was used together with a National Instruments FPGA card.

The programmed code executes from left to right in a parallel manner. In the case where a calculation is dependent on a variable not yet available, the particular code segment in question will idle until the variable is finally accessible. A segment of code with input and output nodes is called a VI (virtual instrument) and basically constituting a class. It is possible to create sub-VIs which corresponds to sub-routines used by a VI higher up in the hierarchy. The purpose of implementing a sub-VI is twofold. First of all, it is possible to make the code simpler to follow if one implements a reoccurring code segment in a separate subroutine called by different virtual instruments. Another aspect is the fact that one can minimize the hardware usage by making the subroutines non-reentrant. This means that only a VI at a time can access the sub-routine and hence there is only need of a single instance of the subroutine on the FPGA hardware. This is one possible approach to reduce the risk of overmapping the hardware. Overmapping is a result of a larger resource usage than 100%, which is of course not possible to achieve in reality. Maintaining the FPGA programs in the range of the resources available can also be achieved by compilation constraints. The compilation can be set to achieve good FPGA area utilization, reducing compilation time and faster execution. Different settings of the compilation will render different area distributions on the FPGA card.

SETUP

A NI PCIe RIO-7852R was bought from National Instruments together with a license for the LabVIEW 2011

software. The FPGA was connected to a laptop through an external PCI bus as shown in Figure 3. The connection between the external bus and the laptop was through an express card. The code is compiled on the laptop and then flashed to the FPGA card for time result extraction of an iteration of the model.



Figure 3. Setup used in the simulations. The external PCI bus is a PCIE4MULTI-EX containing the FPGA card NI PCIe RIO 7852R. The laptop is connected to the bus via an express card.

POLYNOMIAL APPROXIMATIONS

In order to evade divisions and nonlinear calculations, while at the same time creating non-reentrant subroutines, it is possible to approximate some of the calculation steps in the model with polynomials. Polynomials are used to calculate the oxygen concentration, specific heat, square root of the oxygen concentration and temperature drop due to dissociation of carbon dioxide and water. The polynomials are derived using a least squares criterion where a polynomial function of two variables is derived for every case: temperature drop, specific heat etcetera. The equations used to derive the polynomials are:

$$E(\bar{P}) = \sum_{i=1}^n (z_i - \bar{P}\bar{Q}_i)^2 \quad (17)$$

$$\nabla E(\bar{P}) = -2 \sum_{i=1}^n (z_i - \bar{P}\bar{Q}_i) \bar{Q}_i = 0 \quad (18)$$

The variable z contains the output, $\bar{P}$ contains the polynomial coefficients and $\bar{Q}$ corresponds to the variables used to interpolate. In the case of specific heat at a given lambda this will give a polynomial function $f(EGR, T)$. The balance and equilibrium equations in (6),(7),(8),(9),(10),(11) were solved by implementing a Newton-Gauss solver and then approximating the output concentration with polynomials. The approximating polynomial is a function of pressure and temperature, at a given constant local lambda (in this case $\lambda_{local} = 1$). The same was the case for the approximation of the temperature drop due to dissociation of carbon dioxide and water. It is of course straightforward to realize that only one implementation of the polynomial