May 2020

Order Independent Tiebreaker For Motion Search

Anonymous

Order Independent Tiebreaker For Motion Search

ABSTRACT

A motion search integrated circuit (IC) is configured to determine the motion of objects in a video. In motion search, multiple candidate motion vectors may often have the same cost, such that the multiple candidates are each optimal and equally acceptable. The multiplicity of candidates leads to difficulties in verifying the IC design, e.g., in comparing the output of the IC to a reference (golden) model implemented in software. This disclosure describes tiebreaking techniques that enable the IC, as designed in a hardware description language, and the reference model in software to identify the same optimal motion vector for a given video.

KEYWORDS

- Motion vector
- Motion search
- Regression testing
- Hardware description language (HDL)
- IC design
- Video compression

BACKGROUND

A common paradigm in hardware engineering is to develop, in parallel to the design of an integrated circuit (IC) in a hardware description language (such as VHDL), a reference model of the IC in a high-level language (such as C). The reference model serves as a target model for the behavior of the IC. The reference model is also known as a behavioral model or a golden model. The advantage of having a reference model in parallel to the HDL design of the IC is that the reference model is much faster to develop and execute. Bugs in the HDL design of the IC can be

rapidly uncovered by comparing outputs of the reference and the HDL versions of the IC in response to a suite of test inputs.

A motion search IC is an IC that determines the motion of objects in a video. In motion search, multiple candidate motion vectors may often have the same cost such that multiple candidates are each optimal and equally acceptable. This can lead to the HDL design and the reference model selecting different optimal candidates. In theory, the differing optimal-candidate selections of the HDL and the reference models should not matter; in practice, however, it is extremely difficult to differentiate the differences in selections of motion vectors due to such factors from errors caused by a bug in the design of the IC. To ensure that regression testing over a large number (e.g., thousands to millions) of test patterns can be conducted reliably, it is necessary that the HDL and the reference models agree on their selections of the optimal motion-vector candidate for each test case.

Traditional techniques to achieve a match between the HDL and the reference model impose an ordering during optimal candidate search. For example, search can be conducted in a pixel-raster order, by following the ordering of a spiral that emanates from a center (zero motion vector), etc. Further order can be enforced, e.g., by preferring smaller motion vectors. While these techniques do result in a matched selection of the optimal motion-vector candidate between the HDL and the reference models, ordered search is compute-intensive and often results in an unnatural order of processing of the input. For example, the IC may optimally operate in an order dictated by the fetching of pixels from the memory; enforcing a search order leads to design complications and additional hardware. Spiral order of search is not parallelizable in hardware. Also, ordered search is unscalable: when the search area (the region of the video frame(s) within which an object could’ve moved) changes, the ordering can change.

DESCRIPTION

Per techniques of this disclosure, the search order for the HDL model is uncoupled from that of the reference model, e.g., each model can search for optimality in a mutually independent manner and based on its context. If the HDL and the reference models arrive at different candidate motion vectors that have the same cost, a tiebreaker, described below in greater detail, is used by each model to select an optimal motion vector from the candidates. When the tie breaking procedure finishes, the reference and the HDL models converge to the same optimal motion vector.

```
bmv: best motion vector
cmv: candidate (or current) motion vector

if abs(cmv) < abs(bmv)
   bmv ← cmv;
elseif abs(cmv.y) < abs(bmv.y)
   bmv ← cmv;
elseif cmv.y > 0
   bmv ← cmv;
elseif cmv.x > 0
   bmv ← cmv;
```

Fig. 1: Tiebreaker to determine an optimal motion vector used in reference and HDL models

Fig. 1 illustrates pseudocode for a tiebreaker to determine an optimal motion vector common to both reference and HDL models. A motion vector includes three components, e.g., an x-component, a y-component, and a cost. For a motion vector `mv`, the x-component is denoted as `mv.x`, the y-component as `mv.y`, and the cost as `mv.cost`. As mentioned before, the reference model and the HDL model each identify multiple candidate motion vectors that have the same cost. For example, the vectors selected by the HDL and the reference models can differ in x-component, y-component, or both.

For each element within the set of candidate motion vectors with the same cost, each model executes a tiebreaker as follows. The best motion vector thus far found is denoted `bmv`. A candidate motion vector, e.g., one with the same cost as `bmv` but with different x- or y-components, is denoted as `cmv`. The candidate motion vector replaces the best motion vector if

- the absolute value of the `cmv` is less than that of `bmv`; and if not,
- if the absolute value of `cmv.y` is less than that of `bmv.y`; and if not,
- if `cmv.y` is greater than zero; and if not,
- if `cmv.x` is greater than zero.

```
1. if abs(mv1.x) + abs(mv1.y) == abs(mv2.x) + abs(mv2.y), goto 2;
   else
   if abs(mv1.x) + abs(mv1.y) < abs(mv2.x) + abs(mv2.y)
     mv1 wins;
   else mv2 wins;

2. if abs(mv1.y) == abs(mv2.y), goto 3;
   else
   if abs(mv1.y) <  abs(mv2.y) mv1 wins;
   else mv2 wins;

3. if mv1.y == mv2.y, goto 4;
   else if mv1.y > 0 mv1 wins;
   else mv2 wins;

4. if mv1.x > 0 mv1 wins;
   else mv2 wins
```

Fig. 2: Tiebreaker to determine an optimal motion vector between two candidate motion vectors

Fig. 2 illustrates an alternate tiebreaker procedure to determine an optimal motion vector between two candidate motion vectors `mv1` and `mv2` of equal cost but unequal x- or y-components. Per this tiebreaker, a candidate motion vector whose sum of absolute values of components is smaller is preferred over other motion vectors of the same cost. If the sums of the absolute values of components of the candidate motion vectors are equal, then the motion vector

with smaller absolute y-component is chosen. If the candidate motion vectors have equal absolute y-components then the motion vector with positive y-component is chosen, failing which the motion vector with positive x-component is chosen.

In this manner, even if the reference and the HDL models start off with different candidate motion vectors of the same cost, and even if they each searched for the motion vectors using distinct search orderings, it is ensured that both models still arrive at the same optimal motion vector. The techniques give preferences to motion vector candidates with smaller absolute value, since smaller motion vectors are more probable than larger motion vectors. In this aspect, the techniques are similar to spiral ordering.

The described techniques are scalable: the tiebreaker has no extra cost for variable-size search areas. The techniques are also parallelizable: the search space can be split into blocks whose results can be combined to discover the global optimum. The small increase in chip area needed to implement the tiebreaker is mainly in the form of comparators. These present no substantial increase in complexity, and are much less complex than traditional techniques of changing search order.

CONCLUSION

A motion search integrated circuit (IC) is configured to determine the motion of objects in a video. In motion search, multiple candidate motion vectors may often have the same cost, such that the multiple candidates are each optimal and equally acceptable. The multiplicity of candidates leads to difficulties in verifying the IC design, e.g., in comparing the output of the IC to a reference (golden) model implemented in software. This disclosure describes tiebreaking techniques that enable the IC, as designed in a hardware description language, and the reference model in software to identify the same optimal motion vector for a given video.