Multi-Dimensional and Multi-Topological Programming

René Heinzl & Philipp Schwaha

SHENTEQ

Design for Science

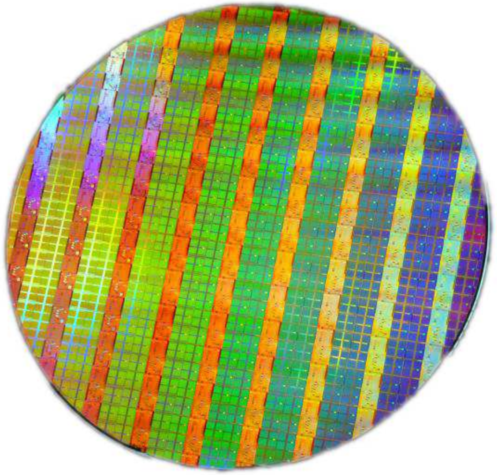
Our deal

Simulation of different physical effects

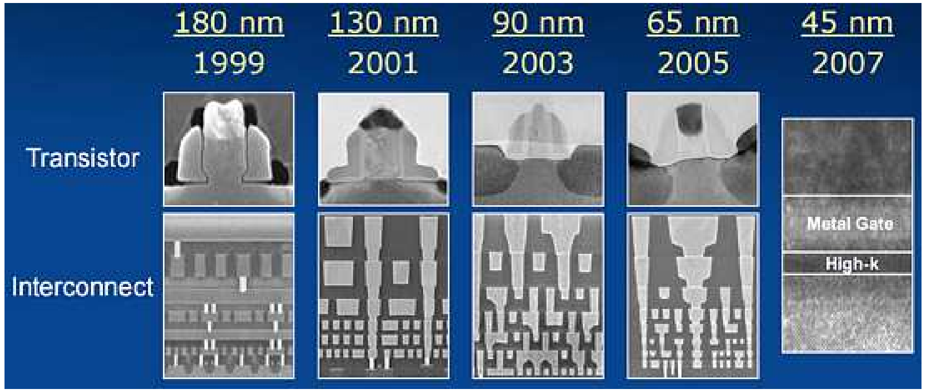
- Diffusion
- Wave propagation
- Quantum phenomena

Simulation on different scales

- Whole wafers (increasing in size)
- Devices (shrinking in size)
- Increasing aspect ratios

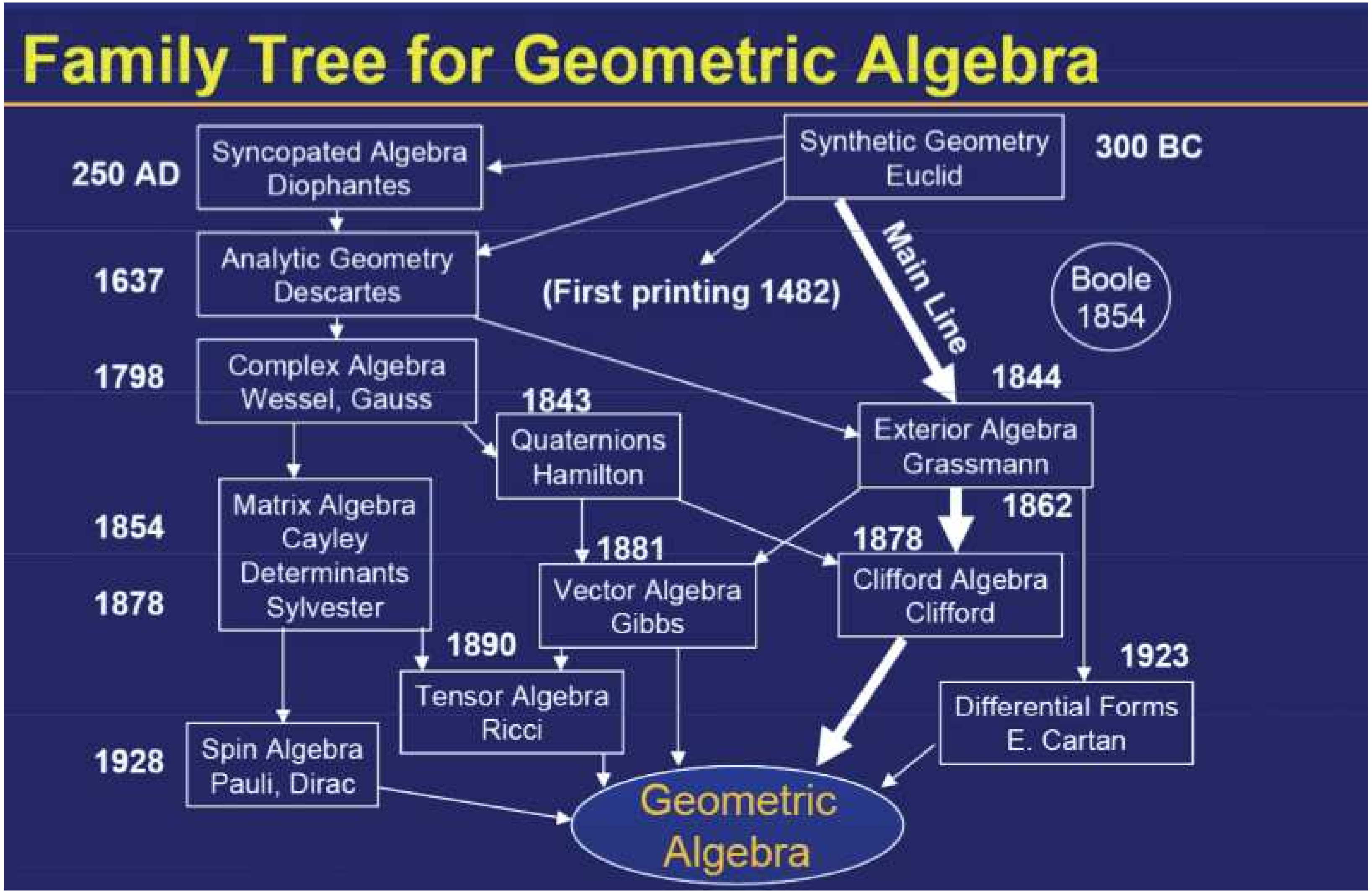


Intel 300mm wafer for 45nm technology node.



Evolution of shrinking (source: intel.com).

(Historic) Illustration



Concepts and us

Mathematical foundations

Group (abelian), monoid, field, vector space, dual space, ...

co/contra-variance

(Co)Vectors, differential forms, base, ...

Maintain integrity of mathematical entities

(Co)Vector, tensor, ...

closed/exact forms, integrability

(Unified) Treatment of discretization schemes

Finite differences

Finite volumes

Finite elements

Desirables

Queryable concepts for containers

- is_associative

Make compile/run time distinctions queryable

- e.g., containers: boost::array<int> vs. std::vector<int>

Control for increasing complexity in the topics of

- Geometry - e.g. metric concepts
- Dimensionality - e.g. increase in available combinations
- Topology - e.g., implicit (structured grid) vs. explicit (unstructured mesh)

Easing the compile time / run time transition

Sample algorithms

Finite Elements:

```
for (long iri = 0; iri < gsse::size(ips); ++iri)
{
 gsse::fem::specific_integration_point ip( ips[iri] , element_trans) ;

 DiffOp::generate_matrix(fe, ip, mx_B);
 dmatop.generate_matrix (fe, ip, mx_D);

 NumericT fac = fabs ( gsse::math::determinant( gsse::fem::get_jacobian (ip) ) )
              * ip.weight;

 Matrix mx_BDB  = gsse::math::transpose(mx_B) * fac * mx_D * mx_B ;
 ...
}
```

Sample algorithms

Energy transport (for electrons)

```
(sum<edge>()
[ let(_x = Bern(edge_log<vertex>(equ_T_n)) / equ_T_n *-q/k_B *
   sum<vertex>() [ equ_pot ] + sum<vertex>() [ equ_T_n ]  )
   [
      equ_T_n / Bern(edge_log<vertex>(equ_T_n))  *
      sum<vertex>() [  equ_n *  equ_T_n * Bern( _x  )  ] *
      5/2 * k_B * k_B / q  * n_mob_s  * area / dist
   ]
]
- sum<edge>() [ sum<vertex>() [ equ_pot ] / dist * Jn  ]  *
  vol  + 3/2 * k_B * equ_n * (equ_T_n - T_lattice )/tau_n*vol
)  (vertex);
```

Code examples

```
// 0D
{
 typedef boost::mpl::map<
    boost::mpl::pair<gsse2_env::dimension,   boost::mpl::int_<0> >
  , boost::mpl::pair<gsse2_env::env_storage, double >
  > env_ct_1;

 typedef boost::mpl::map<
    boost::mpl::pair<gsse2_env::dimension,    boost::mpl::int_<0> >
  , boost::mpl::pair<gsse2_env::env_storage,  double >
  , boost::mpl::pair<gsse2_env::env_index_bs, long>
  > env_ct_2;

 typedef boost::mpl::map<
    boost::mpl::pair<gsse2_env::dimension,        boost::mpl::int_<0> >
  , boost::mpl::pair<gsse2_env::env_storage,      double >
  , boost::mpl::pair<gsse2_env::env_index_bs,     long>
  , boost::mpl::pair<gsse2_env::env_index_fs,     std::string>
  , boost::mpl::pair<gsse2_env::env_container_fs, gsse2_env::env_container_map >
  > env_ct_3;
}
```

Code examples II

```
// 1D
{
 typedef boost::mpl::map<
   boost::mpl::pair<gsse2_env::dimension,        boost::mpl::int_<1> >
 , boost::mpl::pair<gsse2_env::env_complex,      gsse2_env::complex_explicit >
 , boost::mpl::pair<gsse2_env::env_container_bs, gsse2_env::env_container_vector >
 , boost::mpl::pair<gsse2_env::env_container_fs, gsse2_env::env_container_map >
 > env_ct_1;

 typedef boost::mpl::map<
   boost::mpl::pair<gsse2_env::dimension,   boost::mpl::int_<1> >
 , boost::mpl::pair<gsse2_env::env_complex, gsse2_env::complex_implicit >
 > env_ct_2;
}
```

Code examples III

```
// 2D / nD
{
 typedef boost::mpl::map<
   boost::mpl::pair<gsse2_env::dimension,   boost::mpl::int_<2> >
 , boost::mpl::pair<gsse2_env::env_cell,    gsse2_env::cell_simplex >
 , boost::mpl::pair<gsse2_env::env_complex, gsse2_env::complex_explicit >
 > env_ct_1;
}
```